
OPTIMALIZÁCIA A APROXIMÁCIA

O čom je tento text

Text je prevažne komplilátorom z viacerých kníh (najmä [6],[11],[14],[15]), článkov, poznámok z prednášok a prezentácií.

Tento text vznikol ako študijný materiál ku kurzu
2-INF-221: Aproximácia optimalizačných problémov
na FMFI UK.

V teste je použitá grafika:

- unit sprites from the project "Battle for Wesnoth" <http://www.wesnoth.org>
- binder graphics from the project digstud <https://github.com/bitfragment/digstud.git>
- images of printers from manufacturers' web presentations

Obsah

1 Aproximačné algoritmy	1
2 Lineárne programovanie	7
2.1 Pár úvodných slov	7
2.2 Simplexová metóda	12
2.3 Zložitosť simplexového algoritmu	20
3 Zaokrúhľovanie lineárnych programov	27
3.1 Celocíselné programy a relaxácia: dobrý, zlý a škaredý	27
3.2 Deterministické zaokrúhľovanie	40
3.3 Randomizované zaokrúhľovanie	52
4 Dualita v lineárnom programovaní	57
4.1 Čo je dualita	57
4.2 MAX-FLOW–MIN-CUT očami duality	64
4.3 Primárno-duálna metóda	67
5 Elipsoidová metóda	85

1

Aproximačné algoritmy

Predpokladáme, že čitateľ pozná techniky návrhu efektívnych algoritmov a stretol sa s analýzou ich časovej zložitosti. Je tiež užitočné, ak má predstavu o základoch teórie zložitosti, aj keď nateraz nám bude stačiť, že za efektívne riešiteľné považujeme tie problémy, pre ktoré existuje polynomiálny algoritmus (t.j. algoritmus, ktorý pre každý vstup vráti správny výstup a jeho čas behu je ohraničený polynómom od veľkosti vstupu) a že existuje veľa problémov, pre ktoré žiadame takýto algoritmus nepoznáme.

V nasledujúcom texte pokračujeme z tohto východiska. Budeme sa, až na zopár výnimiek, zaoberať problémami, pre ktoré nepoznáme polynomiálny algoritmus, a napriek tomu by sme ich chceli nejak efektívne riešiť. Prijmeme tézu, že efektívne riešenie musí byť v polynomiálnom čase, a preto v ďalšom teste budeme, okrem prípadov, keď vyslovene povieme inak, že *algoritmus* považovať *algoritmus pracujúci v polynomiálnom čase*. Budeme skúmať jeden z možných prístupov: ak už nevieme spraviť algoritmus, ktorý vždy vráti správne riešenie, možno by sme sa dokázali uspokojiť s algoritmom, ktorý vždy nájde "takmer správne" riešenie. Toto, samozrejme, závisí od typu problému. Napr. pre rozhodovacie problémy (t.j. také, kde odpoved je *áno alebo nie*) je každá odpoveď "takmer správna". Veľa zaujímavých problémov sú ale *optimalizačné problémy*. Neformálne, pre každý vstup optimalizačného problému existuje množina tzv. *prípustných riešení*. Navyše, každé prípustné riešenie má istú *mieru* (cenu alebo zisk). Cielom je navrhnuť algoritmus, ktorý pre každý vstup nájde prípustné riešenie s optimálnou (minimálnou alebo maximálnou) mierou. Príkladom optimalizačného problému je problém batohu:

Definícia 1.1. Na vstupe je daných n vecí s cenami c_1, \dots, c_n a objemami v_1, \dots, v_n (pričom ceny aj objemy sú prirodzené čísla). Takisto je dané prirodzené číslo B , ktoré reprezentuje veľkosť batohu. Cieľom problému MAX-KNAPSACK je vybrať množinu vecí, ktoré sa zmestia do batohu a ich cena je maximálna, t.j. nájsť množinu $\mathcal{I} \subseteq \{1, 2, \dots, n\}$ takú, že $\sum_{i \in \mathcal{I}} v_i \leq B$ a maximalizuje sa $\sum_{i \in \mathcal{I}} c_i$ spomedzi všetkých množín \mathcal{I} .

Prípustné riešenia sú všetky výbery, ktoré sa zmestia do batohu, miera (zisk) riešenia je súčet cien vybratých vecí a cieľom je nájsť prípustné riešenie, ktoré maximalizuje zisk. MAX-KNAPSACK je príkladom problému, pre ktorý nepoznáme polynomiálny algoritmus.

Prvým pokusom v nami naznačenom smere by mohlo byť navrhnuť algoritmus, ktorý vždy nájde prípustné riešenie s cenou najviac o konštantu menšou ako optimálne riešenie, t.j. chceme mať algoritmus \mathbf{A} a konštantu c tak, že pre každý vstup x , \mathbf{A} nájde riešenie s cenou aspoň $OPT(x) - c$, kde $OPT(x)$ je cena optimálneho riešenia. O taktomto algoritme povieme, že má *absolútну chybu* c . Po krátkej úvahе ale zistíme, že takto postavený cieľ je rovnako ľahký ako pôvodný.

Veta 1.2. Ak existuje polynomiálny algoritmus pre MAX-KNAPSACK s absolútou chybou c , potom existuje aj polynomiálny exaktný algoritmus.

Dôkaz: Nech existuje taký algoritmus \mathbf{A} s absolútou chybou c . Zostrojíme algoritmus \mathbf{A}' , ktorý modifikuje daný vstup x tak, že objemy vecí aj veľkosť batohu ponechá, ale ceny prenásobí koeficientom $c + 1$. Na takto modifikovaný vstup x' potom použije algoritmus \mathbf{A} . Ten vráti riešenie, ktoré je prípustné pre x' a má cenu aspoň $OPT(x') - c$. Lenže v x' sú ceny všetkých vecí násobky $c + 1$, a teda jediná možná cena riešenia v intervale $(OPT(x') - c, OPT(x'))$ je $OPT(x')$. Teda \mathbf{A} musel nájsť optimálne riešenie pre vstup x' . Vidno ale, že prípustné riešenia problémov x a x' sú jednoznačne zodpovedajú, preto máme aj optimálne riešenie pre vstup x . \square

Z predchádzajúceho dôkazu vidno, v čom je problém absolútnej chyby: ak máme úlohu, v ktorej sa cena všetkých prípustných riešení dá rovnomerne "naťuknuť", vieme algoritmus s konštantou

absolútonu chybou prinútiť, aby našiel optimum. Túto ”nafukovaciu” vlastnosť má veľa problémov, ktoré nás budú zaujímať, a preto až na niekoľko výnimiek nebudem mať algoritmy s ohraničenou konštantou chybou.

Trieda APX

Druhý prístup k meraniu chyby, ktorý je v praxi častý, je merat *relatívnu chybu*, t.j. o akú pomernú časť sme sa pomýlili. Formálne to docielime tak, že absolútnu chybu preškálujeme tak, aby sme dostali číslo z intervalu $(0, 1)$

Definícia 1.3. Majme vstup x a prípustné riešenie y s cenou $c(y)$. Relatívna chyba riešenia y je

$$\mathcal{E}(x, y) = \frac{|OPT(x) - c(y)|}{\max\{OPT(x), c(y)\}}$$

Predchádzajúca definícia je zapísaná tak, aby zahŕňala maximalizačné aj minimalizačné problémy, ale pre názornosť je lepšie si ju rozpísat zvlášť. Pre maximalizačné problémy je relatívna chyba

$$\mathcal{E}_{\max}(x, y) = \frac{OPT(x) - c(y)}{OPT(x)} = 1 - \frac{c(y)}{OPT(x)}$$

a pre minimalizačné problémy

$$\mathcal{E}_{\min}(x, y) = \frac{c(y) - OPT(x)}{c(y)} = 1 - \frac{OPT(x)}{c(y)}$$

V oboch prípadoch vidno, že optimálny algoritmus vždy vráti riešenie s cenou $OPT(x)$, a preto relatívna chyba je 0. Zároveň, pretože budeme uvažovať iba problémy s nezápornými cenami, je relatívna chyba vždy menšia ako 1. Pre maximalizačné problémy má relatívnu chybu 1 triviálny algoritmus, ktorý vráti riešenie s hodnotou 0 (horšie to pri nezáporných cenách nemôže byť); pre minimalizačné problémy sa relatívna chyba blíži k 1 pre algoritmus, ktorého cena riešenia sa blíži k ∞ . Aby sme predišli technickým problémom, budeme predpokladať, že $\max\{OPT(x), c(y)\} > 0$. Ďalší patoogický prípad môže nastaviť, ak máme minimalizačný problém, ktorého optimum je 0, vtedy totiž každý algoritmus vracajúci kladné riešenie má relatívnu chybu 1. Namiesto toho, aby sme v definícii takéto prípady ošetrili, ostaneme pri tejto jednoduchšej verzii a slúbime si, že také problémy nebudem študovať. Za ”*aproximovateľné*“ budeme považovať problémy, pre ktoré vieme navrhnuť algoritmus s ohraničenou relatívnu chybou:

Definícia 1.4. Triedu APX tvoria optimalizačné problémy, pre ktoré existuje polynomiálny algoritmus A a konštantu ε , $0 < \varepsilon < 1$ taká, že na každom vstupe je relatívna chyba riešenia algoritmu A najviac ε .

V niektorých prípadoch je pracovať s relatívnu chybou trochu ťažkopádne, a preto sa používa ekvivalentný pojem *aproximačného pomeru*:

$$\mathcal{R}(x, y) = \frac{1}{1 - \mathcal{E}(x, y)}$$

Rozpísané zvlášť pre maximalizačné a minimalizačné problémy:

$$\mathcal{R}_{\max}(x, y) = \frac{OPT(x)}{c(y)} \quad \mathcal{R}_{\min}(x, y) = \frac{c(y)}{OPT(x)}$$

Vidíme, že approximačný pomer je vždy aspoň 1 a optimálny algoritmus má approximačný pomer 1. Algoritmus, ktorého approximačný pomer je najviac r budeme volať *r-aproximačný*. Pre minimalizačné problémy to znamená, že vždy vráti riešenie, ktorého cena je najviac r -násobkom optima, pre

maximalizačné problémy vždy vráti riešenie, ktorého cena je aspoň $1/r$ -násobok optima. Trieda APX je teda tvorená problémami, pre ktoré existuje r -aproximačný algoritmus s konštantným r .

Pozrime sa teraz, či sa nám podarí navrhnuť aproximačný algoritmus z triedy APX pre problém MAX-KNAPSACK. Máme teda dve úlohy: navrhnuť algoritmus a dokázať, že má konštantný aproximačný pomer. Druhá úloha je väčšinou tažšia, lebo potrebujeme porovnať riešenie nášho algoritmu s optimálnym riešením, ktoré nepoznáme. Ako ešte veľakrát v tomto texte uvidíme, jadrom dôkazu je najst prešíkaný spôsob, ako odhadnúť optimálne riešenie. V prípade problému batohu si môžeme zobrať modifikovaný problém, v ktorom povolíme veci krájať: z i -tej veci môžeme zobrať časť α_i . Dostávame nasledovný problém

Definícia 1.5. Na vstupe je daných n vecí s cenami c_1, \dots, c_n a objemami v_1, \dots, v_n (pričom ceny aj objemy sú prirodzené čísla). Takisto je dané prirodzené číslo B , ktoré reprezentuje veľkosť batohu. Cieľom problému MAX-Q-KNAPSACK je najst racionálne čísla $\alpha_1, \dots, \alpha_n$ tak, aby pre všetky i platilo $0 \leq \alpha_i \leq 1$ a

1. $\sum_i \alpha_i v_i \leq B$
2. $\sum_i \alpha_i c_i$ je maximálne možné

Riešenia problému MAX-KNAPSACK sú aj riešeniami problému MAX-Q-KNAPSACK; tým, že veci môžeme krájať sme množinu prípustných riešení rozšírili a optimálne riešenie sme možno zväčšili. Budeme hovoriť, že MAX-Q-KNAPSACK je zvoľnením (*relaxáciou*) MAX-KNAPSACK. Zároveň je problém MAX-Q-KNAPSACK ľahko riešiteľný greedy algoritmom A_g , ktorý utriedi veci podľa jednotkovej ceny c_i/v_i od najdrahšej po najlacnejšiu a ukladá ich do batohu (celé, t.j. s $\alpha_i = 1$) kým sa zmestia. Z prvej veci, ktorá sa nezmestí, zoberie takú časť, aby bol batoh celkom zaplnený (samozrejme, ak sa do batohu zmestia všetky veci, tak vezme všetky). Je jednoduchým cvičením presvedčiť sa, že A_g je optimálny: k ľubovoľnému inému prípustnému riešeniu vieme najst lepšie tak, že v batohu vymeníme kúsok nejakej veci za rovnako veľký kúsok drahšej nezobratej veci.

Rovnaký greedy algoritmus sa dá použiť aj na problém MAX-KNAPSACK (s tým, že do batohu ukladá veci, kym sa zmestia, a potom skončí), ale, žiaľ, nielen že nenájde optimum, ale ani dobrú approximáciu: uvažujme napr. veci s cenami $1, 1, \dots, 1, 2^n - 1$ a objemami $1, 1, \dots, 1, 2^n$, pričom batoh má veľkosť $B = 2^n$. Greedy algoritmus zoberie všetky jednotkové veci (majú jednotkovú cenu 1, kym posledná vec má jednotkovú cenu $1 - 1/2^n$) a skončí s riešením s cenou $n - 1$, pretože posledná vec sa do batohu nezmestí. Optimálne riešenie je ale zobrať poslednú vec, ktorá zaplní celý batoh, a zarobiť $2^n - 1$. Aproximačný pomer greedy algoritmu je teda aspoň $(2^n - 1)/(n - 1)$, čo v žiadnom prípade nie je konštanta pre rastúce n .

Urobme ešte zúfalý pokus na záchranu situácie: uvažujme algoritmus A , ktorý porovná riešenie získané greedy algoritmom a riešenie, ktoré zoberie jedinú vec s najväčšou cenou c_{\max} (bez ujmy na všeobecnosti predpokladáme, že každá vec sa sama osebe do batohu zmestí, ináč ju hned v predspracovaní vyhodíme) a vráti väčšie z tých dvoch.

V nasledujúcim dôkaze sa ukáže idea, ktorá sa neskôr bude v rôznych obmenach opakovať: optimálne riešenie relaxovaného problému (ktoré poznáme) použijeme ako horný odhad optima (ktoré nepoznáme).

Veta 1.6. Algoritmus A je 2-aproximačný.

Dôkaz. Potrebujeme ukázať, že pre každý vstup je aproximačný pomer algoritmu A najviac 2, t.j. že vždy vráti riešenie s cenou aspoň $OPT/2$, kde OPT je hodnota optimálneho riešenia. Majme veci utriedené podľa jednotkovej ceny a nech j je index prvej veci, ktorá sa nezmestí celá do batohu, t.j. optimálne riešenie MAX-Q-KNAPSACK má hodnotu $\sum_{i=1}^{j-1} c_i + \alpha c_j$ pre nejaké α , $0 < \alpha \leq 1$. Označme $\bar{c}_j = \sum_{i=1}^{j-1} c_i$. Pretože MAX-Q-KNAPSACK je relaxáciou MAX-KNAPSACK, je $OPT \leq \bar{c}_j + \alpha c_j \leq \bar{c}_j + c_j$. Rozlíšime dva prípady:

- Ak $\bar{c}_j \leq c_j \leq c_{\max}$, máme $OPT \leq 2c_{\max}$

- Ak $\bar{c}_j > c_j$, máme $OPT < 2\bar{c}_j$

Pretože riešenie, ktoré nájde algoritmus A má cenu $\max\{\bar{c}_j, c_{\max}\}$, je dôkaz hotový. \square

Lepšie ako APX: PTAS a FPTAS

V predchádzajúcej časti sme predstavili triedu APX ako triedu "aproximovateľných" problémov. Ako je to s praktickým využitím takýchto algoritmov? Problémom tu je slovíčko "existuje konštanta" z Definície 1.4. Podobne, ako polynomiálny algoritmus, ktorého čas je ohraničený polynomom n^{234} , aj approximačný algoritmus, ktorý vždy vráti 856-násobok optima, nie je v praxi použiteľný. Podobne, ako pri stupni polynómu, aj pri approximačnom pomere je tažké (a nepraktické) stanoviť konštantu, nad ktorou už príslušný approximačný pomer nie je zaujímavý. Občas sa nám ale podarí tento problém obistiť elegantne a navrhnuť tzv. *approximačné schému* (PTAS, polynomial-time approximation scheme):

Definícia 1.7. Triedu PTAS tvoria optimalizačné problémy, pre ktoré pre každú fixnú konštantu $0 < \varepsilon < 1$ existuje polynomiálny algoritmus A_ε , ktorý na každom vstupe vráti riešenie s relatívou chybou najviac ε .

Ako uvidíme o chvíľu, v tejto definícii je kľúčové, že ε je fixná konšstanta, t.j. čas algoritmu A_ε musí byť polynomiálny od veľkosti vstupu, ale môže (aj nepolynomiálne) závisieť od ε . Uvažujme problém MIN-PARTITION. V rozhodovacej verzii je cieľom zistiť, či sa prirodzené čísla zo vstupej dajú rozdeliť na dve časti s rovnakým súčtom. V optimalizačnej verzii sa budeme snažiť nájsť "čo najpodobnejšie" časti, pričom je niekoľko spôsobov, ako definovať "čo najpodobnejšie". Keby sme napríklad povedali, že chceme minimalizovať rozdiel väčšej a menšej časti, dostali by sme minimalizačný problém, ktorého optimum môže byť 0, a také sme si slíbili neuvažovať. Zvolme preto inú definíciu, v ktorej sa snažíme minimalizovať veľkosť väčšej časti:

Definícia 1.8. Na vstupe je daných n prirodzených čísel a_1, \dots, a_n . Úlohou problému MIN-PARTITION je nájsť rozklad množiny $\{1, \dots, n\}$ na dve dizjunktné množiny Y_1, Y_2 tak, aby cena

$$\max \left\{ \sum_{i \in Y_1} a_i, \sum_{i \in Y_2} a_i \right\}$$

bola najmenšia možná.

Podobne ako v predchádzajúcim príklade, lepšie sa nám bude pracovať s approximačným pomerom namiesto relatívnej chyby. Naším cieľom bude pre ľubovoľné fixné $r > 1$ navrhnúť r -approximačný algoritmus. Pretože každé riešenie má veľkosť aspoň $1/2 \sum_{i=1}^n a_i$, pre $r \geq 2$ stačí vrátiť rozdelenie $Y_1 = \emptyset, Y_2 = \{1, \dots, n\}$. Predpokladajme preto, že $r < 2$. Pozrime sa najprv na jednoduchý "greedy" algoritmus, ktorý utriedi čísla od najväčšieho po najmenšie, a potom postupne vždy pridáva do tej časti, ktorá je práve menšia. Lahko vidno, že takýto algoritmus nemôže mať approximačný pomer lepší ako $7/6$: pre vstup $3, 3, 2, 2, 2$, ktorého optimum je 6 ($3+3, 2+2+2$), greedy algoritmus vráti rozdelenie $3 + 2, 3 + 2 + 2$ s hodnotou 7. Budeme vychádzať z intuície, že najdôležitejšie je správne rozdeliť veľké čísla, lebo pri ich nesprávnom uložení vznikne väčšia chyba. Pri malých číslach, naopak, ani greedy algoritmus veci príliš nepokazí. Navrhnite preto nasledovný algoritmus A:

- 1: vstup: čísla $a_1 \geq a_2 \geq \dots \geq a_n$, a $r, 1 < r < 2$
- 2: $k := \alpha$
- 3: nájdi optimálne riešenie Y_1, Y_2 pre a_1, \dots, a_k
- 4: **for** $i = k + 1$ to n **do**
- 5: pridaj i do množiny Y_ℓ s menšou hodnotou $\sum_{j \in Y_\ell} a_j$

6: end for

Algoritmus A prvých k čísel rozloží optimálne (napr. preskúšaním všetkých 2^k možností) a zvyšok dorobí greedy algoritmom. Klúčom k riešeniu je vhodne zvoliť parameter k . Pozrite sa, aký aproximáčný pomer A dosahuje pri zvolenom parametri k . Zavedme najprv niekoľko označení. Nech $W_1 = \sum_{j \in Y_1} a_j$ a $W_2 = \sum_{j \in Y_2} a_j$. Bez ujmy na všeobecnosti predpokladajme $W_1 > W_2$, takže A nájde riešenie s hodnotou W_1 . Ďalej nech $L = (W_1 + W_2)/2$, t.j. L je dolné ohraničenie optimálneho riešenia. Nech h je maximálny index prvku v Y_1 .

Ak $h \leq k$, tak všetky prvky sa do Y_1 dostali na riadku 3 a v cykle na riadku 5 sa nepridá žiadny pravok.

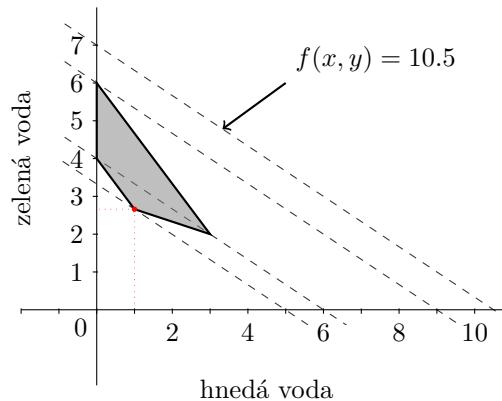
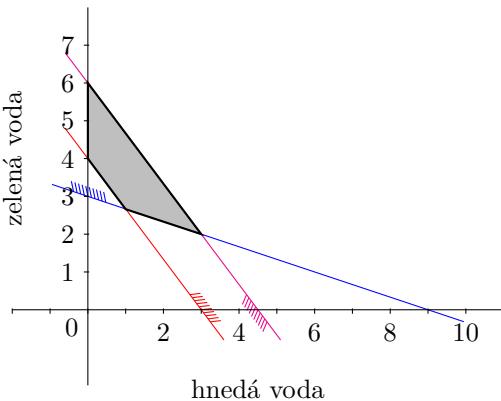
TODO: *FPTAS Knapsack, $O(\log n)$ approximácia Set Cover*

2.1 Pár úvodných slov

Začneme typickým príkladom, ktorý sa v tej či onej modifikácii objaví v každom teste o lineárnom programovaní. Zoberme si študenta, ktorý na prípravu na skúšku potrebuje dostať do tela aspoň 270 mg kofeínu a 120 g cukru. Zároveň nechce prekročiť dávku 180 mg aspartámu. K dispozícii má dva nápoje: hnedú vodu za €1/dl a zelenú vodu za €1,50/dl. Hnedá voda obsahuje 30 mg kofeínu, 40 g cukru a 40 mg aspartámu, kým zelená voda obsahuje 90 mg kofeínu, 30 g cukru a 30 mg aspartámu. Koľko dl ktorej vody si má študent kúpiť, aby čo najlacnejšie uspokojil svoje požiadavky? Ak si študent kúpi x dl hnedej vody a y dl zelenej vody, úlohu (nazývanú *lineárny program*) môžme formulovať takto:

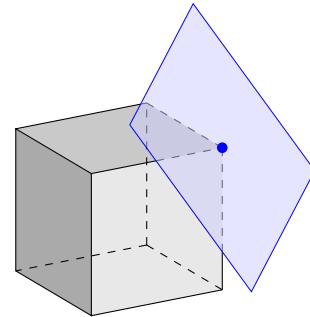
$$\begin{array}{llllll}
 & & \text{dl vody} & & & \\
 & \text{hnedej} & & \text{zelenej} & & \\
 \text{minimalizovať} & x & + & 1.5y & =: f(x, y) & \text{cena} \\
 \text{pri obmedzeniach} & 30x & + & 90y & \geq 270 & \text{kofeín} \\
 & 40x & + & 30y & \geq 120 & \text{cukor} \\
 & 40x & + & 30y & \leq 180 & \text{aspartám} \\
 & x, y & \geq 0 & & &
 \end{array} \tag{1}$$

Lahko vidno, že napr. 4 dl zelenej vody uspokoja všetky požiadavky za cenu €6, pričom kofeínu je aj viac, ako je nutné a aspartámu menej, ako je dovolené. Na druhej strane, ak by študent chcel kupovať iba hnedú vodu, na splnenie potreby kofeínu by jej musel kúpiť 9 dl, čím by ale prekročil prípustný príjem aspartámu (a navyše by to bolo drahé). Pri hľadaní optimálneho riešenia pomôže nasledovná geometrická reprezentácia: ak riešeniu s x dl hnedej a y dl zelenej vody priradíme bod v rovine so súradnicami (x, y) , každé z obmedzení určuje polovinu, v ktorej prípustné riešenie musí ležať. Do úvahy preto prichádzajú iba riešenia v štvoruholníku z ľavého obrázka:



Zároveň vieme, že $f(x, y) = x + 1.5y$ je lineárna funkcia, a preto body s rovnakou hodnotou funkcie f ležia na priamke (vid. pravý obrázok). Preto je zrejmé, že stačí overiť štyri vrcholy štvoruholníka a nájdeme optimálne riešenie, ktoré je v tomto prípade kúpiť 1 dl hnedej vody a $2\frac{2}{3}$ dl zelenej vody za €5.

Z týchto úvah ľahko odvodíme algoritmus na riešenie úlohy s dvoma premennými a obmedzeniami s nerovnosťami: pre každé obmedzenie zostrojíme príslušnú polrovinu, skonštruujeme mnohouholník, ktorý je prienikom všetkých polrovín a vyberieme optimálnu hodnotu spomedzi jeho vrcholov. Pre tri premenné obmedzenia tvaru $a_1x + a_2y + a_3z \geq b$ tvoria polpriestory, ktorých prienikom dostaneme mnohostenu. Body s rovnakou hodnotou funkcie $f(x, y, z) = c_1x + c_2y + c_3z$ tvoria rovinu a preto na nájdenie optimáma stačí overiť všetky vrcholy mnohostena.



S narastajúcim počtom premenných začnú rásť aj problémy a je zrejmé, že priamočiarym zovšeobecňovaním sa ďaleko nedostaneme. Skúsme sa preto pozrieť na geometriu lineárneho programu trochu inak. Najprv si upravme program (1) do ekvivalentnej podoby takto: funkciu $f(x, y)$ vynásobíme -1 a z minimalizačnej úlohy dostaneme maximalizačnú. Potom prvé a druhé obmedzenie vynásobíme -1 a všetky obmedzenia (okrem tých na nezápornosť x, y) budú v tvare " \leq ". Nakoniec, keďže v každom obmedzení je ľavá strana menšia ako pravá, môžeme pridať novú nezápornú premennú, ktorej hodnota bude práve rozdiel ľavej a pravej strany a dostaneme nasledujúci program, ktorý je očividne ekvivalentný s pôvodným.

$$\begin{array}{lllll} \text{maximalizovať} & -x & -1.5y & & =: f(x, y, s_1, s_2, s_3) \\ \text{pri obmedzeniach} & -30x & -90y & +s_1 & = -270 \\ & -40x & -30y & +s_2 & = -120 \\ & 40x & +30y & +s_3 & = 180 \\ & x, y, s_1, s_2, s_3 & \geq 0 & & \end{array} \quad (2)$$

Ak si označíme

$$\mathbf{c} = \begin{pmatrix} -1 \\ -1.5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} -30 & -90 & 1 & 0 & 0 \\ -40 & -30 & 0 & 1 & 0 \\ 40 & 30 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} -270 \\ -120 \\ 180 \end{pmatrix}$$

tak program (2) môžeme skrátene zapísť

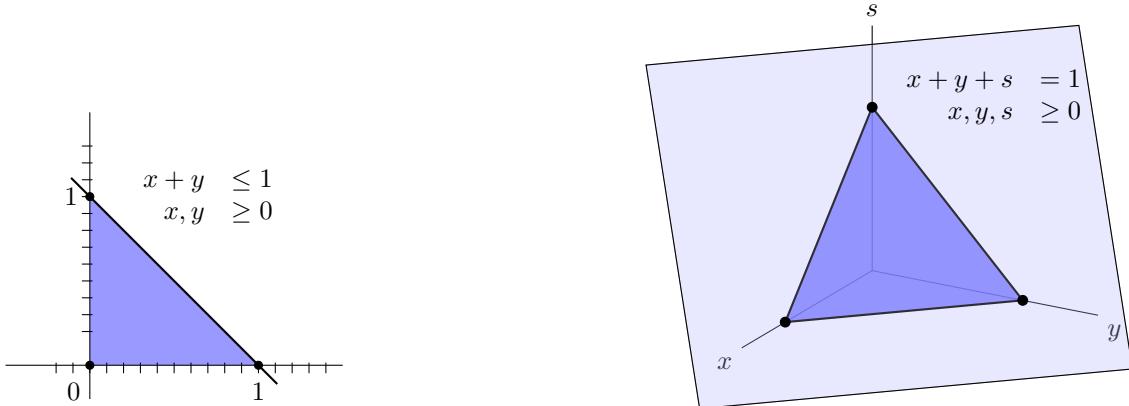
$$\max_{\mathbf{x} \in \mathbb{R}^5} \{ \mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0 \}.$$

Týmto prepísaním sme zvýšili dimenziu problému z 2 na 5 (a tak sme stratili možnosť "obrázkovo" riešenia), ale získali sme krajsí tvar množiny prípustných riešení: sú to nezáporné riešenia systému lineárnych rovníc. V našom prípade má matica A hodnosť 3 (riadky sú lineárne nezávislé) a riešenia systému $A\mathbf{x} = \mathbf{b}$ tvoria dvojrozmerný podpriestor $\mathbf{o} + \mathbf{c}_1\boldsymbol{\alpha} + \mathbf{c}_2\boldsymbol{\beta}$ kde

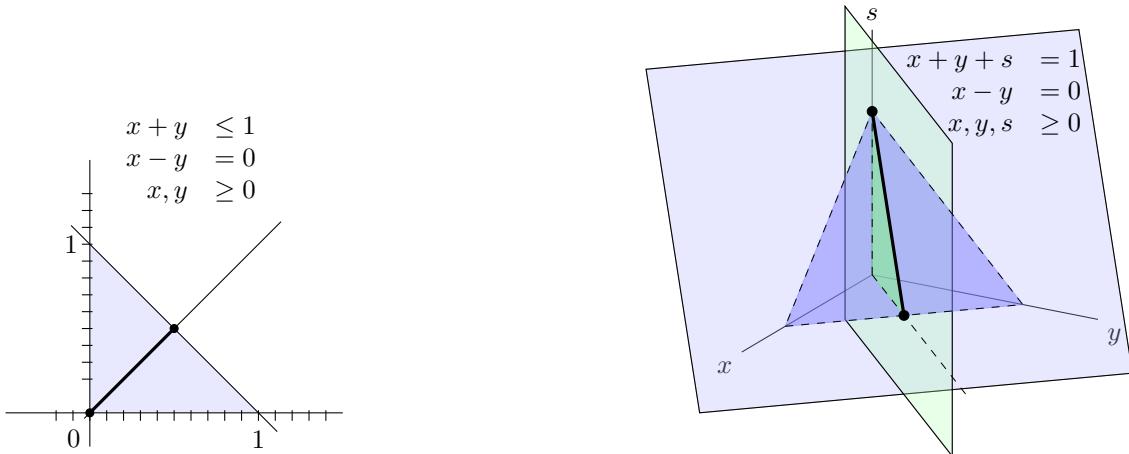
$$\mathbf{o} = \begin{pmatrix} 0 \\ 0 \\ -270 \\ -120 \\ 180 \end{pmatrix} \quad \boldsymbol{\alpha} = \begin{pmatrix} 1 \\ 0 \\ 30 \\ 40 \\ -40 \end{pmatrix} \quad \boldsymbol{\beta} = \begin{pmatrix} 0 \\ 1 \\ 90 \\ 30 \\ -30 \end{pmatrix}.$$

Inými slovami, prípustné riešenia programu (1) tvoria 2-rozmerný útvor (štvrúholník) v dvojrozmernom priestore (rovine), kým prípustné riešenia programu (2) tvoria 2-rozmerný útvor \mathcal{D} v 5-rozmernom priestore, pričom \mathcal{D} je prienikom (2-rozmernej) roviny a kladného ortantu¹. Pre ilustráciu uvažujme nasledovné jednoduchšie programy (uvádzame iba obmedzenia, na účelovej funkcií v tomto prípade nezáleží):

¹ako je kvadrant v rovine a oktant v 3D priestore, používame slovo ortant v n -rozmernom priestore



Vľavo je program s dvoma premennými a dvojrozmerným priestorom riešení. Po transformácii do troch rozmerov priestor riešení ostal dvojrozmerný útvár a je prienikom roviny a kladného oktantu. V nasledujúcom prípade je v pôvodnom probléme priestor riešení jednorozmerný; po transformácii do trojrozmerného priestoru je priestor riešení stále jednorozmerný a je prienikom priamky a kladného oktantu.



Výhoda takto zapísaného programu je v tom, že sa nám budú ľahšie hľadať vrcholy telesa prípustných riešení: budú určite ležať v nejakej stene tvaru $x_i = 0$.

Skôr, ako pokročíme v našich úvahách je dobré si uvedomiť, že bez ujmy na všeobecnosti môžeme predpokladať, že riadky matice A sú lineárne nezávislé: ak je nejaký riadok lineárnej kombináciou iných riadkov potom buď neexistuje žiadne prípustné riešenie, alebo jeho odstránením nijak nezmeníme priestor prípustných riešení. Naše úvahy môžeme zhrnúť a zaviesť *normálny tvar* lineárneho programu takto:

Definícia 2.1. Lineárny program je v **normálnom tvare**, ak je zapísaný ako

$$\max_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \}, \quad (3)$$

kde $A \in \mathbb{R}^{m \times n}$ má hodnosť m . Množina prípustných hodnôt $\mathcal{D} = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$.

- **Cieľom je maximalizácia.** Ak bolo pôvodným cieľom minimalizovať lineárnu funkciu $f(\mathbf{x})$, novým cieľom bude maximalizovať lineárnu funkciu $-f(\mathbf{x})$.
- **Všetky premenné sú nezáporné.** Pre každú premennú x , ktorá nemá obmedzenie $x \geq 0$ pridáme dve premenné $p_x, q_x \geq 0$ a každý výskyt x nahradíme $p_x - q_x$.
- **Každé obmedzenie okrem tých tvaru $x \geq 0$ má tvar rovnosti.** Ak bolo pôvodné ob-

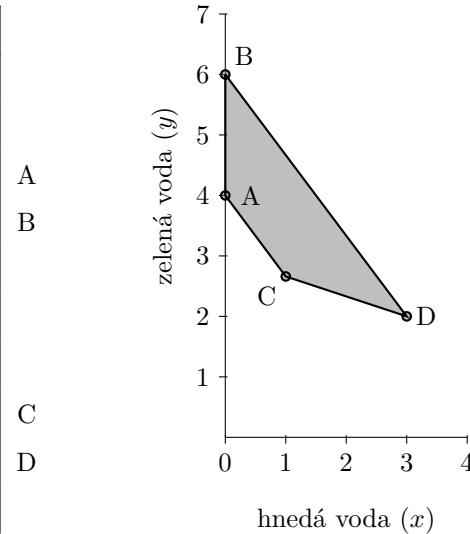
medzenie v tvare $\sum_i a_i x_i \geq b$, najprv ho prenásobením -1 upravíme na tvar $\sum_i -a_i x_i \leq -b$. Ked máme všetky nerovnosti v obmedzeniach otočené rovnakým smerom, pre každé obmedzenie tvaru $\sum_i a_i x_i \leq b$ zavedieme novú premennú s (*rezerva, slack*), ktorá reprezentuje hodnotu, o kolko je pôvodná ľavá strana menšia ako b . Lahko vidno, že $s \geq 0$ a $s + \sum_i a_i x_i = b$.

- Matica A má m lineárne nezávislých riadkov.

Majme teraz program v normálnom tvari. Podobne ako v úvodnom príklade, chceme nájsť množinu vrcholov telesa \mathcal{D} ohraničujúceho prípustné riešenia tak, aby stačilo overiť tieto vrcholy na nájdenie optimálneho riešenia. Kedže \mathcal{D} je prienikom priestoru riešení $Ax = \mathbf{b}$ a kladného ortantu, vrcholy budú mať jednu alebo niekoľko súradník x_{i_1}, \dots, x_{i_k} nulových (nadrovina $x_i = 0$ je na hranici ortantu). Navyše, vrcholy sú “špicaté” (na rozdiel od bodov v nejakej stene $x_i = 0$, v ktorej leží napr. celá úsečka). To, že vrchol je “špicatý”, formulujeme tak, že ak k obmedzeniam $Ax = \mathbf{b}$ pridáme navyše $x_{i_1} = 0, \dots, x_{i_k} = 0$, dostaneme jednoznačné riešenie (t.j. vrchol leží na prieniku nejakých stien ortantu a žiadnen iný bod v tom istom prieniku neleží). Kedže A má hodnosť m , na to, aby sme dostali jednoznačné riešenie, musí byť $k = n - m$.

Vráťme sa teraz k programu (2). Hodnosť matice A je 3, preto vrcholy dostaneme tak, že k systému $Ax = \mathbf{b}$ pridáme dve obmedzenia $x_i = 0$ a $x_j = 0$. Dostávame 10 rôznych systémov lineárnych rovníc s nasledovnými riešeniami:

obmedzenia	x	y	s_1	s_2	s_3
$x = y = 0$	0	0	-270	-120	180
$x = s_1 = 0$	0	3	0	-30	90
$x = s_2 = 0$	0	4	90	0	60
$x = s_3 = 0$	0	6	270	60	0
$y = s_1 = 0$	9	0	0	240	-180
$y = s_2 = 0$	3	0	-180	0	60
$y = s_3 = 0$	4.5	0	-135	60	0
$s_1 = s_2 = 0$	1	$\frac{8}{3}$	0	0	60
$s_1 = s_3 = 0$	3	2	0	60	0
$s_2 = s_3 = 0$					neexistuje riešenie



Riešenia systému $Ax = \mathbf{b}$ tvoria rovinu v 5-rozmernom priestore. Premenné x, y zodpovedajú množstvu kúpenej hnedej a zelenej vode, premenné s_1, s_2, s_3 udávajú rezervu, ktorá ostáva k prekročeniu príslušného obmedzenia. Takže napríklad štvrtý riadok tabuľky s obmedzeniami $x = s_3 = 0$ hovorí, že ak študent nekupuje žiadnu hnédú vodu a zároveň chce presne dosiahnuť povolené množstvo aspartámu, musí kúpiť 6 dl zelenej vody, pričom dostane viac kofénu a cukru ako potrebuje. V poslednom riadku vidno, že obmedzenia sa nedajú pridávať ľubovoľne, ale treba dávať pozor, aby pridaním obmedzení nevznikli lineárne závislé riadky (v našom prípade sú červená a fialová priamka z prvého obrázka rovnobežné, takže neexistuje žiadnen bod v ich priesecníku). V reči lineárnych programov sa riadkom tejto tabuľky hovorí *bázové riešenia* a tie z nich, ktoré sú zároveň prípustné (zvýraznené riadky) sú *prípustné bázové riešenia* a zodpovedajú vrcholom \mathcal{D} , t.j. prípustné riešenia programu (2) tvoria dvojrozmerný štvoruholník v päťrozmernom priestore. Za povšimnutie stojí, že prípustné bázové riešenia programu (2), t.j. vrcholy štvoruholníka prípustných riešení v päťrozmernom priestore, zodpovedajú vrcholom štvoruholníka prípustných riešení programu (1): každý vrchol štvoruholníka vpravo leží na priesecníku dvoch priamok, z ktorých každá zodpovedá obmedzeniu tvaru $x_i = 0$ (a príslušné bázové riešenie je prípustné). Naopak, každé prípustné bázové riešenie leží na priesecníku dvoch takýchto priamok.

Keď si uvedomíme, že pridaním obmedzenia tvaru $x = 0$ vlastne pri riešení príslušného systému vymažeme stĺpec premennej x z matice dostaneme nasledovnú definíciu.

Označenie. Majme maticu $A \in \mathbb{R}^{m \times n}$ s m riadkami a n stĺpcami. Pre množinu $B \subseteq \{1, 2, \dots, n\}$ označíme A_B podmaticu A , ktorá pozostáva zo stĺpcov indexovaných množinou B . Rovnakú notáciu \mathbf{x}_B budeme používať pre vektory.

Napríklad

$$A = \begin{pmatrix} -30 & -90 & 1 & 0 & 0 \\ -40 & -30 & 0 & 1 & 0 \\ 30 & 40 & 0 & 0 & 1 \end{pmatrix} \quad A_{\{1,2\}} = \begin{pmatrix} -30 & -90 \\ -40 & -30 \\ 30 & 40 \end{pmatrix} \quad A_{\{3,4,5\}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Definícia 2.2. Majme lineárny program v normálnom tvare, kde $A \in \mathbb{R}^{m \times n}$. **Bázové riešenie** je vektor $\mathbf{x} \in \mathbb{R}^n$, pre ktorý existuje m -prvková množina $B \subseteq \{1, \dots, n\}$ taká, že

1. matica $A_B \in \mathbb{R}^{m \times m}$ má hodnosť m (t.j. je regulárna)
2. $x_j = 0$ pre všetky $j \notin B$

Teraz ukážeme, že naša predstava bázového riešenia ako vrchola je dobrá v tom, že na nájdenie optima stačí overiť prípustné bázové riešenia.

Veta 2.3. Majme daný lineárny program v normálnom tvare, pričom hodnota účelovej funkcie je $\mathbf{c}^\top \mathbf{x}$ na telesie \mathcal{D} je zhora ohraničená. Potom pre každé prípustné riešenie \mathbf{x}_0 existuje prípustné bázové riešenie \mathbf{x} , pre ktoré $\mathbf{c}^\top \mathbf{x} \geq \mathbf{c}^\top \mathbf{x}_0$.

Dôkaz: Zoberme si ľubovoľné prípustné riešenie \mathbf{x}_0 a uvažujme všetky také prípustné riešenia \mathbf{x} , pre ktoré $\mathbf{c}^\top \mathbf{x} \geq \mathbf{c}^\top \mathbf{x}_0$. Za \mathbf{x} vyberme také z nich, ktoré má najväčší počet nulových zložiek. Ukážeme, že \mathbf{x} je bázové. Ak $\mathbf{x} = \mathbf{0}$, je zrejmé bázové. Nech teda \mathbf{x} má aspoň jednu nenulovú zložku. Označme $K = \{j \in \{1, \dots, n\} \mid \tilde{x}_j > 0\}$ množinu kladných (žiadne prípustné riešenie nemá záporné zložky) zložiek vektora \mathbf{x} a uvažujme dva prípady.

- Stĺpce matice A_K sú lineárne nezávislé. Zjavne $|K| \leq m$ (matica A má m riadkov). Ak $|K| = m$, \mathbf{x} v zhode s Definíciou 2.2 má $\tilde{x}_j = 0$ pre všetky $j \notin K$ a matica A_K je regulárna. Ak $|K| < m$, môžeme $|K|$ stĺpcov matice A_K doplniť $m - k$ stĺpcami z A tak, aby boli lineárne nezávislé². Takže dostaneme množinu K' tak, že $|K'| = m$, $A_{K'}$ je regulárna a $\tilde{x}_j = 0$ pre všetky $j \notin K' \supseteq K$.

- Stĺpce matice A_K sú lineárne závislé, to znamená, že existuje vektor $\vartheta \in \mathbb{R}^{|K|}$ taký, že $A_K \vartheta = \mathbf{0}$ (ϑ určuje lineárnu kombináciu stĺpcov A_K , ktorej výsledkom je nulový vektor). Doplňme ϑ na n -rozmerný vektor \mathbf{w} tak, že na miesta mimo množinu K dosadíme 0, takže $\mathbf{w}_K = \vartheta$ a $A\mathbf{w} = \mathbf{0}$. Pre ľubovoľné reálne $t \geq 0$ označme $\mathbf{x}(t) = \mathbf{x} + t\mathbf{w}$. Kedže \mathbf{x} je prípustné riešenie, platí $A\mathbf{x} = \mathbf{b}$. Zároveň platí $A\mathbf{w} = \mathbf{0}$ a teda aj $A\mathbf{x}(t) = \mathbf{b}$.

Prv, než budeme pokračovať v dôkaze, upravíme vektor \mathbf{w} tak, aby $\mathbf{c}^\top \mathbf{w} \geq 0$ a zároveň $w_j < 0$ pre nejaké $j \in K$. Ak $\mathbf{c}^\top \mathbf{w} = 0$ a pre všetky $j \in K$ platí $w_j > 0$, stačí \mathbf{w} prenásobiť -1 a máme ho v požadovanom tvare. Nech teda $\mathbf{c}^\top \mathbf{w} \neq 0$. Ak $\mathbf{c}^\top \mathbf{w} < 0$, môžeme opäť \mathbf{w} prenásobiť -1, takže bez ujmy na všeobecnosti nech $\mathbf{c}^\top \mathbf{w} > 0$. Ukážeme, že teraz musí také existovať $j \in K$, že $w_j < 0$. Ak by to tak nebolo, t.j. ak pre všetky $j \in K$ je $w_j > 0$, zjavne $\mathbf{w} \geq \mathbf{0}$ (zložky $i \notin K$ sme doplnili nulami). Potom ale $\mathbf{x}(t) = \mathbf{x} + t\mathbf{w} \geq 0$ pre všetky $t \geq 0$, takže $\mathbf{x}(t)$ je prípustné riešenie. Hodnota účelovej funkcie je $\mathbf{c}^\top \mathbf{x}(t) = \mathbf{c}^\top \mathbf{x} + t\mathbf{c}^\top \mathbf{w}$. Kedže $\mathbf{c}^\top \mathbf{w} > 0$, pre $t \mapsto \infty$ je $\mathbf{c}^\top \mathbf{x}(t) \mapsto \infty$, a teda lineárny program nebol ohraničený.

Majme teraz vektor \mathbf{w} upravený tak, že spĺňa $\mathbf{c}^\top \mathbf{w} \geq 0$ a zároveň $w_j < 0$ pre nejaké $j \in K$. Ukážeme, že pre nejaké $t_1 > 0$ je vektor $\mathbf{x}(t_1)$ prípustné riešenie s viacerými nulovými zložkami ako \mathbf{x} . To bude ale v spore s tým, že \mathbf{x} má najviac nulových zložiek spomedzi všetkých prípustných riešení \mathbf{x} , pre ktoré $\mathbf{c}^\top \mathbf{x} \geq \mathbf{c}^\top \mathbf{x}_0$, pretože $\mathbf{c}^\top \mathbf{x}(t_1) = \mathbf{c}^\top \mathbf{x} + t_1 \mathbf{c}^\top \mathbf{w} \geq \mathbf{c}^\top \mathbf{x}_0$ (lebo $\mathbf{c}^\top \mathbf{x} \geq \mathbf{c}^\top \mathbf{x}_0$ a $\mathbf{c}^\top \mathbf{w} \geq 0$).

²Toto tvrdenie je súčasťou základného kurzu algebry.

Vektor $\mathbf{x}(t_0) = \mathbf{x}$ je prípustné riešenie a má zložky $j \in K$ (ostro) kladné a zvyšné zložky nulové. Zároveň vieme, že existuje aspoň jedno $j \in K$, kde $w_j < 0$. Keďže j -ta zložka $\mathbf{x}(t)$ je $x(t)_j = \tilde{x}_j + tw_j$, s rastúcim t klesajú hodnoty $x(t)_j$ pre všetky j , kde $w_j < 0$. Zvolme za t_1 také t , keď prvá z hodnôt $x(t)_j$ dosiahne 0. Zjavne $\mathbf{x}(t_1)$ je prípustné riešenie a má viac nulových zložiek ako \mathbf{x} . \square

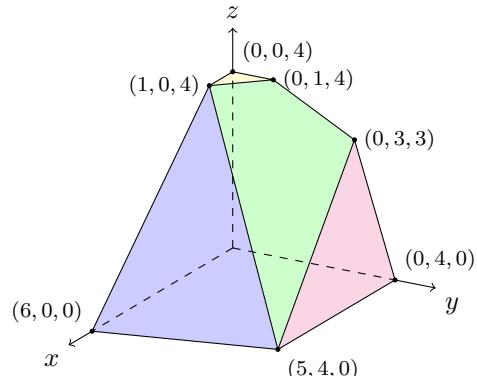
Dôsledkom tejto vety je, že na nájdenie optimálneho riešenia pre lineárne programy, ktoré majú konečné optimum, stačí prehľadať všetky prípustné bázové riešenia. Toto prehľadávanie je zo všeobecnením prístupu v dvoch rozmeroch z úvodného príkladu, kde stačilo prehľadať vrcholy vhodného mnohouholníka. Ako sa dajú bázové riešenia nájsť? Stačí si uvedomiť, že pre danú množinu $B \subseteq \{1, \dots, n\}$ existuje najviac jedno bázové riešenie³: ak by boli dve bázové riešenia \mathbf{y}, \mathbf{z} s tou istou množinou B , musí platiť $A\mathbf{y} = A\mathbf{z} = \mathbf{b}$ a teda $A_B\mathbf{y} = A_B\mathbf{z} = \mathbf{b}$. Keďže A je regulárna štvorcová matica, systém $A_B\mathbf{x} = \mathbf{b}$ má jednoznačné riešenie a preto $\mathbf{y} = \mathbf{z}$. Preto stačí vyskúšať všetky množiny B , overiť, či príslušná A_B je regulárna (napr. Gaussovou elimináciou), overiť, či je získané riešenie $A_B\mathbf{x} = \mathbf{b}$ prípustné a spomedzi všetkých takto získaných riešení \mathbf{x} vybrať to najlepšie. Problém s týmto algoritmom je, že pri n premenných a m obmedzeniach môže byť potenciálne $\binom{n}{m}$ rôznych bázových riešení, a teda vo všeobecnosti nie je polynomiálny⁴. V nasledujúcej časti si ukážeme, ako úlohu lineárneho programovania riešiť efektívnejšie pomocou simplexovej metódy.

2.2 Simplexová metóda

Ako simplexový algoritmus sa označuje každý taký greedy algoritmus, ktorý prechádza bázové riešenia (vrcholy \mathcal{D}) tak, že vždy sa presunie po hrane smerom, v ktorom rastie (pri maximalizácii) hodnota účelovej funkcie. Začnime opäť príkladom. Uvažujme nasledovný lineárny program:

$$\begin{array}{lll} \text{maximalizovať} & x + y + z & =: f(x, y, z) \\ \text{pri obmedzeniach} & x + y + 2z & \leq 9 \\ & 4x + y + 5z & \leq 24 \\ & 3y + z & \leq 12 \\ & z & \leq 4 \\ & x, y, z & \geq 0 \end{array} \quad (4)$$

Obmedzenia tvoria polpriestory v trojrozmernom priestore. Hraničné roviny (v poradí zelená, modrá, červená a žltá) vymedzujú mnohosten \mathcal{D} . Preskúšaním všetkých vrcholov \mathcal{D} zistíme, že maximum sa dosahuje v bode $(5, 4, 0)$.



Podobne ako v predchádzajúcej časti si zavedieme rezervné (*slack*) premenné s_1, \dots, s_4 ; ak si premenné očísľujeme v poradí x, y, z, s_1, \dots, s_4 , dostaneme ekvivalentný program v normálnom tvare

$$\max_{\mathbf{x} \in \mathbb{R}^7} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0 \}$$

³Naopak to neplatí, to isté bázové riešenie \mathbf{x} je možné dostať z rôznych množín B, B' . Ak napríklad vektor $\mathbf{0}$ je prípustné riešenie, potom je aj prípustné bázové riešenie pre ľuboľovnú množinu B .

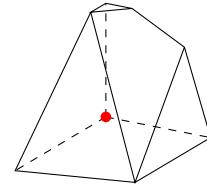
⁴Napríklad pre $m = n/2$ sa zo Stirlingovej approximácie $n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n (1 + o(n))$ ľahko ukáže, že $\binom{n}{\frac{n}{2}} = \frac{n!}{\left[\left(\frac{n}{2}\right)!\right]^2} \geq 2^n/n^2$.

kde

$$\mathbf{c} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 4 & 1 & 5 & 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 9 \\ 24 \\ 12 \\ 4 \end{pmatrix}$$

Máme 7 premenných a matica A má hodnosť 4, teda riešenia systému $A\mathbf{x} = \mathbf{b}$ tvoria trojrozmerný podpriestor v sedemrozmernom priestore. Špeciálne, môžeme si vybrať ľubovoľné tri premenné ako parametre a ostatné premenné (aj hodnotu funkcie) vyjadriť pomocou nich. V našom prípade je matica $A_{4,5,6,7}$ diagonálna, takže ľahko vidno, že program (4) je ekvivalentne zapísateľný takto:

$$\begin{array}{rccccc} f = & x & + y & + z & & \\ \hline s_1 = & 9 & -x & -y & -2z & \\ s_2 = & 24 & -4x & -y & -5z & \\ s_3 = & 12 & & -3y & -z & \\ s_4 = & 4 & & & -z & \end{array} \quad (5)$$



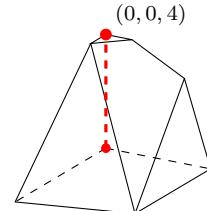
Zápisu (5) budeme hovoriť *tablo* a znamená toto: hľadáme parametre x, y, z tak, aby hodnota f bola maximálna a pritom parametre s_1, \dots, s_4 boli nezáporné. V našom prípade vidno, že pri voľbe $x = y = z = 0$ bude nezápornosť s_1, \dots, s_4 splnená. Tablo (5) preto bude reprezentovať bázové riešenie $(0, 0, 0, 9, 24, 12, 4)$ pre bázu $\{s_1, s_2, s_3, s_4\}$ s hodnotou funkcie $f = 0$. Bázové premenné sú v riadkoch a nebázové nulové premenné sú parametre v stĺpcoch.

Obrázok vpravo graficky reprezentuje bázové riešenie z tabla (5). Čitateľa môže miast, že na obrázku je trojrozmerné telo, hoci nás program má 7 rozmerov. Problém je v tom, že obrázky zo sedemrozmerného priestoru sa do textu veľmi zle vkladajú. Vizualizáciu preto budeme robiť v pôvodnom trojrozmernom priestore z príkladu (4). Konkrétnie, ak riešeniu $(x, y, z, s_1, s_2, s_3, s_4)$ priradíme bod (x, y, z) , hodnoty (s_1, s_2, s_3, s_4) sa dajú dopočítať ako vzdialenosť od príslušných stien (pretože matica A má hodnosť 4, máme vždy 3 volné parametre a vrcholy mnogohostena vpravo zodpovedajú bázovým riešeniam).

Naším cieľom je nájsť najlepšie prípustné bázové riešenie, resp. jeho bázu. Chceme teda nájsť nejaké tri nebázové premenné (parametre), ktoré sa nastavia na 0 a z vyjadrenia ostatných premenných (a účelovej funkcie) pomocou týchto parametrov vieme určiť ich hodnoty. Vyskúšanie všetkých trojíc parametrov by preto zodpovedalo prehľadaniu všetkých bázových riešení (vrcholov mnogohostenu). Tomuto sa ale snažíme vyhnúť.

Ako môžeme lokálne zväčšiť hodnotu funkcie f ? Vidíme, že napr. premenná z je v prvom riadku s kladným koeficientom, takže ak zväčšíme hodnotu z , vzrástie aj f . Ako veľa môžeme z zväčšiť? Všetky premenné s_1, \dots, s_4 musia zostať nezáporné, takže každý riadok nám dáva limit na maximálnu hodnotu z v poradí $\frac{9}{2}, \frac{24}{5}, 12, 4$. Môžeme teda nastaviť $z = 4$, čím dostaneme, že $s_4 = 0$. Naše riešenie sa teda zmenilo tak, že nebázové premenné (parametre) budú x, y, s_4 namiesto x, y, z . Prispôsobíme tomu aj nás zápis tak, že z rovnice pre s_4 vyjadrieme z a dosadíme do ostatných rovníc. Dostaneme tak zápis

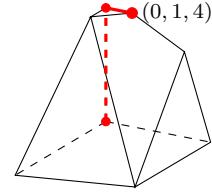
$$\begin{array}{rccccc} f = & 4 & + x & + y & - s_4 & \\ \hline z = & 4 & & & - s_4 & \\ s_1 = & 1 & -x & -y & + 2s_4 & \\ s_2 = & 4 & -4x & -y & + 5s_4 & \\ s_3 = & 8 & & -3y & + s_4 & \end{array} \quad (6)$$



Tablo (6) zodpovedá bázovému riešeniu $(0, 0, 4, 1, 4, 8, 0)$ pre bázu $\{z, s_1, s_2, s_3\}$ s hodnotou funkcie

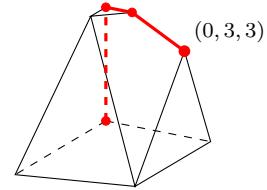
$f = 4$. Opäť máme zápis, kde v riadkoch sú bázové premenné a parametre stĺpcov sú nebázové premenné. Cieľom je nájsť parametre x, y, s_4 tak aby sa maximalizovala hodnota f a zároveň s_1, s_2, s_3, z ostali nezáporné. Je klúčové si uvedomiť, že sme urobili iba ekvivalentnú úpravu systému lineárnych rovníc a teda riešenia (5) a (6) sú rovnaké. Krok z (5) do (6) zodpovedá prejdeniu po jednej hrane mnohostena riešení. Budeme ho volať *pivotný krok*: jedna nebázová premenná, *pivot*, v našom prípade z , sa presunula do bázy a jedna bázová premenná (v našom prípade s_4) sa stala nebázovou. Tento postup môžeme opakovat. Vidíme napr., že y je v prvom riadku s kladným koeficientom a teda jeho zväčšením zväčšíme f . Jednotlivé riadky nám dávajú obmedzenia na maximálnu hodnotu y v poradí $1, 4, \frac{8}{3}$; v poslednej rovnosti y nevystupuje a preto naň nekladie ani žiadne obmedzenia. Zvolíme $y = 1$ a urobíme pivotný krok s pivotom y , pri ktorom y vystrieda v báze s_1 . Vyjadríme $y = 1 - x - s_1 + 2s_4$ a po dosadení dostaneme

$$\begin{array}{rcccc} f & = & 5 & - s_1 & + s_4 \\ \hline y & = & 1 & - x & - s_1 + 2s_4 \\ z & = & 4 & & - s_4 \\ s_2 & = & 3 & - 3x & + s_1 + 3s_4 \\ s_3 & = & 5 & + 3x & + 3s_1 - 5s_4 \end{array} \quad (7)$$



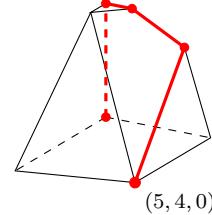
Máme tablo pre bázové riešenie $(0, 1, 4, 0, 3, 5, 0)$ pre bázu $\{y, z, s_2, s_3\}$ s hodnotou funkcie $f = 5$. Pokračujeme ďalej; jediná možnosť, ako zvýšiť f je zvoliť pivota s_4 a nahradíť ním v báze s_3 . Dostaneme

$$\begin{array}{rcccc} f & = & 6 & + \frac{3}{5}x & - \frac{2}{5}s_1 - \frac{1}{5}s_3 \\ \hline y & = & 3 & + \frac{1}{5}x & + \frac{1}{5}s_1 - \frac{2}{5}s_3 \\ z & = & 3 & - \frac{3}{5}x & - \frac{3}{5}s_1 + \frac{1}{5}s_3 \\ s_2 & = & 6 & - \frac{6}{5}x & + \frac{14}{5}s_1 - \frac{3}{5}s_3 \\ s_4 & = & 1 & + \frac{3}{5}x & + \frac{3}{5}s_1 - \frac{1}{5}s_3 \end{array} \quad (8)$$



Opäť jediná možnosť, ako spraviť pivotný krok, je zobrať do bázy x . Pri nastavení $x = 5$ sa ale vynulujú z aj s_2 a môžeme si vybrať, ktoré z nich v báze ponecháme. Nech x vystrieda v báze z , dostaneme

$$\begin{array}{rcccc} f & = & 9 & - z & - s_1 \\ \hline x & = & 5 & - \frac{5}{3}z & - s_1 + \frac{1}{3}s_3 \\ y & = & 4 & - \frac{1}{3}z & - \frac{1}{3}s_3 \\ s_2 & = & 2z & + 4s_1 & - s_3 \\ s_4 & = & 4 & - z & \end{array} \quad (9)$$



Dostali sme sa do situácie, keď nie je možné urobiť žiadny pivotný krok. Vieme ale, že sme robili iba ekvivalentné úpravy, a teda riešenia programov (4) a (9) sú rovnaké (programy majú rovnakú množinu prípustných riešení a rovnaké hodnoty účelovej funkcie). Lenže z (9) jasne vidno, že pre ľubovoľné nezáporné z a s_1 , hodnota f je vždy nanajvýš 9, takže nájdené riešenie je optimálne.

Tento príklad môžeme zovšeobecniť. Formálne si tablo prislúchajúce prípustnému bázovému riešeniu môžeme zadefinovať takto:

Definícia 2.4. Majme program v normálnom tvare

$$\max_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0 \}$$

kde $A \in \mathbb{R}^{m \times n}$, a jeho prípustné bázové riešenie prislúchajúce báze B . **Tablo** $\mathcal{T}(B)$ prislúchajúce báze B je systém $m+1$ lineárnych rovníc v premenných x_1, \dots, x_n, f , ktorý má rovnakú množinu riešení ako systém $A\mathbf{x} = \mathbf{b}, f = \mathbf{c}^\top \mathbf{x}$ a v maticovom zápise vyzerá

$$\begin{array}{rcl} f & = & f_0 + \mathbf{r}^\top \mathbf{x}_N \\ \mathbf{x}_B & = & \mathbf{p} + Q \mathbf{x}_N \end{array}$$

kde \mathbf{x}_B je vektor bázových premenných, $N = \{1, \dots, n\} - B$, \mathbf{x}_N je vektor nebázových premenných, $\mathbf{r} \in \mathbb{R}^{n-m}$, $\mathbf{p} \in \mathbb{R}^m$ a $Q \in \mathbb{R}^{m \times n-m}$.

Kedže sme v definícii tabla vychádzali z prípustného bázového riešenia B , zrejme $\mathbf{p} \geq \mathbf{0}$. Lahko sa presvedčíme, že takáto definícia tabla je korektná. Stačí si uvedomiť, že $\mathbf{b} = A\mathbf{x} = A_B \mathbf{x}_B + A_N \mathbf{x}_N$ a A_B je regulárna, takže existuje inverzná matica A_B^{-1} . Preto platí $\mathbf{x}_B = A_B^{-1} \mathbf{b} - A_B^{-1} A_N \mathbf{x}_N$ a dostávame nasledujúcu lemu, ktorej podrobny dôkaz prenechávame na čitateľa:

Lema 2.5. Každému prípustnému bázovému riešeniu B programu z definície 2.4 prislúcha práve jedno tablo $\mathcal{T}(B)$ a platí

$$\mathbf{p} = A_B^{-1} \mathbf{b} \quad Q = -A_B^{-1} A_N \quad f_0 = \mathbf{c}_B^\top A_B^{-1} \mathbf{b} \quad \mathbf{r} = \mathbf{c}_N - (\mathbf{c}_B^\top A_B^{-1} A_N)^\top.$$

Z rovnakých úvah, ako sme robili v úvodnom príklade vyplýva

Tvrdenie 2.6. Nech B je báza prípustného riešenia a nech v $\mathcal{T}(B)$ je $\mathbf{r} \leq \mathbf{0}$. Potom f_0 je maximálna hodnota daného programu.

Na to, aby sme dokončili opis simplexového algoritmu, potrebujeme definovať pivotný krok: vybrať nejakú nebázovú premennú, ktorá je v $\mathbf{r}^\top \mathbf{x}_N$ s kladným koeficientom, zvýšiť ju kolko sa dá tak, aby bázové premenné ostali nezáporné a zmeniť bázu. Označme si $B = \{\beta_1, \dots, \beta_m\}$ tak, že $\beta_1 < \beta_2 < \dots < \beta_m$ a podobne $N = \{\mu_1, \dots, \mu_{n-m}\}$, kde $\mu_1 < \mu_2 < \dots < \mu_{n-m}$. V tomto označení môžeme tablo rozpísat ako

$$\begin{array}{rcl} f & = & f_0 + \sum_{j=1}^{n-m} r_j x_{\mu_j} \\ \hline x_{\beta_1} & = & p_1 + \sum_{j=1}^{n-m} q_{1,j} x_{\mu_j} \\ \vdots & & \vdots \\ x_{\beta_m} & = & p_m + \sum_{j=1}^{n-m} q_{m,j} x_{\mu_j} \end{array} \quad (10)$$

Za pivota môžeme zobrať hocijakú premennú x_{μ_j} takú, že $r_j > 0$. Ak je $q_{i,j} > 0$, i -ty riadok nekladie žiadne obmedzenia, inak musí platiť $p_i + q_{i,j} x_{\mu_j} > 0$. Dostávame sa tak k nasledovnej definícii:

Definícia 2.7. Majme daný lineárny program v normálnom tvare a bázu B prislúchajúcu prípustnému riešeniu. Nech $\mathcal{T}(B)$ je zapísané ako v (10) a nech $r_e > 0$ pre nejaké e . Označme

$$s := \min_{i=1, \dots, m} \left\{ -\frac{p_i}{q_{i,e}} \mid q_{i,e} < 0 \right\}.$$

Pivotný krok podľa premennej x_{μ_e} zmení bázu B na bázu

$$B' := (B - \{\beta_\ell\}) \cup \{\mu_e\},$$

kde β_ℓ je ľubovoľný index, pre ktorý platí $q_{\ell,e} < 0$ a $-\frac{p_\ell}{q_{\ell,e}} = s$.

Čitateľ sa ľahko presvedčí, že B' je opäť báza prípustného riešenia. Simplexový algoritmus začína z nejakého prípustného riešenia s bázou B_0 a aplikuje pivotné kroky, kym sa dá. Podľa tvrdenia 2.6, ak algoritmus nájde bázu B , pre ktorú je $\mathbf{r} \leq \mathbf{0}$, tak našiel optimálne riešenie. Aby sme ukázali korektnosť simplexovej metódy, potrebujeme vyriešiť tri problémy: jednako ukázať, ako nájsť B_0 , dvak rozhodnúť, čo robiť, keď sa nedá urobiť žiadnen pivotný krok a napokon ukázať, že algoritmus v konečnom čase skončí.

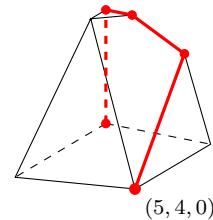
Čo sa stane, keď sa nedá vybrať pivot

Definícia pivotného kroku (Definícia 2.7) vyžaduje, aby pre pivota x_{μ_e} platilo $r_e > 0$. Ak neexistuje x_{μ_e} , pre ktoré $r_e > 0$, podľa Tvrdenia 2.6 máme optimálne riešenie. Ďalej musí platiť, že pivot nahradí v báze premennú x_{β_ℓ} , pre ktorú $q_{\ell,e} < 0$. Ak také ℓ neexistuje, t.j. ak $q_{\ell,e} \geq 0$ pre všetky ℓ , znamená to, že žiadnen riadok tabu nekladie limit na zväčšovanie premennej x_{μ_e} . S rastúcim x_{μ_e} rastie aj hodnota f , a preto daný program nemá konečné maximum.

Ako sa nezacykliť

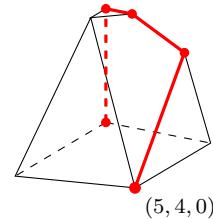
V úvodnom príklade sme v každom pivotnom kroku zväčšili hodnotu f . Ak by sme to vedeli zaručiť vždy, ľahko vidno, že algoritmus v konečnom čase skončí: je totiž iba konečne veľa bázových riešení. Čo by sa ale stalo, keby sme sa v kroku (8) rozhodli, že x nevystrieda v báze z ale s_2 ? Namiesto tabu (9) dostaneme tabu

$$\begin{array}{l} f = 9 + s_1 - \frac{1}{2}s_2 - \frac{1}{2}s_3 \\ \hline x = 5 + \frac{7}{3}s_1 - \frac{5}{6}s_2 - \frac{1}{2}s_3 \\ y = 4 + \frac{2}{3}s_1 - \frac{1}{6}s_2 - \frac{1}{2}s_3 \\ z = -2s_1 + \frac{1}{2}s_2 + \frac{1}{2}s_3 \\ s_4 = 4 + 2s_1 - \frac{1}{2}s_2 - \frac{1}{2}s_3 \end{array} \quad (11)$$



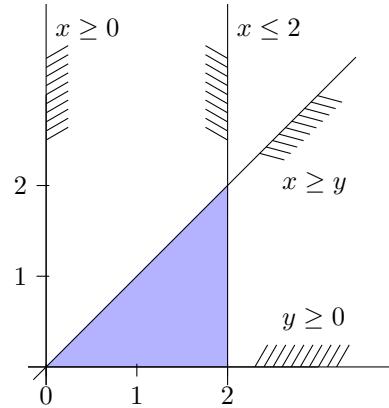
Tabu (9) reprezentuje bázu $\{x, y, s_2, s_4\}$ a tabu (11) bázu $\{x, y, z, s_4\}$, pričom obidvom bázam prislúcha rovnaké riešenie $(5, 4, 0, 0, 0, 0, 4)$. Zo zápisu (11) ale nevidno, že sme už našli optimum, a preto treba urobiť ešte jeden pivotný krok s pivotom s_1 . Pri tomto kroku ale zistíme, že premenná z nedovolí zväčšiť s_1 , a tak sme nútene urobiť krok "naprázdno" a vymeniť v báze s_1 za z . Dostaneme tabu

$$\begin{array}{l} f = 9 - \frac{1}{2}z - \frac{1}{4}s_2 - \frac{1}{4}s_3 \\ \hline x = 5 - \frac{7}{6}z - \frac{1}{4}s_2 + \frac{1}{12}s_3 \\ y = 4 - \frac{1}{3}z - \frac{1}{3}s_3 \\ s_1 = -\frac{1}{2}z + \frac{1}{4}s_2 + \frac{1}{4}s_3 \\ s_4 = 4 - z \end{array} \quad (12)$$



z ktorého už vidno optimalitu. Tento degenerovaný krok ale nezvýšil hodnotu f , čím rozobil náš pôvodný argument o zastavení: nevieme totiž zaručiť, že hodnota f sa v každom kroku zväčší. Nutnosť spraviť degenerovaný pivotný krok nemusí nastať iba na konci, keď už máme optimálne riešenie, ako ukazuje jednoduchý príklad (podľa [11]):

$$\begin{array}{ll} \text{maximalizovať} & y =: f(x, y) \\ \text{pri obmedzeniach} & -x + y \leq 0 \\ & x \leq 2 \\ & x, y \geq 0 \end{array}$$



Kedž zavedieme rezervné premenné s_1, s_2 , dostaneme tablo

$$\begin{array}{rcl} f = & & y \\ \hline s_1 = & x & -y \\ s_2 = & 2 & -x \end{array}$$

pre bázu $\{s_1, s_2\}$ s hodnotou riešenia $f = 0$. Jediná možnosť, ako pokračovať (a dostať sa k optimálnemu riešeniu s hodnotou 2), je spraviť degenerovaný pivotný krok, v ktorom y vystrieda v báze s_1 :

$$\begin{array}{rcl} f = & x & -s_1 \\ \hline y = & x & -s_1 \\ s_2 = & 2 & -x \end{array}$$

Podobná situácia je sa vyskytuje pomerne často.

Definícia 2.8. *Degenerovaný krok simplexového algoritmu je taký pivotný krok, pri ktorom sa báza B transformuje na bázu B' s rovnakým bázovým riešením.*

Degenerovaným krokom sa nevieme vyhnúť a navyše nasledovný príklad z [4] ukazuje, že ak nie sme dosť opatrní, môžme sa zacykliť. Uvažujme nasledovné tablo:

$$\begin{array}{rcl} f = & 10x_1 & -57x_2 & -9x_3 & -24x_4 \\ \hline x_5 = & -0.5x_1 & +5.5x_2 & +2.5x_3 & -9x_4 \\ x_6 = & -0.5x_1 & +1.5x_2 & +0.5x_3 & -x_4 \\ x_7 = & 1 & -x_1 \end{array} \tag{13}$$

pre bázu $\{x_5, x_6, x_7\}$. Predpokladajme, že konkrétny simplexový algoritmus vždy vyberie ako pivota nebázovú premennú x_{μ_e} s maximálnou hodnotou r_e . V prípade, že pivotný krok vynuluje viacero bázových premenných, vyberie sa premenná s minimálnym indexom. Nechávame ako cvičenie pre čitateľa overiť, že algoritmus v nasledujúcich iteráciách prejde cez bázy $\{x_1, x_6, x_7\}$, $\{x_1, x_2, x_7\}$, $\{x_2, x_3, x_7\}$, $\{x_3, x_4, x_7\}$, $\{x_4, x_5, x_7\}$ a napokon sa dostane naspäť do $\{x_5, x_6, x_7\}$. Kedže vznikol cyklus z degenerovaných pivotných krovov, algoritmus sa zacykľí.

Vidíme, že nemôžeme dúfať, že dokážeme termináciu simplexovej metódy pre ľubovoľnú voľbu pivota, ale musíme zafixovať nejaký konkrétny algoritmus pre pivotný krok. Existuje veľa alternatívnych pravidiel na výber pivota, s rôznymi prístupmi k problému zacyklenia. My na dôkaz terminácie použijeme *pravidlo najmenšieho indexu* pôvodne z [3]

Definícia 2.9. *Pravidlo najmenšieho indexu* vyberie za pivota premennú x_{μ_e} , kde μ_e je najmenšie také, že $r_e > 0$. Ak pivotný krok vynuluje viacero bázových premenných, z bázy sa vyhodí premenná x_{β_ℓ} s najmenším indexom β_ℓ .

Veta 2.10. Simplexový algoritmus, ktorý používa pravidlo najmenšieho indexu, vždy skončí a nájde optimálne riešenie.

Dôkaz: Vieme, že ak algoritmus skončí, nájde optimálne riešenie. Najprv si uvedomíme, že jediný spôsob, ako algoritmus môže neskončiť je, že sa dostane do cyklu, ktorý pozostáva zo samých degenerovaných krokov. Vskutku, ak algoritmus neskončí, musí nekonečne veľakrát spracovať nejakú bázu B . Kedže k B prislúcha práve jedno (prípustné) bázové riešenie, vždy, keď algoritmus spracováva B , je hodnota f rovnaká. Každý nedegenerovaný krok ale hodnotu f zväčší a degenerované kroky ju nemenia. Preto musí všetky kroky k ďalšiemu výskytu B musia byť degenerované. Na to, aby sme dokázali tvrdenie vety, nám teda stačí ukázať, že pri použití pravidla najmenšieho indexu nemôže nastať cyklus zo samých degenerovaných krokov.

Budeme postupovať sporom. Predpokladajme, že algoritmus má tablo s bázou B_0 a postupne vytvára tablá pre bázy $B_1, \dots, B_k = B_0$, pričom všetky pivotné kroky sú degenerované (t.j. všetky bázy B_1, \dots, B_k majú rovnaké bázové riešenie). Premennú x_i nazveme *nestála*, ak sa vyskytuje v niektornej báze B_j , ale nevyskytuje v inej $B_{j'}$. Nech t je najväčšie také, že x_t je nestála. Kedže x_t je nestála, existuje pivotný krok, v ktorom x_t vypadne z bázy, t.j. pre nejaké j je $x_t \in B_j$ a $x_t \notin B_{j+1}$. Takže musí existovať nejaká iná (nestála) premenná x_s , ktorá x_t nahradí v báze: $x_s \notin B_j$ a $x_s \in B_{j+1}$. Zároveň, ak skúmame postupnosť báz $B_j, \dots, B_k, B_1, B_2, \dots, B_{j+1}$, tak x_t sa zasa musí nejak do bázy vrátiť, t.j. musí existovať báza B_{j^*} , že $x_t \notin B_{j^*}$ a $x_t \in B_{j^*+1}$. Nech $\mathcal{T}(B_j)$ vyzerá takto:

$$\begin{array}{rcl} f & = & f_0 + \sum_{k \notin B_j} r_k x_k \\ \hline x_{\beta_1} & = & p_{\beta_1} + \sum_{k \in B_j} q_{\beta_1, k} x_k \\ \vdots & & \vdots \\ x_{\beta_m} & = & p_{\beta_m} + \sum_{k \in B_j} q_{\beta_m, k} x_k \end{array} \quad (14)$$

kde $B_j = \{\beta_1, \dots, \beta_m\}$. Kedže predpokladáme, že všetky kroky cyklu sú degenerované, bázy B_j a B_{j^*} majú rovnaké bázové riešenie (t.j. hodnoty všetkých premenných aj f sú rovnaké). Preto môžeme napísat

$$f = f_0 + \sum_{k=1}^n r_k^* x_k \quad (15)$$

kde

$$r_k^* = \begin{cases} 0 & \text{ak } k \in B_{j^*} \\ \text{koeficient } r \text{ pri } x_k \text{ v table } \mathcal{T}(B_{j^*}) & \text{inak} \end{cases}$$

Pretože $\mathcal{T}(B_{j^*})$, a špeciálne rovnicu (15), sme dostali ekvivalentnými úpravami systému (14), všetky riešenia systému (14) spĺňajú (15). Vyrobme si teraz nejaké (nie bázové, ani prípustné) riešenie systému (14): zvolme ľubovoľné y a položme

$$x_i = \begin{cases} y & \text{ak } i = s \\ 0 & \text{ak } i \neq s \text{ a } i \notin B_j \\ p_i + q_{i,s}y & \text{ak } i \in B_j \end{cases}$$

Lahko vidno, že takto zvolené \mathbf{x} je riešením systému (14)⁵. Kedže naše \mathbf{x} spĺňa (14) aj (15), z vyjadrenia f v obidvoch dostaneme

$$f_0 + r_s y = f_0 + \sum_{k=1}^n r_k^* x_k = f_0 + r_s^* y + \sum_{k \in B_j} r_k^* (p_k + q_{k,s}y)$$

⁵Je to ako keby sme robili pivotný krok s premennou x_s o y , pričom sa nestaráme o to, aby premenné ostali nezáporné.

a po úprave

$$\left(r_s - r_s^* - \sum_{k \in B_j} r_k^* q_{k,s} \right) y = \sum_{k \in B_j} r_k^* b_k.$$

Tento vzťah platí pre každé y , a nakoľko pravá strana od y nezávisí, dostávame, že

$$r_s - r_s^* - \sum_{k \in B_j} r_k^* q_{k,s} = 0.$$

Pretože premenná x_s bola v báze B_j vybratá ako pivot, musí byť $r_s > 0$. V B_{j^*} bola ako pivot vybratá premenná x_t , a keďže $t > s$, musí byť $r_s^* \leq 0$. Kedže $r_s - r_s^* > 0$, musí existovať nejaké $z \in B_j$, pre ktoré

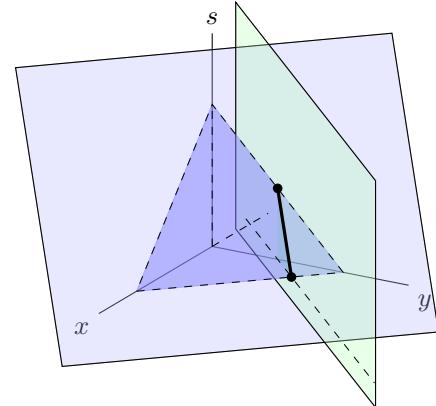
$$r_z^* q_{z,s} > 0.$$

Premenná x_z je bázová v B_j , ale zároveň $r_z^* \neq 0$, preto z definície r^* vyplýva, že $z \notin B_{j^*}$; x_z je teda nestála premenná a z definície t platí $z \leq t$. Zároveň $z \neq t$: pretože x_t bolo vyhodené z bázy B_j pri pivotnom kroku, musí byť $q_{t,s} < 0$ a aj $r_t^* q_{t,s} < 0$ (lebo x_t bol pivot pri B_{j^*}). Teraz vieme, že $z < t$. Lenže x_z neboli v B_{j^*} pivot, preto musí byť $r_z^* \leq 0$. Kedže $r_z^* q_z > 0$, musí byť $q_{z,s} < 0$. Kedže všetky bázové riešenia v degenerovanom cykle sú rovnaké a $z \notin B_{j^*}$, je $x_z = 0$ v bázovom riešení B_j aj B_{j^*} . Pretože $z \in B_j$, musí byť $p_z = 0$. To ale znamená, že x_z sa dalo vyhodiť z bázy B_j , ale namiesto neho sa vyhodilo x_t , čo je v spore s pravidlom minimálneho indexu. \square

Ako začať

Posledný detail, ktorý potrebujeme vyriešiť, je otázka, ako simplexový algoritmus naštartovať. Doteraz sme totiž predpokladali, že začíname z bázy B_0 , ktorá má prípustné bázové riešenie. V úvodnom príklade sme za štartovaciu bázu B_0 v (5) zvolili rezervné premenné s_1, \dots, s_4 . Tento prístup zjavne funguje pre programy tvaru $\max_{\mathbf{x} \in \mathbb{R}^7} \{\mathbf{c}^\top \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0\}$ s pridanými rezervnými premennými, ak $\mathbf{b} \geq \mathbf{0}$. Čo ale s inými programami? Uvažujme nasledovný program:

$$\begin{array}{ll} \text{maximalizovať} & 4x - z =: f(x, y, z) \\ \text{pri obmedzeniach} & \begin{aligned} x + y + z &= 4 \\ x - y &= -2 \\ x, y, z &\geq 0 \end{aligned} \end{array} \quad (16)$$



Prípustné riešenia tvoria úsečku $(1, 3, 0) - (0, 2, 2)$. Na to, aby sme mohli spustiť simplexový algoritmus, potrebujeme nájsť nejaké prípustné riešenie. Pre bod (x, y, z) si označme $p_1 := 4 - x - y - z$; p_1 nám hovorí, ako veľmi je porušená prvá rovnosť⁶. Podobne nech $p_2 := 2 + x - y$ (všimnite si, že sme rovnici upravili tak, aby absolútny člen bol nezáporný). Nájsť prípustné riešenie znamená nájsť taký bod (x, y, z) , že $p_1 = p_2 = 0$, a teda $p_1, p_2 \geq 0$ a $p_1 + p_2 = 0$. Lahko vidno, že program (16) má prípustné riešenie práve vtedy, ak program

$$\begin{array}{ll} \text{maximalizovať} & -p_1 - p_2 \\ \text{pri obmedzeniach} & \begin{aligned} p_1 + x + y + z &= 4 \\ p_2 - x + y &= 2 \\ x, y, z, p_1, p_2 &\geq 0 \end{aligned} \end{array} \quad (17)$$

⁶nie je to vzdialenosť bodu (x, y, z) od roviny $x + y + z = 4$

má riešenie s hodnotou 0. V tomto programe ľahko vidno, že $\{p_1, p_2\}$ je báza prípustného riešenia. Môžeme teda použiť simplexový algoritmus na nájdenie optimálneho riešenia a toto použiť ako počiatočné prípustné riešenie pôvodného programu.

Tento postup môžme uplatniť vždy. Majme lineárny program v normálnom tvare

$$\max_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \}$$

Najprv zabezpečíme, aby $\mathbf{b} \geq 0$: ak $b_i < 0$ pre nejaké i , tak príslušnú rovnicu prenásobíme -1 . Zavedieme nové premenné x_{n+1}, \dots, x_{n+m} a zostavíme pomocný program

$$\max_{\tilde{\mathbf{x}} \in \mathbb{R}^{n+m}} \left\{ -x_{n+1} - \dots - x_{n+m} \mid \tilde{A}\tilde{\mathbf{x}} = \mathbf{b}, \tilde{\mathbf{x}} \geq \mathbf{0} \right\}$$

kde $\tilde{A} = (A \mid I_m)$ dostaneme z A pripojením identickej matice rozmerov $m \times m$. Pretože $\mathbf{b} \geq 0$, $\{x_{n+1}, \dots, x_{n+m}\}$ tvoria bázu prípustného riešenia a môžeme použiť simplexový algoritmus na získanie optimálneho riešenia. Ak je optimálne riešenie 0, máme prípustné riešenie pôvodného programu. Naopak, pre každé prípustné riešenie pôvodného programu existuje riešenie pomocného programu s hodnotou 0, takže ak je optimum pomocného programu záporné, vieme, že pôvodný program nemal žiadne prípustné riešenie.

Cvičenie. Naprogramujte simplexový algoritmus s pravidlom najmenšieho indexu.

2.3 Zložitosť simplexového algoritmu

V predchádzajúcej kapitole sme sa si priblížili simplexovú metódu, ktorá umožňuje riešiť úlohy lineárneho programovania efektívnejšie ako prehľadávaním všetkých vrcholov telesa prípustných riešení. Ukázali sme, že simplexová metóda s pravidlom najmenšieho indexu vždy skončí. Otázkou teraz je, či je naozaj efektívna. Odpoveď je prekvapivá. Napriek tomu, že v praxi sa simplexový algoritmus ukazuje ako veľmi rýchly, jeho zložitosť v najhoršom prípade je exponenciálna, ako o chvíľu ukážeme.

Najprv je však namieste zopakovať niekoľko základných faktov, keďže v tomto prípade záleží na subtitlných detailoch. Keď analyzujeme zložitosť nejakého algoritmu, robíme tak v závislosti od parametra, ktorý je, v princípe, súčasťou definície problému. Keď napríklad povieme, že nejaký algoritmus má v najhoršom prípade zložitosť $O(n^2)$, myslíme tým, že existuje konštantă c a n_0 taká, že pre ľubovoľný vstup, ktorého parameter $n > n_0$, je čas algoritmu $\leq cn^2$. Prirodzeným parametrom, ktorý sa dá použiť univerzálne, je dĺžka vstupu: súčasťou definície problému je vždy aj spôsob kódovania stupu do refazca a počet bitov, potrebných na zápis daného vstupného refazca je dobrý zložitostný parameter. Niekoľko (a vlastne dosť často) sa ale používajú iné parametre, ktoré sú pre daný problém prirodzenejšie: keď sa napríklad analyzuje zložitosť triedenia postupnosti prirodzených čísel, je zväčša parametrom počet triedených čísel n , aj keď dĺžka vstupu závisí od veľkosti triedených čísel. Podobným príkladom sú grafové algoritmy, ktoré sa niekedy analyzujú vzhľadom na počet vrcholov, aj keď na zápis n -vrcholových grafov je treba vo všeobecnosti až $\Omega(n^2)$ bitov⁷.

Vstupom lineárneho programu s n premennými a m obmedzeniami sú dva vektory \mathbf{c} a \mathbf{b} reálnych čísel a matica $A \in \mathbb{R}^{m \times n}$. Prirodzenými parametrami sú teda m , n a dĺžka vstupu, pričom v poslednom prípade treba brať do úvahy aj spôsob kódovania reálnych čísel a zmierňať sa s faktom, že ak chceme mať konečné vstupy, tak nemôžeme zapísat všetky reálne čísla.

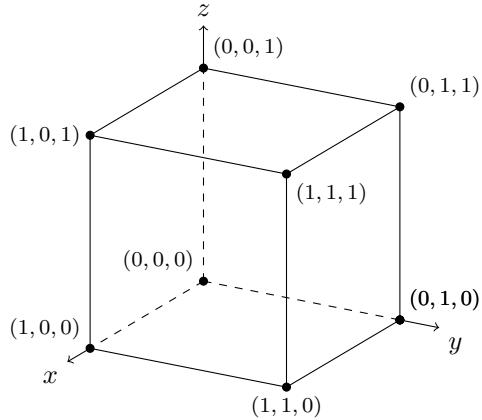
Tieto aspekty je dobré mať na pamäti, aj keď nás momentálne nemusia príliš trápiť: zostrojíme vstup s n premennými a $2n$ obmedzeniami, na ktorom je čas simplexového algoritmu $\Omega(2^n)$. Navyše pri tom použijeme čísla s krátkym zápisom (stačia nám čísla $\{\pm 1, \pm \frac{1}{4}, 0\}$), takže ukážeme, že algoritmus je exponenciálny od hociktorého zo spomenutých parametrov.

⁷Stačí si uvedomiť, že v očíslovanom grafe je $\binom{n}{2}$ potenciálnych hrán a každá v ňom môže byť alebo nebyť prítomná, t.j. je $2^{\binom{n}{2}}$ grafov a na identifikáciu každého z nich je z Dirichletovho princípu treba aspoň $\binom{n}{2}$ bitov.

Budeme uvažovať simplexový algoritmus, ktorý používa pravidlo najmenšieho indexu (pre veľa iných pravidiel existujú podobné kontrapríklady) a pre každé n skonštruujeme vstup s n premennými a $2n$ obmedzeniami tak, že teleso prípustných riešení má 2^n vrcholov a simplexový algoritmus ich všetky prehľadá. V nami konštruovanom zadaní bude cieľom maximalizovať x_n a obmedzenia budú tvoriť "pokrivenú" kocku. Začnime s tým, že pomocou $2n$ obmedzení vyrobíme n -rozmernú kocku:

$$\begin{aligned} 0 \leq x_1 &\leq 1 \\ 0 \leq x_2 &\leq 1 \\ &\dots \\ 0 \leq x_n &\leq 1 \end{aligned}$$

V troch rozmeroch teleso prípustných riešení je kocka:



Našim cieľom bude posunúť vrcholy kocky tak, aby vznikla dlhá rastúca "špirála". Zvoľme si nejaké $\varepsilon < \frac{1}{2}$ a definujme obmedzenia

$$\begin{aligned} \varepsilon \leq x_1 &\leq 1 \\ \varepsilon x_1 \leq x_2 &\leq 1 - \varepsilon x_1 \\ &\dots \\ \varepsilon x_{n-1} \leq x_n &\leq 1 - \varepsilon x_{n-1} \end{aligned}$$

Program prepíšeme do normálneho tvaru tak, že zavedieme rezervné premenné r_i, s_i a obmedzenia budú mať formu rovností. Dostávame program:

$$\begin{array}{ll} \text{maximalizovať} & x_n \\ \text{pri obmedzeniach} & \begin{array}{l} x_1 - r_1 = \varepsilon \\ x_1 + s_1 = 1 \\ x_2 - \varepsilon x_1 - r_2 = 0 \\ x_2 + \varepsilon x_1 + s_2 = 1 \\ \dots \dots \\ x_n - \varepsilon x_{n-1} - r_n = 0 \\ x_n + \varepsilon x_{n-1} + s_n = 1 \end{array} \end{array} \quad (18)$$

kde všetky premenné sú nezáporné. Ako vyzerajú bázové riešenia? Kvôli pridaným premenným sú jednotlivé obmedzenia nezávislé (každé obmedzenie obsahuje jednu premennú, ktorá sa nevyskytuje nikde inde), preto báza má $2n$ prvkov. Zároveň vidno, že $r_1 + s_1 = 1 - \varepsilon$ a pre každú dvojicu premenných r_i, s_i , kde $i > 1$, platí $r_i + s_i = 1 - 2\varepsilon x_{i-1} > 0$. Preto nemôže platiť $r_i = s_i = 0$, a teda každá báza musí obsahovať aspoň jednu z premenných r_i, s_i . Navyše všetky $x_i > 0$ a teda sú v každej báze. Každá báza B sa preto dá jednoznačne charakterizovať množinou $R_B \subseteq \{1, \dots, d\}$: premenné bázy B sú potom

$$\{x_1, \dots, x_n\} \cup \bigcup_{i \in R_B} \{r_i\} \cup \bigcup_{i \notin R_B} \{s_i\}$$

Zároveň je zrejmé nasledovné tvrdenie:

Tvrdenie 2.11. Každý pivotný krok je jednoznačne charakterizovaný indexom i , pričom vymení príslušnosť do bázy pre premenné r_i a s_i .

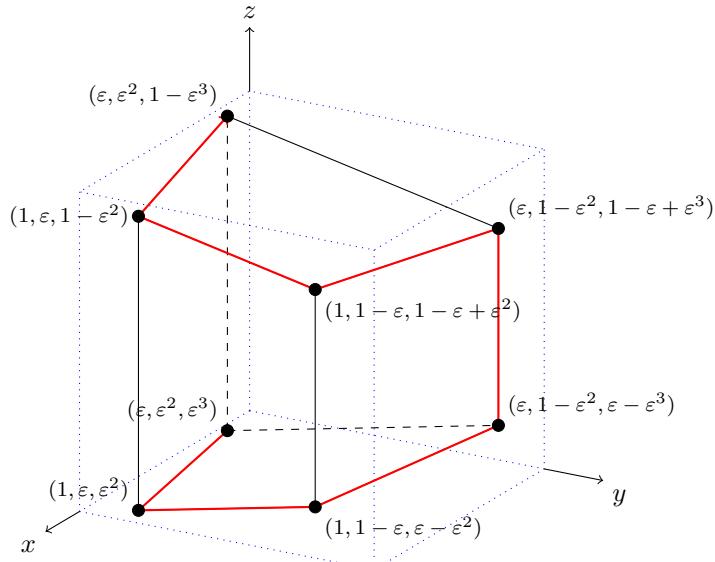
Pre ilustráciu, nech $n = 3$. V maticovom zápise máme program

$$\max\{x_3 \mid Ax = \mathbf{b}, \mathbf{x} \geq 0\}$$

kde

$$A = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -\varepsilon & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ \varepsilon & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\varepsilon & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & \varepsilon & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ r_1 \\ s_1 \\ r_2 \\ s_2 \\ r_3 \\ s_3 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} \varepsilon \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Matica A má hodnosť 6 a $R_B \subseteq \{1, 2, 3\}$. Máme teda 8 bázových riešení, ktoré tvoria pokrivenú kocku:



Červeným je vyznačená rastúca cesta, ktorá začína v riešení s množinou $R_{B_0} = \emptyset$ a prejde všetky vrcholy, pričom vždy sa posunie po prvej možnej dimenzii, v ktorej účelová funkcia rastie. Aby sme mohli tento príklad zovšeobecniť na n dimensií, potrebujeme vedieť argumentovať o pivotných krokoch algoritmu s pravidlom najmenšieho indexu. K tomu nám pomôže tvrdenie 2.11 a nasledovná lema:

Lema 2.12. Majme bázu B programu (18) a k nej tablo $\mathcal{T}(B)$. Nech je účelová funkcia v $\mathcal{T}(B)$ vyjadrená pomocou nebázových premenných ako $x_n = c_0 + c_1 v_1 + c_2 v_2 + \dots + c_n v_n$, kde v_i je r_i alebo s_i a c_i je koeficient. Potom c_i je kladný práve vtedy, ak počet bázových premenných r_j pre $j \geq i$ je párný, t.j.

$$|\{j \mid j \in R_B, j \geq i\}| \equiv 0 \pmod{2}$$

Dôkaz: Dôkaz urobíme indukciou na rozmer problému n . Pre $n = 1$, ak r_1 je v báze, máme $x_1 = 1 - s_1$ a c_1 je záporný, ak r_1 nie je v báze, máme $x_1 = \varepsilon + r_1$ a c_1 je kladný.

Nech tvrdenie platí pre $n - 1$. Ak $n \in R_B$, tak v zápisе x_n musí figurovať s_n a teda musí byť tvaru $x_n = 1 - s_n - \varepsilon(c'_0 + c'_1 v'_1 + \dots + c'_{n-1} v'_{n-1})$, kde $x_{n-1} = c'_0 + c'_1 v'_1 + \dots + c'_{n-1} v'_{n-1}$ je zápis x_{n-1} pomocou nebázových premenných v_1, \dots, v_{n-1} . Roznásobením a použitím indukčného predpokladu dostaneme výsledok. Ak $n \notin R_B$, postup je analogický s použitím vzťahu $x_n = r_n + \varepsilon x_{n-1}$. \square

Teraz môžeme ukázať, že simplexový algoritmus navštíví všetky vrcholy:

Veta 2.13. *Nech $i \in \{1, \dots, n\}$ a $R \subseteq \{i+1, \dots, n\}$. Ak simplexový algoritmus, ktorý používa pravidlo najmenšieho indexu, začína z bázy B_0 , kde $R_{B_0} = R$ (resp. $R_{B_0} = \{i\} \cup R$) a $|R|$ je párne (resp. $|R|$ je nepárne), tak prejde cez všetky bázy tvaru $R' \cup R$ kde $R' \subseteq \{1, \dots, i\}$ a skončí v báze B_1 , kde $R_{B_1} = \{i\} \cup R$ (resp. $R_{B_1} = R$).*

Dôkaz: Indukciou na i . Ak $i = 1$, potom v obidvoch prípadoch ($R_{B_0} = R$, $|R|$ je párne, aj $R_{B_0} = \{1\} \cup R$, $|R|$ je nepárne) je podľa lemy 2.12 koeficient pri v_1 kladný a algoritmus urobí pivotný krok s indexom 1.

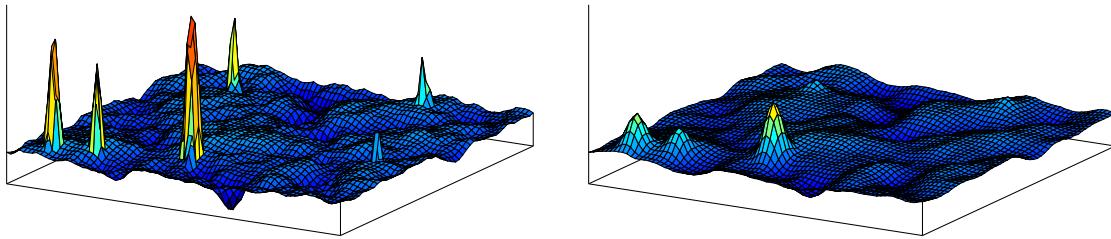
Nech teraz tvrdenie platí pre $i - 1$. Máme dva prípady. Nech najprv $R_{B_0} = R$ a $|R|$ je párne. Kedže $R \subseteq \{i, \dots, n\}$, môžme použiť indukčný predpoklad: algoritmus prejde všetky bázy tvaru $R' \cup R$, kde $R' \subseteq \{1, \dots, i-1\}$ a skončí v $\{i-1\} \cup R$. Pretože $|R|$ je párne, podľa lemy 2.12 algoritmus prejde to bázy $\{i-1, i\} \cup R$. Použitím indukčného predpokladu pre $i - 1$ a nepárnu množinu $\{i\} \cup R$ dostávame výsledok.

Druhý prípad, keď $R_{B_0} = \{i\} \cup R$ a $|R|$ je nepárne je analogický a prenechávame ho na čitateľa. \square

Dôsledok 2.14. *Simplexový algoritmus s pravidlom najmenšieho indexu urobí exponenciálne veľa iterácií na programe (18).*

Vidíme teda, že simplexový algoritmus je v najhoršom prípade exponenciálny, nech už za parameter zoberieme počet premenných, počet obmedzení, alebo dĺžku vstupu. Ako si ale vysvetliť, že v praxi funguje ozaj dobre? Možným smerom by bolo analyzovať priemerný prípad. Hned ale narážame na problém, ako priemerný prípad definovať. Vskutku, existujú výsledky, ktoré hovoria, že simplexový algoritmus urobí v "priemernom" prípade polynomiálny počet krokov, kde "priemerný prípad" znamená očakávanú hodnotu, ak matica A aj vektory \mathbf{c} , \mathbf{b} sú vybrané náhodne z daného pravdepodobnostného rozdelenia. Toto ale stále nie je zdaleka uspokojivá odpoveď: priemerný prípad je totiž veľmi daleko od "typického", v praxi sa vyskytujeceho, prípadu; program, ktorého matica by bola náhodná by bol v skutočnosti veľmi podivná výnimka. Vysvetlenie priniesol pojem *vyhľadenej zložitosti*⁸, ktorý je kombináciou najhoršieho a priemerného prípadu: uvažujeme najhoršiu možnú inštanciu, ale pre každú inštanciu neuvažujeme iba čas potrebný na jej vyriešenie, ale priemerný čas potrebný na vyriešenie inštancií z jej blízkeho okolia. Okolie inštancie dostaneme tak, že každé číslo, vyskytujúce sa vo vstupe, posunieme o malú náhodnú hodnotu. Spielman a Teng [13] ukázali, že vyhľadená zložitosť simplexového algoritmu je pre každú inštanciu polynomiálna. Ak si intuitívne predstavíme priestor všetkých vstupov ako rovinu, zložitosť simplexového algoritmu je ako na obrázku vľavo: väčšinou je polynomiálna a má iba riedko rozmiestnené jednotlivé "zlé" inštancie.

⁸smoothed complexity



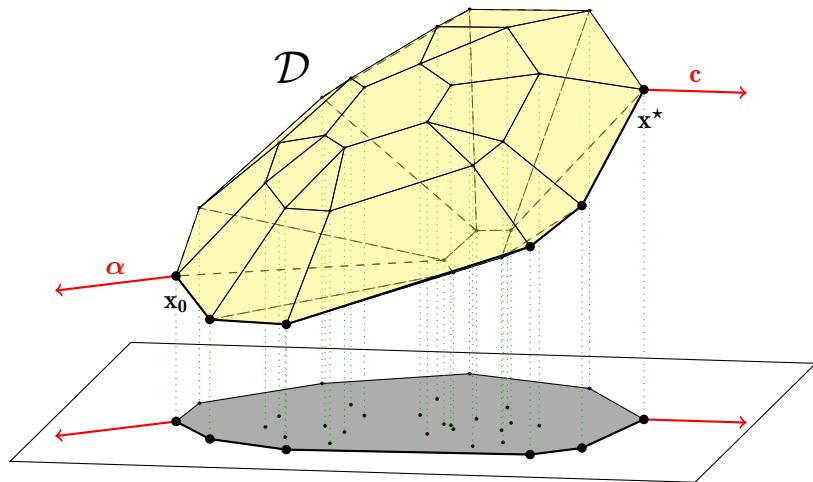
Po spriemerovaní cez malé okolie sa zlé inštancie vyhľadia ako na obrázku vpravo. Toto vysvetluje, prečo sa simplexová metóda dá považovať za efektívnu metódu riešenia lineárnych programov: existujú súčasťne exponenciálne zlé vstupy, ale na to, aby sme na taký natrafili, musia byť všetky vstupné čísla nastavené veľmi presne – stačí, ak vstupné hodnoty obsahujú malý náhodný šum a v očakávanom prípade je každá inštancia polynomiálna.

Výsledok Spielmana a Tenga sa považuje za prelomový a na prvý (aj na druhý a tretí) pohľad môže vyzerat úplne nepochopiteľne. Nie je v možnostiach tohto textu ukázať kompletný dôkaz, ktorý je pomerne náročný, ničmenej na záver tejto kapitoly by sme chceli ukázať aspoň jednoduchú vizuálnu predstavu, ktorá by zvedavému čitateľovi naznačila, že nejde o žiadnu mágiu.

V predchádzajúcim texte sme predstavili simplexový algoritmus s pravidlom najmenšieho indexu, ale pre teraz sa pre naše účely bude viac hodíť iné pravidlo, ktoré sa volá *pravidlo sledovania tieňa*. Majme lineárny program v tvare

$$\max_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0 \}$$

Prípustné riešenia tvoria mnohosten \mathcal{D} v n -rozmernom priestore a nájsť optimálne bázové riešenie znamená nájsť vrchol \mathcal{D} , ktorý je najďalej v smere vektora \mathbf{c} . Kedže vieme, že v dvoch rozmeroch je tento problém ľahký, môžeme urobiť nasledovnú úvahu: zoberme si štartovacie bázové riešenie \mathbf{x}_0 ; keďže je to vrchol \mathcal{D} , existuje vektor $\boldsymbol{\alpha}$, že vrchol \mathbf{x}_0 maximalizuje hodnotu $\boldsymbol{\alpha}^\top \mathbf{x}$ (\mathbf{x}_0 je najďalej v smere $\boldsymbol{\alpha}$). Zoberme si rovinu danú vektormi $\boldsymbol{\alpha}$ a \mathbf{c} a premietnime každý vrchol \mathcal{D} do nej – dostaneme množinu bodov v rovine a ich konexný obal bude *tieň*, ktorý \mathcal{D} vrhá do roviny.



Nie je ľahké vidieť, že \mathbf{x}_0 aj optimálne riešenie \mathbf{x}^* ležia na hranici tieňa. Simplexový algoritmus s pravidlom sledovania tieňa postupuje v princípe takto: keď sa v nejakom bázovom riešení treba

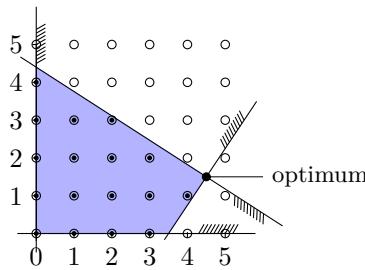
rozhodnúť, ako zmeniť bázu, vyberie sa bázové riešenie, ktoré leží na hranici tieňa (to sa dá otestovať napr. tak, že zostrojíme všetky susedné bázové riešenia a porovnáme, kde ležia ich priemety). Počet iterácií simplexového algoritmu s týmto pravidlom je zjavne nanajvýš počet bodov na hranici tieňa. Dôležitý medzikrok v dôkaze je, že sa riešený program prevedie do tvaru, kde obmedzenia majú tvar $\mathbf{a}_i^T \mathbf{x} \leq 1$; v tomto prípade sa dá ukázať, že počet vrcholov tieňa je nanajvýš počet vrcholov mnohouholníka \mathcal{M} , ktorý dostaneme ako prienik konvexného obalu bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$ a roviny definovej vektormi $\boldsymbol{\alpha}, \mathbf{c}$. Jadrom dôkazu je geometrické tvrdenie: ak máme n bodov v d -rozmernom priestore, každý z nich posunieme o náhodný vektor s normálnym rozdelením a disperziou σ^2 , tak v očakávanom prípade sú uhly medzi susednými úsečkami \mathcal{M} dosť ostré, a preto \mathcal{M} nemôže mať príliš veľa vrcholov, konkrétnie nanajvýš nejaký polynóm $\text{poly}(n, d, \frac{1}{\sigma})$. Od týchto letmých úval je k dôkazu ešte veľmi dlhá cesta; našim cieľom však bolo iba naznačiť smer, akým sa dôkaz tohto typu môže uberať. Záujemcov o detaily odkazujeme na články [Spielman,Teng] a prednášky <http://www.cs.yale.edu/homes/spielman/BAP/>

3

Zaokrúhľovanie lineárnych programov

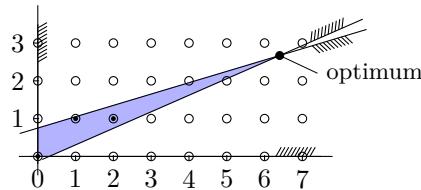
3.1 Celočíselné programy a relaxácia: dobrý, zlý a škaredý

V predchádzajúcich častiach sme ukázali, ako efektívne riešiť úlohu lineárneho programovania. Častokrát nás ale zaujímajú iba také riešenia, v ktorých sú hodnoty nájdených premenných celočíselné. V prvom motivačnom príklade nám vyšlo, že optimálne je kúpiť $2\frac{2}{3}$ dl zelenej vody. To ale väčšinou nie je možné, lebo nápoje sa zvyčajne predávajú v násobkoch nejakého fixného objemu.



Prípustné riešenia lineárneho programu a jeho celočíselnej reštrikcie.

Na prvý pohľad by sa možno zdalo, že ide iba o kozmetický problém: nájdeme optimálne riešenie spojitej verzie, vhodne ho zaokrúhlime a dostaneme, ak nie priamo optimum, tak určite niečo blízke optimu. Na druhý pohľad začne byť vidno, že to také jednoduché nebude: ak by nám lineárny program, ktorý hľadá optimálny spôsob cestovania odporučil kúpiť 0.5 letenky a 0.5 vlakového lístka, vhodné zaokrúhlenie je vlastne celé riešenie. Ako vidno z nasledujúceho obrázka, najbližšie celočíselné riešenie môže byť od optimálneho pomerne daleko a nie je zrejmé, ako by sme ho mali hľadať.



Pridaním dodatočných obmedzení, že niektoré z premenných lineárneho programu musia byť celočíselné, dostávame tzv. celočíselné lineárne programy:

Definícia 3.1. *Celočíselný lineárny program (ILP) v normálnom tvare je linárny program*

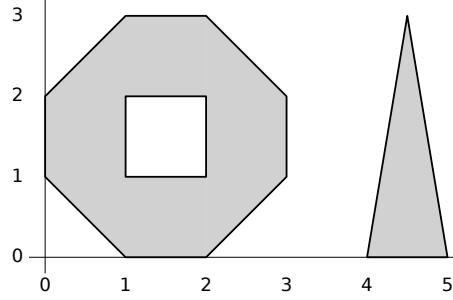
$$\max_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0 \}$$

s dodatočnými obmedzeniami tvaru $x_i \in \mathbb{Z}$.

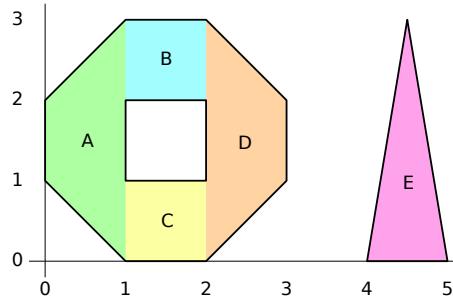
Skúseného čitateľa isto neprekvapí, že celočíselné obmedzenia podstatne zvyšujú vyjadrovaciu schopnosť lineárnych programov. Pri formulovaní zložitejších vzťahov pomocou ILP hrajú dôležitú úlohu tzv. *indikátorové premenné*: sada premenných $\delta_1, \dots, \delta_k \in \mathbb{Z}$ s obmedzeniami

$$\begin{aligned} \delta_1 + \delta_2 + \dots + \delta_k &= 1 \\ \delta_i &\geq 0 \text{ pre } i = 1 \dots k \end{aligned}$$

má tú vlastnosť, že v ľubovoľnom prípustnom riešení je práve jedna $\delta_i = 1$ a ostatné sú nulové. Indikátorové premenné umožňujú vyjadriť alternatívnu medzi viacerými možnosťami a tým napríklad popísat zložité nekonvexné obory hodnôt. Ako demonštráciu predpokladajme, že chceme maximalizovať funkciu $f(x, y) = x + y$ na takejto množine \mathcal{D} :



Množinu \mathcal{D} môžeme rozdeliť na 5 častí



z ktorých každú vieme reprezentovať lineárnymi obmedzeniami:

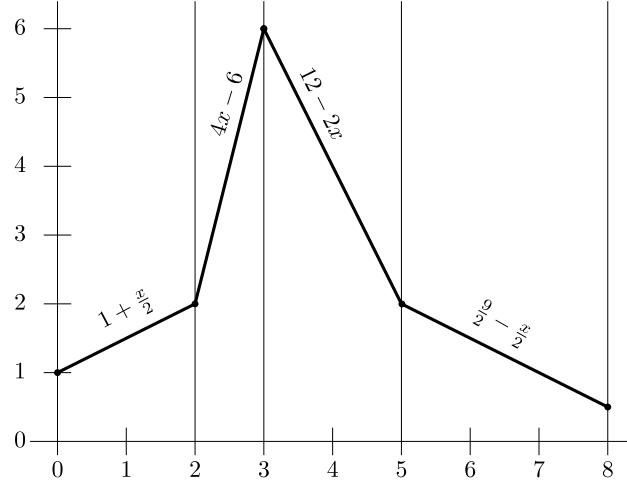
$$\begin{array}{lllll}
 x \geq 0 & x \geq 1 & x \geq 1 & x \geq 2 & x \geq 4 \\
 x \leq 1 & x \leq 2 & x \leq 2 & x \leq 3 & x \leq 5 \\
 y \geq 1 - x & y \geq 2 & y \geq 0 & y \geq x - 2 & y \leq 6x - 24 \\
 y \leq 2 + x & y \leq 3 & y \leq 1 & y \leq 5 - x & y \leq -6x + 30 \\
 & & & & y \geq 0
 \end{array}$$

Zavedieme indikátorové premenné $\delta_1, \dots, \delta_5 \in \mathbb{Z}$ a každú sadu obmedzení určujúcich jednu oblasť prepíšeme tak, aby v prípade, že príslušný indikátor je nulový, dávali triviálne obmedzenia a aby sa zachovali pôvodné obmedzenia, ak je indikátor jednotkový. Dostaneme tak ILP: maximalizovať $x + y$ pri obmedzeniach

$$\begin{array}{lllll}
 x \geq 0 & x \geq \delta_2 & x \geq \delta_3 & x \geq 2\delta_4 & x \geq 4\delta_5 \\
 x \leq 5 - 4\delta_1 & x \leq 5 - 3\delta_2 & x \leq 5 - 3\delta_3 & x \leq 5 - 2\delta_4 & x \leq 5 \\
 y \geq \delta_1 - x & y \geq 2\delta_2 & y \geq 0 & y \geq -5 + 3\delta_4 + x & y \leq 3 - 27\delta_5 + 6x \\
 y \leq 3 - \delta_1 + x & y \leq 3 & y \leq 3 - 4\delta_3 & y \leq 8 - 3\delta_4 - x & y \leq 33 - 3\delta_5 - 6x \\
 & & & & y \geq 0
 \end{array}$$

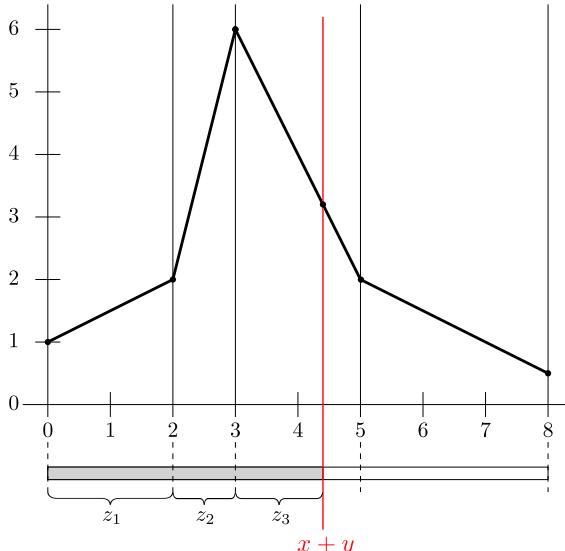
$$\begin{aligned}
 \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 &= 1 \\
 \delta_i &\geq 0, \delta_i \in \mathbb{Z} \text{ pre } i = 1 \dots 5
 \end{aligned}$$

Inou v praxi dôležitou vlastnosťou celočíselných lineárnych programov je schopnosť optimizovať nelineárne úcelové funkcie. Predpokladajme, že v predchádzajúcom príklade namiesto funkcie $f(x, y) = x + y$ chceme maximalizovať funkciu $f(x, y) = \varphi(x + y)$, kde funkcia φ je zložená z lineárnych úsekov:



Zavedieme štyri pomocné premenné z_1, z_2, z_3, z_4 , ktoré udávajú, aká časť príslušného intervalu je "zaplnená" hodnotou $x + y$:

$$x + y = z_1 + z_2 + z_3 + z_4$$



Potrebuje sa pridať ďalšie obmedzenia, ktoré zabezpečia, že z_i sa budú "zapĺňať postupne", t.j. ak z_i je nenulová, tak z_{i-1} je na svojej maximálnej hodnote. Pridáme tri binárne premenné $\alpha_1, \alpha_2, \alpha_3$, kde $\alpha_i \in \mathbb{Z}$, $0 \leq \alpha_i \leq 1$, pričom chceme, aby premenná $\alpha_i \in \{0, 1\}$ udávala, či je z_{i+1} ne-nulová. Obmedzenia

$$\begin{aligned} 2\alpha_1 &\leq z_1 \leq 2 \\ \alpha_2 &\leq z_2 \leq \alpha_1 \\ 2\alpha_3 &\leq z_3 \leq 2\alpha_2 \\ 0 &\leq z_4 \leq 3\alpha_3 \end{aligned}$$

vynútia požadované vlastnosti: napríklad ak $\alpha_1 = 0$, tak $0 \leq z_1 \leq 2$, ale $z_2 = z_3 = z_4 = 0$. Ak ale $\alpha_1 = 1$, $z_1 = 2$ a $\alpha_2 \leq z_2 \leq 1$. S takto nastavenými premennými z_1, \dots, z_4 ľahko vyjadríme úcelovú funkciu

$$f(x, y, z_1, \dots, z_4, \alpha_1, \dots, \alpha_3, \delta_1, \dots, \delta_5) = 1 + \frac{1}{2}z_1 + 4z_2 - 2z_3 - \frac{1}{2}z_4$$

Prezentované príklady snáď utvrdili čitateľovu intuiciu, že pomocou ILP sa dajú zapísat oveľa zložitejšie problémy, ako pomocou lineárnych programov, a teda aj riešenie ILP by malo byť oveľa tažšie. Ukážeme si, že tomu tak naozaj je: už aj úloha zistíť, či daný ILP má vôbec nejaké prípustné

riešenie, je NP -úplná, a teda neočakávame, že by sme našli polynomiálny algoritmus na riešenie ILP. Pre úplnosť zopakujeme definíciu problému SAT

Definícia 3.2. Majme n logických premenných $x_1, \dots, x_n \in \{\text{true}, \text{false}\}$. Literál je premenná x_i alebo jej negácia \bar{x}_i . Klaузula je dizjunkcia literálov $C = l_1 \vee l_2 \vee \dots \vee l_{k_C}$. Formula v konjunktívnej normálnej forme (CNF) je konjunkcia klauzí $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$. Problém SAT je rozhodovací problém, v ktorom je na vstupe daná formula F v CNF a úlohou je zistit, či existuje ohodnotenie premenných x_1, \dots, x_n tak, aby F bola splnená.

Základné tvrdenie z teórie je Cook-Lewinova veta, ktorá hovorí, že problém SAT je NP -úplný, t.j. keby existoval polynomiálny algoritmus pre SAT, platilo by $P = NP$. Teraz ukážeme, že keby existoval polynomiálny algoritmus A, ktorý pre daný ILP zistí, či má nejaké prípustné riešenie, existoval by aj polynomiálny algoritmus A' riešiaci SAT. Predpokladajme, že máme algoritmus A a ideme popísat, ako pracuje A'. Nech vstupná formula F pozostáva z klaузí C_1, \dots, C_m . Označme P_i množinu premenných, ktoré sa vyskytujú v C_i ako pozitívne literály, a N_i množinu tých premenných, ktoré sa v C_i vyskytujú v negovaných literáloch. Napríklad pre klaузulu $C_i = (x_1 \vee \bar{x}_2 \vee x_3)$ je $P_i = \{x_1, x_3\}$ a $N_i = \{x_2\}$. Zostrojíme zadanie ILP I tak, že každej logickej premennej x_i zo vstupnej formuly bude prirodzeným spôsobom zodpovedať (rovnako nazvaná) premenná $x_i \in \mathbb{Z}$ s obmedzeniami $0 \leq x_i \leq 1$. Splňujúce ohodnotenie musí splňať aspoň jeden literál v každej klauzule C_i , čo sa dá vyjadriť obmedzením

$$\sum_{x \in P_i} x + \sum_{x \in N_i} (1 - x) \geq 1$$

Napríklad pre formulu

$$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$

dostaneme obmedzenia

$$\begin{aligned} x_1 + x_2 + x_3 &\geq 1 \\ (1 - x_1) + (1 - x_2) + (1 - x_3) &\geq 1 \\ x_1 + (1 - x_2) + (1 - x_3) &\geq 1 \\ (1 - x_1) + (1 - x_2) + x_3 &\geq 1 \\ (1 - x_1) + x_2 + x_3 &\geq 1 \\ x_1, x_2, x_3 &\geq 0 \\ x_1, x_2, x_3 &\leq 1 \end{aligned}$$

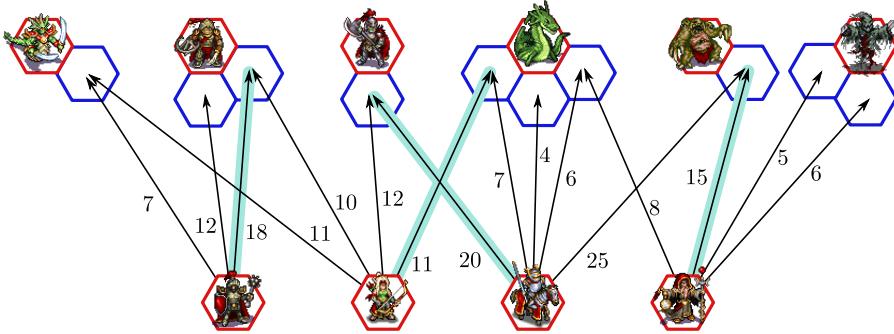
Zjavne každé splňujúce ohodnotenie F je prípustným riešením I a naopak. Preto F je splniteľná práve vtedy, ak existuje prípustné riešenie I .

Cvičenie. Skúste iné známe NP -tažké problémy (napr. problém obchodného cestujúceho, nájdanie najväčšej kliky v grafe, ...) redukovať na ILP.

Naše predchádzajúce úvahy môžme zhrnúť takto: ILP je všeobecný formalizmus, pomocou ktorého sa dajú zapísať mnohé optimalizačné problémy. Táto výrazová sila je zároveň jeho slabinou, lebo neočakávame, že by existoval efektívny algoritmus, ako ILP riešiť.

Dobrý: MAX-WEIGHTED-BIPARTITE-MATCHING

Predpokladajme, že máme za úlohu napísať program, ktorý hrá strategickú hru. V jednom tahu má k dispozícii niekoľko figúr, z ktorých každá môže potenciálne zaútočiť na niektorú nepriateľskú figúru (najprv sa musí presunúť na niektoré prilahlé políčko, pričom sa dve figúry nemôžu presunúť na to isté políčko). Každý potenciálny útok má číselne vyjadrenú silu a cieľom je presunúť figúry tak, aby celková sila útoku bola čo najväčšia.



Každá z figúr v spodnom riadku sa môže presunúť na niektoré z vyznačených modrých políčok a zaútočiť na nepriateľskú figúru; čísla udávajú silu útoku. Najsilnejší útok je zvýraznený.

Problém, ktorý riešime, je známy ako problém najväčšieho bipartitného párovania¹:

Definícia 3.3. Majme daný bipartitný graf s hranami ohodnotenými nezápornými reálnymi číslami. Cieľom problému MAX-WEIGHTED-BIPARTITE-MATCHING je nájsť množinu hrán s najväčším súčtom váh tak, aby žiadne dve vybraté hrany nezdieľali vrchol.

Problém MAX-WEIGHTED-BIPARTITE-MATCHING sa ľahko zapíše ako ILP. Nech je na vstupe bipartitný graf $G = (V, E)$ s ohodnotením $\omega : E \mapsto \mathbb{R}^+$. Pre každú hranu $e \in E$ budeme mať premennú $x_e \in \{0, 1\}$, ktorá bude určovať, či je daná hra vybraná. Cieľom je maximalizovať súčet váh vybratých hrán, t.j. $\max \sum_{e \in E} \omega_e x_e$. Obmedzenia musia zabezpečiť, aby z okolia každého vrchola v bola vybraná najviac jedna hra. Celý ILP potom vyzerá takto (obmedzenia $x_e \leq 1$ netreba písat explicitne, nakoľko vyplývajú z ostatných):

$$\begin{aligned} &\text{maximalizovať } \sum_{e \in E} \omega_e x_e \\ \text{pri obmedzeniach } &\sum_{\substack{e \in E \\ e=(v,w)}} x_e \leq 1 \quad \forall v \in V \\ &x_e \geq 0 \quad \forall e \in E \\ &x_e \in \mathbb{Z} \end{aligned} \tag{19}$$

Napríklad pre graf vľavo dostaneme obmedzenia

$x_{e_1} \leq 1$
 $x_{e_2} + x_{e_4} \leq 1$
 $x_{e_3} + x_{e_5} \leq 1$
 $x_{e_1} + x_{e_2} + x_{e_3} \leq 1$
 $x_{e_4} + x_{e_5} \leq 1$
 $x_{e_1}, x_{e_2}, x_{e_3}, x_{e_4}, x_{e_5} \geq 0$

Kedže na riešenie ILP nemáme efektívny algoritmus, môžme skúsiť vyriešiť *relaxovaný* LP, kde z programu (19) vynecháme obmedzenia $x_e \in \mathbb{Z}$. Zjavne, všetky prípustné riešenia ILP sú aj prípustnými riešeniami relaxovaného programu, preto optimum relaxovaného programu je horný odhad optimálneho riešenia ILP. Ako sme však videli, riešiť relaxovaný program vo všeobecnosti nie je dobrý nápad, lebo optimálne riešenie relaxovaného problému môže byť od optimálneho riešenia pôvodného ILP veľmi ďaleko. Lenže my sa teraz nezaoberáme všeobecným ILP, ale konkrétné takým, ktoré sme dostali uvedeným postupom z nejakého bipartitného grafu. Môžme teda dúfať,

¹Tento problém je trochu výnimka: väčšina problémov prezentovaných v tomto kurze sú NP-ťažké, ale MAX-WEIGHTED-BIPARTITE-MATCHING nie. Čitateľ možno pozná efektívny algoritmus na riešenie tohto problému, napriek tomu ho poprosíme, aby sledoval nás prístup z didaktívnych dôvodov.

že dodatočná štruktúra obmedzení spôsobí, že riešenie relaxovaného problému bude niesť nejakú informáciu o pôvodnom ILP. Skúsme napríklad pre obmedzenia z (20) nastaviť $\omega_{e_1} = 1$ a $\omega_{e_2} = \dots = \omega_{e_5} = 10$ a vyriešiť relaxovaný program. Ak spustíme simplexový algoritmus², na naše potešenie bude nájdené optimálne riešenie celočíselné, aj keď existujú aj neceločíselné optimálne riešenia (napr. $x_{e_1} = 0, x_{e_2} = \dots = x_{e_5} = \frac{1}{2}$). Môžeme to skúsiť s inými váhami aj na iných grafoch a výsledky budú rovnaké: simplexový algoritmus vždy nájde celočíselné optimálne riešenie relaxovaného programu, ktoré je na základe predchádzajúcich úvah optimálnym riešením ILP. Skúsme teraz zistiť príčinu tohto správania. Napíšme si nás program v maticovom tvare

$$\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

kde

$$\mathbf{c} = \begin{pmatrix} 1 \\ 10 \\ 10 \\ 10 \\ 10 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_{e_1} \\ x_{e_2} \\ x_{e_3} \\ x_{e_4} \\ x_{e_5} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Matica A je tzv. *matica incidencie* bipartitného grafu, t.j. matica, ktorá má riadok pre každý vrchol a stĺpec pre každú hranu, pričom v každom stĺpci sú práve dve jednotky, a to práve v riadkoch prislúchajúcich koncovým vrcholom danej hrany. Zároveň vidíme, že podmienka $x_{e_1} \leq 1$ vyplýva z nezápornosti a podmienky $x_{e_1} + x_{e_2} + x_{e_3} \leq 1$, a tak prvý riadok môžeme vyniechať. Pridáme rezervné premenné a dostaneme program v normálnom tvare:

$$\max\{\mathbf{c}^T \mathbf{x} \mid \tilde{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\} \tag{21}$$

kde

$$\mathbf{c} = \begin{pmatrix} 1 \\ 10 \\ 10 \\ 10 \\ 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \tilde{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_{e_1} \\ x_{e_2} \\ x_{e_3} \\ x_{e_4} \\ x_{e_5} \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Pre každú bázu B , je matica \tilde{A}_B regulárna štvorcová matica 4×4 a nenulové premenné príslušného bázového riešenia sú riešením systému

$$\tilde{A}_B \mathbf{x}_B = \mathbf{b},$$

takže podľa Cramerovho pravidla je i -ta zložka riešenia

$$\frac{\det(\tilde{A}_B \langle i \rangle)}{\det(\tilde{A}_B)}$$

kde $\tilde{A}_B \langle i \rangle$ je matica, ktorú dostaneme z \tilde{A}_B , ak i -ty stĺpec nahradíme vektorom \mathbf{b} . Napríklad pre bázu $B = (2, 4, 7, 8)$ dostávame systém

$$\tilde{A}_B \mathbf{x}_B = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_{e_2} \\ x_{e_4} \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

²Pre väčší dramatický efekt to odporúčame čitateľovi naozaj vyskúšať.

Pri použití Cramerovho pravidla potrebujeme tieto determinanty:

$$\det(\tilde{A}_B) = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix} = 1$$

$$\begin{vmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{vmatrix} = 0 \quad \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix} = 1 \quad \begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{vmatrix} = 1 \quad \begin{vmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{vmatrix} = 1$$

Dostali sme riešenie $x_{e_2} = 0, x_{e_4} = 1, s_2 = 1, s_4 = 1$ čo zodpovedá (neoptimálnemu) celočíselnému riešeniu, v ktorom je vybratá iba hrana e_4 . Rezervné premenné $s_2 = s_3 = 1$ hovoria, že vrcholy v_3 a v_4 (druhý a tretí riadok matice \tilde{A}) majú voľnú kapacitu. Namiesto toho, aby sme takýmto spôsobom overili všetkých $\binom{9}{4} = 126$ potenciálnych báz, pokúsime sa tento postup zovšeobecniť.

Definícia 3.4. Štvorcová matica A s celočíselnými prvками sa nazýva *unimodulárna*, ak $\det(A) = \pm 1$, kde \det označuje determinant matice.

Matica (nie nutne štvorcová) A s celočíselnými prvками sa nazýva *totálne unimodulárna* (TUM), ak každá jej regulárna štvorcová podmatica (t.j. matica rozmerov $k \times k$, ktorá vznikne a A výberom nejakých k riadkov a nejakých k stĺpcov) je unimodulárna

Táto definícia špeciálne znamená, že všetky prvky TUM matice A sú $0, \pm 1$: každý prvok je totiž podmatica rozmerov 1×1 .

Veta 3.5. Majme lineárny program v normálnom tvare $\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, kde A je TUM matica a \mathbf{b} je celočíselný vektor. Potom všetky bázové riešenia sú celočíselné.

Dôkaz: Uvažujme bázu B . A_B je štvorcová regulárna matica a nenulové zložky bázového riešenia sú riešenia systému $A_B \mathbf{x}_B = \mathbf{b}$ a dajú sa explicitne vyjadriť ako

$$\frac{\det(A_B \langle i \rangle)}{\det(A_B)}.$$

Kedže A je TUM, $\det(A_B) = \pm 1$. Pre výpočet determinantu $\det(A_B \langle i \rangle)$ použijeme jeho rozvoj podľa i -teho stĺpca

$$\det(A_B \langle i \rangle) = (-1)^{i+1} b_1 C_1 + \cdots + (-1)^{i+m} b_m C_m,$$

kde C_j sú determinnty štvorcových podmatíc A , a teda $C_j \in \{0, \pm 1\}$. Kedže podľa predpokladu $b \in \mathbb{Z}$, dôkaz je hotový. \square

V skutočnosti sme použili iba slabšiu vlastnosť ako unimodularita: stačilo nám aby determinant regulárnej podmatice $m \times m$ bol ± 1 a ostatné determinanty štvorcových podmatíc boli celočíselné. Ničmenej trieda problémov s TUM maticami je dostatočne bohatá. Ukážeme teraz niekoľko tvrdení o TUM:

Veta 3.6. Nech A je TUM rozmerov $n \times m$ a nech $k \leq m$. Potom $B := \left(\begin{array}{c|c} A & \begin{matrix} 0 \\ I_k \end{matrix} \end{array} \right)$, kde I_k je identická matica rozmerov $k \times k$, je TUM.

Dôkaz: Nech C je ľubovoľná regulárna štvorcová podmatica matice B , t.j. $C = (C_1 \mid C_2)$, kde C_1 sú stĺpce matice A a C_2 sú stĺpce z I . Riadky C preusporiadame tak, že riadky, ktoré sú nulové v C_2 dáme na vrch, a teda

$$C = \left(\begin{array}{c|c} A' & 0 \\ \hline X & I_\ell \end{array} \right)$$

kde A' je štvorcová podmatica A . Kedže C je regulárna, platí $\det(C) = \pm 1$. \square

Veta 3.7. Majme lineárny program v tvaru $\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, kde A je TUM matica a \mathbf{b} je celočíselný vektor. Potom všetky bázové riešenia sú celočíselné.

Dôkaz: Podľa Vety 3.6 je matica $(A \mid I)$, kde I je identická matica $m \times m$, TUM; tvrdenie vyplýva z postupu normalizácie lineárnych programov (matica I zodpovedá rezervným premenným). \square

Veta 3.8. Nech A je matica s prvkami $a_{ij} \in \{0, \pm 1\}$, ktorej riadky sa dajú rozdeliť na dve množiny I_1, I_2 tak, že

1. Ak má nejaký stĺpec dva prvky s rovnakým znamienkom, ich príslušné riadky sú v rôznych množinách
2. Ak má nejaký stĺpec dva prvky s rôznym znamienkom, ich príslušné riadky sú v rovnakej množine.

Potom A je TUM.

Dôkaz: Potrebujeme dokázať, že každá regulárna štvorcová podmatica $k \times k$ má determinant ± 1 . Toto tvrdenie dokážeme indukciou na veľkosť podmatíc. Bázu indukcie tvoria podmatice 1×1 , t.j. prvky matice. Predpokladajme teraz, že tvrdenie platí pre podmatice rozmerov $k-1 \times k-1$. Zoberme si ľubovoľnú štvorcovú podmaticu C rozmerov $k \times k$. Ak C má všetky stĺpce nulové, je singulárna. Ak má nejaký stĺpec s práve jedným nenulovým prvkom (t.j. ± 1), použijeme rozvoj determinantu podľa tohto stĺpca a z indukčného predpokladu dostaneme $\det(C) = \pm 1$. Nech teda každý stĺpec C má aspoň dva nenulové prvky. Z podmienok vety ale vyplýva, že pre každý stĺpec j platí

$$\sum_{i \in I_1} c_{ij} = \sum_{i \in I_2} c_{ij}$$

Preto lineárna kombinácia riadkov je 0 a $\det(C) = 0$. \square

Dôsledok 3.9. Každý lineárny program tvaru

$$\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

alebo

$$\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\},$$

kde \mathbf{b} je celočíselný vektor a A je

1. matica incidencie neorientovaného bipartitného grafu, alebo
2. matica incidencie orientovaného grafu³

má všetky bázové riešenia celočíselné.

Dôkaz: V prípade 1) označme I_1 riadky prislúchajúce vrcholom jednej bipartície, v prípade 2) budú v I_1 všetky riadky. Použijeme vetu 3.8. \square

Teraz vidíme, že problém MAX-WEIGHTED-BIPARTITE-MATCHING je iba jedným z neveľkej triedy problémov, kde simplexový algoritmus na relaxovanom lineárnom programe vždy nájde celočíselné optimum. Pochopiteľne, pri NP -tažkých problémoch už také šťastie nebudem mať. Napriek tomu, niekedy relaxovaný lineárny program pomôže nájsť aspoň približné riešenie, ako uvidíme v ďalších častiach.

Cvičenie. Zapíšte problém hľadania najkratšej cesty ako lineárny program.

³matica, ktorá má riadok pre každý vrchol a stĺpec pre každú orientovanú hranu, pričom v stĺpci prislúchajúcim hrane (v_i, v_j) je na i -tom riadku 1, a na j -tom riadku -1 (ostatné prvky sú nulové).

Zlý: MIN-VERTEX-COVER

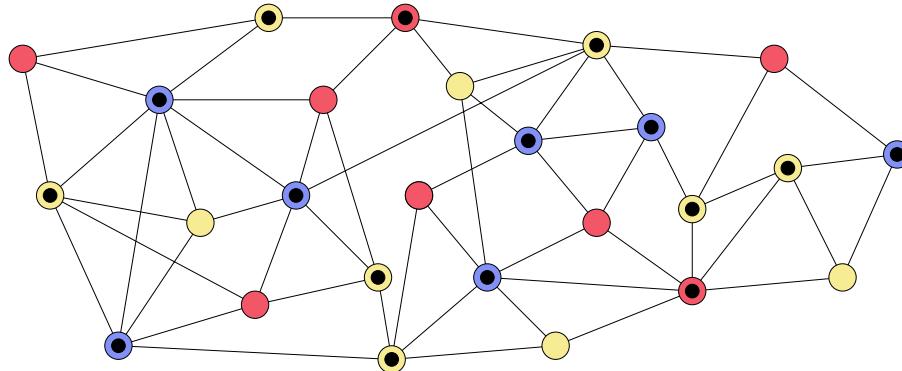
Problémy, v ktorých relaxovaný lineárny program nájde optimálne riešenie, sú skôr výnimkou. Aj v iných prípadoch však môže riešenie relaxovaného problému pomôcť. Ako príklad použijeme aplikáciu z výpočtovej biológie (viď. [?]). Jedinec jedného druhu majú veľmi podobný genóm, ktorý sa líší iba na niekoľkých miestach. Cieľom je identifikovať tzv. *snipy* (*Single Nucleotide Polymorphism*) v genóme, čo sú miesta v genóme, kde sa môžu vyskytovať rôzne hodnoty, pričom naokolo je rovnaký obsah. Genóm je sada sekvenovaných fragmentov a vstupom je matica, ktorá udáva hodnotu (A, B alebo – pre neurčené) daného snipu na danom fragmente. Cieľom je nájsť ofarbenie fragmentov dvoma farbami tak, aby zodpovedali dvom alternatívnym genómom.

	s_1	s_2	s_3	s_4	s_5
f_1		A	B	A	
f_2		B	A	B	
f_3			–		A
f_4			A		B
f_5	A	A	B	–	
f_6	B	B	B	B	

Konfigurácia fragmentov f_4, f_5 a snipov s_2, s_3 tvorí konflikt: podla snipu s_2 musia fragmenty f_4 a f_5 mať rôzne farby, podla snipu s_3 potom ale všetky hodnoty s_3 musia byť B , čiže s_3 nie je snip. Keby hodnota s_3 na fragmente f_6 bola A , zelené a červené fragmenty by zodpovedali dvom alternatívnym genómom.

Vstupné dátá vždy obsahujú chyby. Konfigurácia dvoch fragmentov a dvoch snipov ako na predchádzajúcim obrázku tvorí konflikt: jeden zo snipov musel byť prečítaný zle a treba ho ignorovať. Ak máme pre každý snip číselne vyjadrenú jeho významnosť, môžeme zostrojiť konfliktový graf snipov, t.j. graf, kde snipy sú vrcholy a dva snipy, ktoré sú v konflikte sú spojené hranou. Prvým krokom k nájdeniu alternatívnych genómov je vyhodiť čo možno najmenej snipov tak, aby vo výsledku neboli konflikty. Prirodzene tak dostávame inštanciu problému MIN-VERTEX-COVER:

Definícia 3.10. Daný je jednoduchý graf $G = (V, E)$ s vrcholmi ohodnotenými nezápornými vähami, t.j. funkciou $\omega : V \mapsto \mathbb{R}^+$. Cieľom problému MIN-VERTEX-COVER je vybrať množinu vrcholov tak, aby každá hraha bola incidentná aspoň s jedným vybratým vrcholom a celková váha vybratých vrcholov bola minimálna.



Príklad vrcholového pokrytie. Modré vrcholy majú váhu 1, žlté váhu 2 a červené váhu 13. Celé pokrytie má váhu 47. Dá sa nájsť pokrytie s menšou váhou?

Úloha sa dá jednoducho zapísat ako celočíselný lineárny program. Pre každý vrchol $v \in V$ zavedieme premennú $x_v \in \{0, 1\}$, ktorá indikuje, či je daný vrchol vybratý. Celý lineárny program potom vyzerá nasledovne:

$$\begin{aligned}
& \text{minimalizovať} && \sum_{v \in V} \omega_v x_v \\
\text{pri obmedzeniach} & x_u + x_v &\geq 1 & \forall e = (u, v) \in E \\
& x_v &\geq 0 & \forall v \in V \\
& x_v &\in \mathbb{Z}
\end{aligned} \tag{22}$$

Účelová funkcia hovorí, že sa snažíme minimalizovať súčet váh vybraných vrcholov. Pre každú hranu $(u, v) \in E$ máme obmedzenie tvaru $x_u + x_v \geq 1$, ktoré zaručí, že aspoň jeden z koncových vrcholov hrany je vybratý. Obmedzenia $x_v \geq 0$ a $x_v \in \mathbb{Z}$ spolu zaručia $x_v \in \{0, 1\}$: ak je v nejakom prípustnom riešení niektorá hodnota $x_v > 1$, nastavením $x_v = 1$ ostanú všetky obmedzenia splnené a, keďže váhy sú nezáporné, hodnota účelovej funkcie sa nezväčší.

Problém MIN-VERTEX-COVER je NP -tažký, a tak neočakávame, že by sme program (22) vedeli riešiť v polynomiálnom čase. Uspokojíme sa preto s aproximáciou: chceme nájsť riešenie, ktorého váha nikdy nie je väčšia ako dvojnásobok váhy optimálneho riešenia. Takýto algoritmus sa označuje ako 2-aproximačný (v nasledujúcej definícii r nemusí byť konštantá, preto potrebujeme brať supréum cez všetky vstupy danej velkosti):

Definícia 3.11. Majme optimalizačný problém \mathcal{P} a polynomiálny algoritmus \mathcal{A} , ktorý ho rieši. Nech $m^*(\mathbf{x})$ je hodnota optimálneho riešenia pre vstup \mathbf{x} a $m_{\mathcal{A}}(\mathbf{x})$ je hodnota, ktorú vráti algoritmus \mathcal{A} . *Aproximačný pomer algoritmu \mathcal{A} na vstupe \mathbf{x} je*

$$R_{\mathcal{A}}(\mathbf{x}) = \begin{cases} \frac{m^*(\mathbf{x})}{m_{\mathcal{A}}(\mathbf{x})} & \text{ak } \mathcal{P} \text{ je maximalizačný} \\ \frac{m_{\mathcal{A}}(\mathbf{x})}{m^*(\mathbf{x})} & \text{ak } \mathcal{P} \text{ je minimalizačný} \end{cases}$$

Algoritmus \mathcal{A} je $r(n)$ -aproximačný, ak

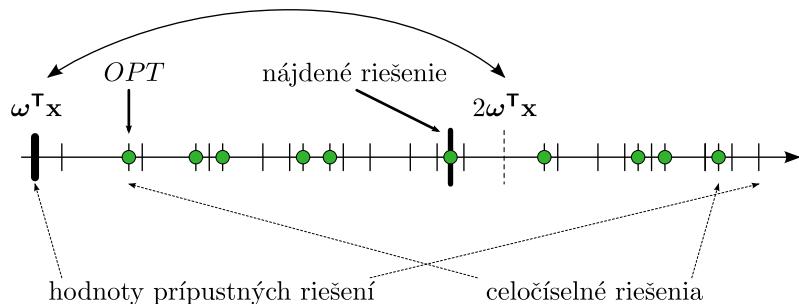
$$\forall n : \sup_{|\mathbf{x}|=n} R_{\mathcal{A}}(\mathbf{x}) \leq r(n)$$

pričom supréum sa berie cez všetky vstupy dĺžky n .

Začneme tým, že z programu (22) odstránime obmedzenia $x_v \in \mathbb{Z}$, čím dostaneme relaxovaný lineárny program

$$\begin{aligned}
& \text{minimalizovať} && \sum_{v \in V} \omega_v x_v \\
\text{pri obmedzeniach} & x_u + x_v &\geq 1 & \forall e = (u, v) \in E \\
& x_v &\geq 0 & \forall v \in V
\end{aligned} \tag{23}$$

ktorý vieme efektívne vyriešiť. Nech teraz OPT je optimálne (celočíselné) riešenie programu (22) a \mathbf{x} je optimálne riešenie programu (23). Zjavne $\omega^T \mathbf{x} \leq OPT$ (relaxáciou sme množinu prípustných riešení iba zväčšili, preto optimum sa mohlo iba zmeniť). Otázka, ktorá nás trápi, je, ako daleko je OPT od $\omega^T \mathbf{x}$ a ako dostať z \mathbf{x} celočíselné riešenie, ktoré nebude ďaleko od OPT . V tomto prípade sa ukáže, že máme štastie a $OPT \leq 2\omega^T \mathbf{x}$. Situácia je ako na nasledujúcom obrázku:



Spomedzi možných hodnôt úcelovej funkcie je $\omega^\top \mathbf{x}$ minimálna. Nás zaujíma OPT , čo je minimálne celočíselné riešenie. Ak sa nám podarí skonštruovať celočíselné riešenie s váhou nanajvýš $2\omega^\top \mathbf{x}$, máme zaručené, že jeho váha je najviac $2OPT$.

Najjednoduchší spôsob, ako z \mathbf{x} dostať celočíselné riešenie je zaokrúhlit hodnoty x_v . Vieme, že všetky $x_v \geq 0$ a takisto bez ujmy na všeobecnosti $x_v \leq 1$ (v optimálnom riešení nemá zmysel mať hodnotu $x_v > 1$: keby to tak bolo, nastavením $x_v = 1$ sa hodnota $\omega_v x_v$ nezväčší a všetky obmedzenia ostanú splnené), preto zaokrúhlenie vlastne znamená vybrať tie vrcholy, pre ktoré $x_v \geq \frac{1}{2}$. Najprv si treba uvedomiť, že riešenie, ktoré takto dostaneme, je prípustné: ak v pôvodnom riešení platilo $x_u + x_v \geq 1$ pre hranu (u, v) , tak aspoň jedna z hodnôt $x_u, x_v \geq \frac{1}{2}$ a teda ju pri zaokrúhlení nastavíme na 1. Fakt, že zaokrúhlené riešenie je najviac dvojnásobok optimálneho riešenia vyplýva z toho, že pre zaokrúhlené riešenie $\hat{\mathbf{x}}$ platí

$$\omega^\top \hat{\mathbf{x}} = \sum_{v \in V} \omega_v \hat{x}_v = \sum_{\substack{v \in V \\ x_v \geq \frac{1}{2}}} \omega_v \hat{x}_v \leq 2 \sum_{\substack{v \in V \\ x_v \geq \frac{1}{2}}} \omega_v x_v \leq 2 \sum_{v \in V} \omega_v x_v = 2\omega^\top \mathbf{x}.$$

Prvá rovnosť je iba rozpísaný skalárny súčin. V druhej rovnosti sme využili, že vrcholy v , pre ktoré $x_v < \frac{1}{2}$, majú $\hat{x}_v = 0$. Ďalšia nerovnosť vyplýva z toho, že všetky zostávajúce vrcholy majú $x_v \geq \frac{1}{2}$ a $\hat{x}_v = 1$.

To, že jednoduché zaokrúhlenie pomerne dobre zafungovalo, je dôsledkom špeciálnej štruktúry relaxovaného programu (23). O chvíľu ukážeme, že každé prípustné bázové riešenie programu (23) je *poloceločíselné*, t.j. $x_v \in \{0, \frac{1}{2}, 1\}$. Z toho vyplýva, že existuje optimálne poloceločíselné riešenie a keď v ňom nahradíme všetky $\frac{1}{2}$ jednotkami, celková váha sa najviac zdvojnásobí. Pri dôkaze využijeme nasledujúcu všeobecnú lemu, ktorá hovorí, že v lineárnom programe prípustné bázové riešenia tvoria vrcholy \mathcal{D} : bod, ktorý nie je vrchol, leží na nejakej úsečke, ktorá je celá vohnutri telesa, a teda sa dá napísat ako $t\mathbf{y} + (1-t)\mathbf{z}$ pre dva body \mathbf{y}, \mathbf{z} a $0 < t < 1$.

Lema 3.12. Majme ľubovoľný lineárny program v normálnom tvaru. Potom žiadne prípustné riešenie \mathbf{x} , ktoré je konvexnou kombináciou prípustných riešení \mathbf{y}, \mathbf{z} (t.j. $\mathbf{x} = t\mathbf{y} + (1-t)\mathbf{z}$ pre nejaké $t : 0 < t < 1$) nie je bázové.

Dôkaz: Majme prípustné bázové riešenie \mathbf{x} a predpokladajme sporom, že je konvexnou kombináciou prípustných riešení \mathbf{y}, \mathbf{z} . Nech B je bázová množina riešenia \mathbf{x} , t.j. A_B je regulárna štvorcová matica a $x_j = 0$ pre všetky $j \notin B$. Uvažujme vektor \mathbf{e} taký, že $\tilde{c}_j = 0$ pre $j \in B$ a $\tilde{c}_j = -1$ inak. Zjavne $\mathbf{e}^\top \mathbf{x} = \mathbf{0}$ a zároveň pre ľubovoľné $\mathbf{v} \geq \mathbf{0}$ platí $\mathbf{e}^\top \mathbf{v} \leq \mathbf{0}$. Navyše, ak \mathbf{v} má aspoň jednu zložku $v_j > 0$ pre $j \notin B$, tak $\mathbf{e}^\top \mathbf{v} < \mathbf{0}$. Vieme, že $\mathbf{x} = t\mathbf{y} + (1-t)\mathbf{z}$ a teda $\mathbf{0} = \mathbf{e}^\top \mathbf{x} = t\mathbf{e}^\top \mathbf{y} + (1-t)\mathbf{e}^\top \mathbf{z}$ pre $\mathbf{y}, \mathbf{z} \geq \mathbf{0}$. To je možné iba vtedy, ak $\mathbf{e}^\top \mathbf{y} = \mathbf{e}^\top \mathbf{z} = \mathbf{0}$, a teda \mathbf{y} aj \mathbf{z} majú všetky $y_j = z_j = 0$, $j \notin B$. Lenže $\mathbf{x}, \mathbf{y}, \mathbf{z}$ sú prípustné, preto $A\mathbf{x} = A\mathbf{y} = A\mathbf{z} = \mathbf{b}$. Kedže $\mathbf{x}, \mathbf{y}, \mathbf{z}$ majú nebázové zložky nulové, máme $A_B \mathbf{x} = A_B \mathbf{y} = A_B \mathbf{z} = \mathbf{b}$. A_B je však regulárna štvorcová matica a preto má jednoznačné riešenie, takže $\mathbf{x} = \mathbf{y} = \mathbf{z}$. \square

Vieme teda, že nám stačí ukázať, že každé prípustné riešenie \mathbf{x} programu (23), ktoré priradí nejakému vrcholu hodnotu $x_v \notin \{0, \frac{1}{2}, 1\}$ sa dá napísat ako konvexná kombinácia dvoch prípustných riešení \mathbf{y} a \mathbf{z} .

Zoberme si ľubovoľné prípustné riešenie \mathbf{x} , ktoré priradí nejakému vrcholu hodnotu $x_v \notin \{0, \frac{1}{2}, 1\}$. Vrcholy, v ktorých \mathbf{x} má iné hodnoty ako $\{0, \frac{1}{2}, 1\}$ rozdelme do dvoch skupín takto

$$V_+ = \left\{ v \mid 0 < x_v < \frac{1}{2} \right\} \quad V_- = \left\{ v \mid \frac{1}{2} < x_v < 1 \right\}$$

Zafixujme si konštantu ε a definujme dve riešenia \mathbf{y} a \mathbf{z} takto:

$$y_v = \begin{cases} x_v + \varepsilon, & \text{ak } x_v \in V_+ \\ x_v - \varepsilon, & \text{ak } x_v \in V_- \\ x_v, & \text{inak} \end{cases} \quad z_v = \begin{cases} x_v - \varepsilon, & \text{ak } x_v \in V_+ \\ x_v + \varepsilon, & \text{ak } x_v \in V_- \\ x_v, & \text{inak} \end{cases}$$

Zjavne $\mathbf{y} \neq \mathbf{z}$ a $\mathbf{x} = \frac{1}{2}(\mathbf{y} + \mathbf{z})$. Takisto ľahko vidieť, že pri voľbe $\varepsilon < \min\{x_v \mid v \in V\}$ budú $\mathbf{y}, \mathbf{z} \geq \mathbf{0}$. Ostáva nám ukázať, že \mathbf{y}, \mathbf{z} splňajú obmedzenia na hranach. Zoberme si hranu (u, v) , kde $x_u + x_v > 1$. Ak $\varepsilon < \frac{1}{2}(x_u + x_v - 1)$, tak toto obmedzenie je splnené v \mathbf{y} aj \mathbf{z} . Ak $x_u + x_v = 1$, sú dve možnosti: bud $x_u, x_v \in \{0, \frac{1}{2}, 1\}$ alebo $u \in V_+, v \in V_-$ (alebo naopak). V každom prípade, pre ľubovoľnú voľbu ε , aj tátó podmienka ostane splnená v \mathbf{y} aj \mathbf{z} .

Na tomto mieste sme čitatelovi dlžní vysvetlenie. Našli sme 2-aproximačný algoritmus a tvárime sa spokojne. Prečo? V realite je predsa riešenie, ktoré by bolo dvojnásobkom hľadaného optimá, zväčša nanič. Odpovedí, prečo sú algoritmy s dokázateľným aproximačným pomerom zaujímavé, je niekoľko.

Teoretický⁴ záujem. Napriek silným výsledkom teórie zložitosti, ani po desaťročiach snahy priadne nerozumieme, prečo niektoré problémy sú "ľahké" a iné "tažké", či už "tažkosť" znamená optimálne riešenie alebo approximáciu. Sú problémy, pre ktoré sa dá nájsť v polynomiálnom čase ľubovoľne presná approximácia. Na druhej strane sú iné problémy, pre ktoré nevieme nájsť ani veľmi slabý aproximačný algoritmus (napríklad s pomerom $n^{0.99}$), ale vieme, že existencia takého algoritmu by znamenala $P = NP$. Ak o nejakom probléme ukážeme, že preňho existuje napríklad 2-aproximačný algoritmus, zaradí sa tak do hierarchie známych problémov s podobnými vlastnosťami a dá sa dúfať, že túto podobnosť časom budeme vedieť nejak využiť.

Použitie ako zarázka. Väčšinou sa na riešenie (NP -)tažkých problémov používajú rôzne heuristiky, buď navrhnuté špecificky pre daný problém alebo metaheuristiky typu simulované žíhanie, tabu-search, či neurónové siete. Heuristiky väčšinou nájdú riešenie pomerne blízko optimu, ale v zriedkavých prípadoch môžu skončiť s veľmi, ale ozaj veľmi zlým riešením. Aproximačné algoritmy sú zvyčajne rýchle, o niekoľko rádov rýchlejšie ako niektoré metaheuristiky. Preto je rozumné spustiť rýchly výpočet, ktorý zaručí, že ani v zlom prípade nebude riešenie katastroficky zlé.

Použitie ako heuristika. Dokázaná hranica je najhorší prípad spomedzi všetkých vstupov. Navyše pri dôkaze aproximačného pomeru treba väčšinou odhadnúť hodnotu optimálneho riešenia, a tieto odhady sú často pomerne hrubé. Dá sa dúfať, a nezriedka je to aj pravda, že navrhnuté algoritmy dávajú na "bežných" inštanciách oveľa lepšie výsledky, ako garantovaná hranica. Analyzovať "bežné" inštancie je ale ošemetná záležitosť. Analýza priemerného prípadu sa spolieha na apíornu informáciu o pravdepodobnostnom rozdelení vstupov; v realite sú ale dokonale náhodné vstupy veľkou výnimkou. V prípade, že algoritmus je plánovaný pre konkrétné nasadenie, dajú sa použiť vstupy z podobných situácií (napr. pri použití MIN-VERTEX-COVER problému pre hľadanie snipov sa dajú na testovanie použiť známe konfliktové grafy snipov vo viere, že v skutočnom nasadení budú mať vstupy podobné vlastnosti). Keď nás ale zaujíma algoritmus ako taký (napríklad ako súčasť nejakej všeobecnej knižnice), pojem "bežnej" inštancie nedáva zmysel a musíme pracovať s najhorším prípadom. V tomto smere je výhodou mnohých algoritmov založených na lineárnom programovaní, že dávajú s každým riešením aj odhad optimu z opačnej strany (v našom prípade je hodnota relaxovaného optimu dolným odhadom pre skutočné optimum); máme teda pre každú inštanciu interval, v ktorom vieme, že sa skutočné optimum nachádza.

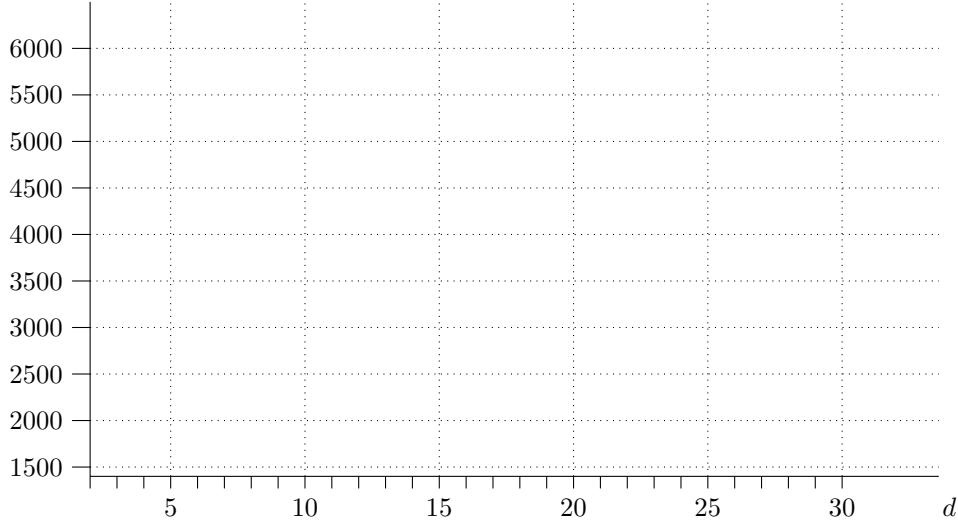
Ako ilustráciu sa pozrieme na náš 2-aproximačný algoritmus pre MIN-VERTEX-COVER. Ako prvý test použijeme napríklad zoznam všetkých súvislých kubických⁵ grafov na 20 vrcholoch (je ich 510489). V neváhovanom prípade (t.j. všetky vrcholy majú váhu 1) sú všetky relaxované riešenia tvaru $x_{v_1} = \dots = x_{v_{20}} = \frac{1}{2}$; optimum relaxovaného programu je teda 10 a po zaokruhlení máme triviálne riešenie s hodnotou 20. Priemerné celočíselné optimum je ≈ 11.5 . Ak sme každému vrcholu nastavili váhu náhodné číslo z rozsahu $\{1, \dots, 100\}$, v riešení relaxovaného programu sa oveľa viac vyskytovali hodnoty 0 a 1 a dostali sme priemerný aproximačný faktor 1.352433. Pri váhach tvaru $1 + 2^i$, kde i je náhodné číslo z rozsahu $\{1, \dots, 10\}$, to bolo dokonca iba 1.041625 a ak sme navyše pridali dva vrcholy spojené so všetkými vrcholmi pôvodného grafu, výsledok bol 1.086788.

Teraz sa pozrieme na náhodné grafy. Testovali sme 60-vrcholové grafy; pre očakávaný priemerný stupeň d bola každú dvojicu vrcholov (u, v) nezávislá pravdepodobnosť $d/59$, že budú spojené hranou. Spomedzi takto vygenerovaných grafov sme pre každú hodnotu d vybrali 1000 takých,

⁴autor nevnáma slovo "teoretický" derogatívne

⁵t.j. každý vrchol má práve troch susedov

ktoré boli súvislé. Nasledujúci obrázok ukazuje závislosť priemerného relaxovaného optimu, zaokrúhleného riešenia a skutočného optimu pre rôzne hodnoty d s váhami tvaru $1 + 2^i$, kde i je náhodné číslo z rozsahu $\{1, \dots, 10\}$.



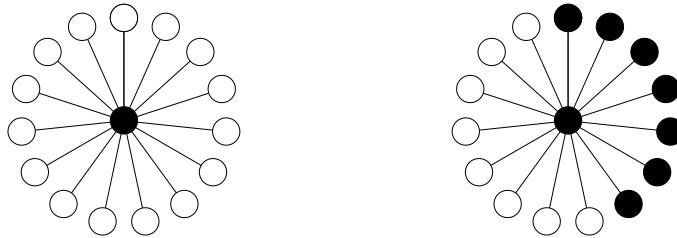
Vidíme, že pre riedke grafy (s očakávaným stupňom cca do 7), je očakávané relaxované riešenie veľmi blízke celočíselnému optimu. Zároveň má veľa položiek celočíselných, takže zaokrúhlením sa príliš nestráca (pre $d \approx 3$ je optimalizačný pomer ≈ 1.01 , t.j. chyba je 1%, pri $d \approx 5$ je chyba cca 8%). Ako stúpa hustota, pribúda neceločíselných položiek v relaxovanom riešení a tým sa zväčšuje rozdiel medzi celočíselným a relaxovaným optimom, aj medzi relaxovaným optimom a zaokrúhlením. Pri očakávanom stupni cca 15 začne byť relaxovaé riešenie zložené takmer zo samých $\frac{1}{2}$, takže zaokrúhlením sa získava dvojnásobok; zároveň ale stúpa celočíselné optimum, takže aproximačný faktor postupne klesá. Vidíme teda, že aj pre jednoduchý algoritmus, ktorý má garantovaný aproximačný pomer 2, existujú triedy vstupov, kde dosahuje oveľa lepší pomer.

Škaredý: MAX-INDEPENDENT-SET

V predchádzajúcej časti sme si za úlohu dali minimalizovať celkovú váhu vyhodených snipov tak, aby z každej hrany konfliktového grafu bol vyhodený aspoň jeden snip. Tú istú úlohu môžeme formulovať aj tak, že chceme zachovať čo najviac snipov tak, aby žiadne dva vybraté neboli spojené hranou. Dostávame tak problém MAX-INDEPENDENT-SET:

Definícia 3.13. Daný je jednoduchý graf $G = (V, E)$ s vrcholmi ohodnotenými nezápornými váhami, t.j. funkciou $\omega : V \mapsto \mathbb{R}^+$. Cieľom problému MAX-INDEPENDENT-SET je vybrať množinu vrcholov tak, aby žiadne dva vybraté vrcholy neboli spojené hranou a aby celková váha vybratých vrcholov bola maximálna.

Pre každú nezávislú množinu S zvyšné vrcholy $V - S$ tvoria vrcholové pokrytie: keďže S je nezávislá, každá hrana je incidentná najviac s jedným vrcholom z S a teda aspoň s jedným vrcholom z $V - S$. Naopak, pre každé vrcholové pokrytie C tvoria zvyšné vrcholy $V - C$ nezávislú množinu: každá hrana je incidentná aspoň s jedným vrcholom z C , a teda žiadne dva vrcholy z $V - C$ nie sú spojené hranou. Problémy MIN-VERTEX-COVER a MAX-INDEPENDENT-SET sú preto z pohľadu optimálneho riešenia ekvivalentné. Z pohľadu approximácie sa ale správajú veľmi odlišne.



V neváhovanom hviezdicovom grafe s n vrcholmi je maximálna nezávislá množina S^* veľkosti $n - 1$ a minimálne vrcholové pokrytie $V - S^*$ veľkosti 1. Biele vrcholy vpravo tvoria nezávislú množinu S veľkosti $n/2$, ktorá je 2-aproximáciou optimálnej, ale $n/2$ vrcholov z množiny $V - S$ je iba $n/2$ -aproximáciou optimálneho vrcholového pokrycia.

Rovnako ako problém MIN-VERTEX-COVER, aj MAX-INDEPENDENT-SET sa dá zapísat ako celočíselný program

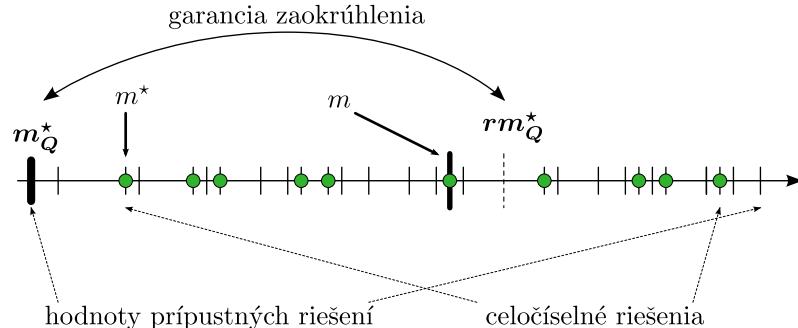
$$\begin{aligned} \text{maximalizovať} \quad & \sum_{v \in V} \omega_v x_v \\ \text{pri obmedzeniach} \quad & x_u + x_v \leq 1 \quad \forall e = (u, v) \in E \\ & x_v \geq 0 \quad \forall v \in V \\ & x_v \in \mathbb{Z} \end{aligned} \tag{24}$$

Ked budeme uvažovať špeciálny prípad neváhovaných grafov, t.j. $\omega_{v_1} = \dots = \omega_{v_n} = 1$ podobne ako v prípade MIN-VERTEX-COVER budú pre husté grafy optimálne riešenia relaxovaného programu tvaru $x_{v_1} = \dots = x_{v_n} = \frac{1}{2}$. Tu sa však podobnosť s MIN-VERTEX-COVER končí: je totiž vela hustých grafov, ktoré majú malú maximálnu nezávislú množinu, ale aj v nich má relaxované optimálne riešenie hodnotu aspoň $n/2$. Nedá sa teda nájsť zaokrúhľovací algoritmus, ktorý by vždy vrátil celočíselné riešenie blízko (napríklad o konštantný faktor) relaxovaného optima. Nie je to náhoda; z výsledkov v [7] vyplýva, že ak by existoval akýkoľvek polynomiálny algoritmus, ktorý by riešil problém MAX-INDEPENDENT-SET s aproximačným pomerom $n^{1-\varepsilon}$ pre nejaké $\varepsilon > 0$, potom $P = NP$.

V tejto kapitole sme ukázali, že rôzne optimalizačné problémy sa dajú zapísat ako celočíselné lineárne programy; v niektorých prípadoch má optimum relaxovaného programu úzky vzťah s celočíselným optimom a v iných prípadoch spolu vôbec nesúvisia. V ďalších kapitolách si ukážeme, ako využiť relaxované lineárne programy na získanie (pomerne) dobrého celočíselného riešenia.

3.2 Deterministické zaokrúhľovanie

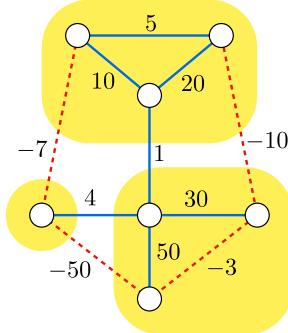
V predchádzajúcej kapitole sme pri probléme MIN-VERTEX-COVER použili nasledovnú všeobecnú schému na navrhovanie r -aproximačných algoritmov:



Ak hľadáme minimum celočíselného programu s hodnotou m^* , nájdeme (potenciálne menšie) minimum relaxovaného programu s hodnotou m_Q^* , zaokrúhlime ho a dostaneme riešenie s hodnotou m . Ak ukážeme, že pri zaokrúhľovaní hodnota riešenia stúpla najviac r -krát, máme zaručený r -aproximačný algoritmus. To, čo sme nazvali zaokrúhlenie, však nemusí byť jednoduché aritmetické zaokrúhlenie, ale hocjaký deterministický algoritmus. Uvedieme teraz príklad takéhoto rafinovaného zaokrúhľovania.

MIN-MULTI-CUT

Ako motivačný príklad zoberme nasledovný problém⁶: máme databázu textov, v ktorých sa vyskytujú odkazy na rôzne osoby a našim cieľom je rozlísiť, kedy sa hovorí o tej istej osobe a kedy nie. To nemusí byť také ľahké, ako sa na prvý pohľad zdá, lebo tá istá osoba sa niekedy referuje celým menom, niekedy iniciálami, niekedy prezývkou, niekedy nepriamo pozíciou (napr. "britský premiér") a pod. Predpokladajme, že máme k dispozícii (napríklad ako výstup algoritmov strojového učenia na tréningovej množine dát) nejaký prediktor, čo je funkcia, ktorá pre dve referencie vráti reálne číslo, pričom čím väčšia kladná hodnota, tým väčšia šanca, že ide o tú istú osobu a čím menšia záporná hodnota, tým skôr ide o dve rôzne osoby (0 znamená, že ani prediktor netuší). Máme teda daný graf, ktorého vrcholy sú jednotlivé výskyty a hrany sú ohodnotené kladnými alebo zápornými číslami. Našim cieľom je rozdeliť vrcholy na komponenty, pričom každý komponent zodpovedá jednej osobe. Kedže prediktor nie je úplne spoľahlivý, nie vždy existuje rozdelenie, ktoré je s ním konzistentné. Chceme preto nájsť rozdelenie, ktoré minimalizuje celkovú chybu: ak kladná hrana spája dva komponenty, alebo záporná hrana je vvnútri komponentu, ich váha (v absolútnej hodnote) sa priráta k celkovej chybe.

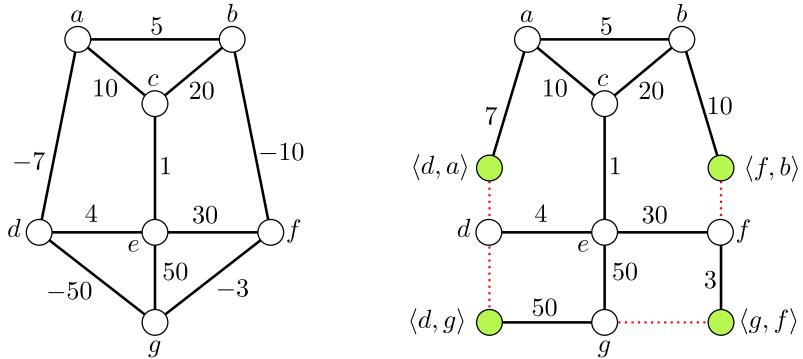


Graf a jeho rozdelenie s celkovou chybou 8: kladné hrany s váhami 4 a 1 idú medzi rôznymi komponentami a záporná hrana s váhou -3 je vnútri komponentu.

Úloha nájsť rozdelenie s minimálnou chybou je NP -tažká, preto sa skromne uspokojíme s približným riešením. Pretransformujme si nás graf takto: kladné hrany zachováme. Pre každú zápornú hranu (u, v) s váhou $-w$ pridáme do grafu nový vrchol $\langle u, v \rangle$, ktorý spojíme hranou váhy w s vrcholom v . Dostaneme tak graf s kladným ohodnotením hrán a v ňom budeme riešiť nasledovnú úlohu: chceme z neho odobrať hrany s minimálnou váhou tak, aby v zostávajúcom grafe neboli vrcholy $\langle u, v \rangle$ a u v rovnakom komponente súvislosti.

Cvičenie. Dokážte, že táto transformácia je ekvivalentná, t.j. ak máme riešenie pôvodného problému s celkovou chybou c , existuje riešenie transformovaného problému s váhou c a naopak, ak máme riešenie transformovaného problému s váhou c , tak existuje riešenie pôvodného problému s celkovou chybou c .

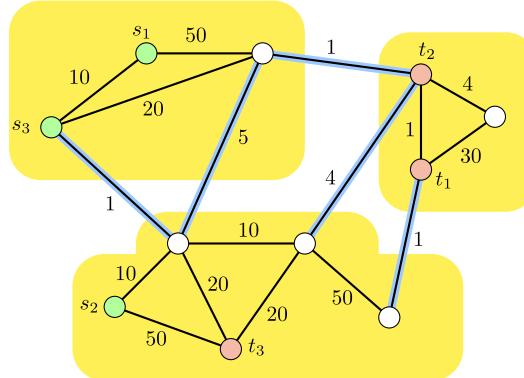
⁶porov. [2, 5]



Pôvodný a transformovaný graf (kedže pôvodný graf je neorientovaný, transformácia nie je jednoznačná). Pridané vrcholy sú zelené, bodkovanou čiarou sú spojené vrcholy, ktoré musia byť v rôznych komponentoch.

Transformovaný problém je špeciálnym prípadom úlohy MIN-MULTI-CUT:

Definícia 3.14. Majme daný jednoduchý graf $G = (V, E)$ s hranami ohodnotenými nezáporými váhami, t.j. funkciou $\omega : E \mapsto \mathbb{R}^+$ a v ňom k dvojíc vrcholov (s_i, t_i) , $i = 1, \dots, k$. Cieľom problému MIN-MULTI-CUT je odobrat z grafu G množinu hrán s minimálnou celkovou váhou tak, aby žiadna dvojica (s_i, t_i) nebola v rovnakom komponente súvislosti výsledného grafu.



Riešenie inštancie problému MIN-MULTI-CUT s celkovou cenou 12.

Špeciálnym prípadom pre $k = 1$ je problém nájdenia minimálneho rezu, ktorý je polynomiálne riešiteľný, ale pre všeobecné k je MIN-MULTI-CUT NP -tažký. Pokúsmo sa teraz formulovať problém MIN-MULTI-CUT ako celočíselný lineárny program. Pre každú hranu e budeme mať premennú $x_e \in \{0, 1\}$, ktorá bude určovať, či sme hranu odstránilí alebo nie. Chceme odstrániť hrany s minimálnou váhou, preto chceme minimalizovať výraz $\sum_{e \in E} x_e \omega(e)$. Obmedzeniami potrebujeme zaručiť, aby vrcholy (s_i, t_i) boli v rôznych komponentoch. Označme si \mathcal{P}_{s_i, t_i} všetky $s_i - t_i$ cesty, t.j. všetky cesty v G , ktoré začínajú v s_i a končia v t_i . Potrebujeme, aby na každej z nich bola vybratá aspoň jedna hrana. Dostávame teda program (obmedzenie $x_e \leq 1$ nemusíme písat explícitne, lebo vyplýva z minimalizácie: ak je nejaké $x_e > 1$, môžeme ho zmenšiť na 1, obmedzenia ostanú zachované a hodnota riešenia klesne):

$$\begin{aligned}
 &\text{minimalizovať} && \sum_{e \in E} x_e \omega_e \\
 &\text{pri obmedzeniach} && \sum_{e: e \in \pi} x_e \geq 1 \quad \forall i \in \{1, \dots, k\}, \quad \forall \pi \in \mathcal{P}_{s_i, t_i} \\
 &&& x_e \geq 0 \quad \forall e \in E \\
 &&& x_e \in \mathbb{Z}
 \end{aligned} \tag{25}$$

Program (25) teraz relaxujeme tak, že odstráňme obmedzenia $x_e \in \mathbb{Z}$. Dostaneme lineárny program

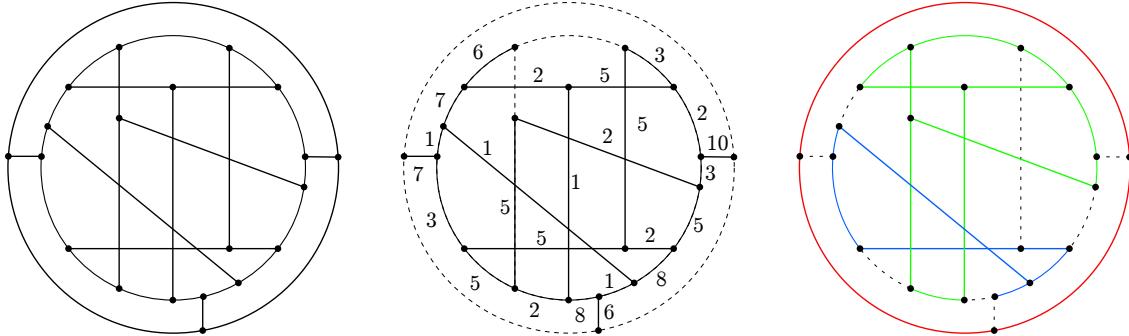
$$\begin{array}{ll} \text{minimalizovať} & \sum_{e \in E} x_e \omega_e \\ \text{pri obmedzeniach} & \begin{array}{l} \sum_{e: e \in \pi} x_e \geq 1 \quad \forall i \in \{1, \dots, k\}, \quad \forall \pi \in \mathcal{P}_{s_i, t_i} \\ x_e \geq 0 \quad \forall e \in E \end{array} \end{array} \quad (26)$$

ktorý môžme interpretovať nasledovne: hranám chceme priradiť dĺžky x_e tak, aby každá dvojica vrcholov (s_i, t_i) bola vzdialenosť aspoň 1. Zároveň, ak ω_e bude prierez hrany, tak $\sum_{e \in E} x_e \omega_e$ hovorí, že chceme minimalizovať objem všetkých hrán (hrubé hrany chceme mať krátke). Nás relaxovaný program má ale ešte jednu chybu: má potenciálne exponenciálne veľa obmedzení, takže simplexovým algoritmom ho nevieme efektívne vyriešiť. Upravme ho preto takto: uvažujme všetky cesty $\pi \in \mathcal{P}_{s_i, t_i}$. Namiesto explicitnej podmienky $\sum_{e \in \pi} x_e \geq 1$ pre každú cestu, priradme každému vrcholu v potenciál $p_v^{(i)} \in \mathbb{R}$ tak, že $p_{t_i}^{(i)} - p_{s_i}^{(i)} \geq 1$. Každej ceste $\pi : s_i = v_1, v_2, \dots, v_z = t_i$ prislúcha postupnosť potenciálov vrcholov $p_{v_1}^{(i)}, p_{v_2}^{(i)}, \dots, p_{v_z}^{(i)}$. Ak vynútimo, aby dĺžka každej hrany cesty bola aspoň rozdiel potenciálov, t.j. $x_{(v_j, v_{j+1})} \geq p_{v_{j+1}}^{(i)} - p_{v_j}^{(i)}$, budeme mať zaručené, že celá cesta bude mať dĺžku aspoň jedna. Namiesto programu (26) nám teda stačí riešiť polynomiálne veľký program

$$\begin{array}{ll} \text{minimalizovať} & \sum_{e \in E} x_e \omega_e \\ \text{pri obmedzeniach} & \begin{array}{l} x_e \geq p_v^{(i)} - p_u^{(i)} \quad \forall i \in \{1, \dots, k\}, \quad \forall e \in E, \quad e = (u, v) \\ x_e \geq p_u^{(i)} - p_v^{(i)} \\ p_{t_i}^{(i)} - p_{s_i}^{(i)} \geq 1 \quad \forall i \in \{1, \dots, k\} \end{array} \end{array} \quad (27)$$

Na jednej strane, každé riešenie programu (27) spĺňa podmienky programu (26), na druhej strane, ak máme riešenie programu (26), nastavíme potenciály $p_v^{(i)}$ ako vzdialenosť v od s_i , kde za dĺžku hrany berieme x_e . Lahko vidno, že takto nastavené potenciály splňajú podmienky programu (27), a teda programy (26) a (27) sú ekvivalentné.

Program (27) môžme vyriešiť napr. simplexovým algoritmom a dostaneme optimálne riešenie s hodnotami premenných x_e^* a cenou $m_Q^* = \sum_{e \in E} x_e^* \omega_e$. Keby všetky $x_e^* \in \{0, 1\}$, mali by sme priamo riešenie programu (25) a teda aj problému MIN-MULTI-CUT. Keď to skúsime s grafom z predchádzajúceho obrázka, zistíme, že optimálne riešenie (27) je celočíselné. To vyzerá nádejne, tak môžme skúsiť napríklad takýto experiment: zoberieme všetkých 510489 kubických (t.j každý vrchol je susedný s troma inými) grafov na 20 vrcholoch, váhy hrán ponecháme na 1, a pre každý graf si náhodne vygenerujeme $k \in \{1, \dots, 20\}$ dvojic s_i, t_i . Experiment dopadol tak, že spomedzi 510489 riešení bolo 268178 (52.5%) celočíselných, 135328 (26.5%) poloceločíselných a dokopy bolo 458008 (89.7%) riešení s najmenšou nenulovou hodnotou $\geq \frac{1}{4}$. Jednoduchým zaokruhlením všetkého nenulového nahor preto v 89.7% prípadov dostaneme prinajhoršom 4-aproximáciu. Prv, než by nás nedôslednosť zviedla k nejakým umáhleným záverom, pozrime si bližšie tých 10.3% zvyšných prípadov. Skutočnosť totiž môže byť taká, že takéto zlé prípady nie sú až také ojedinelé, iba v našich testovacích dátach sa vyskytujú zriedkavo. Zlé boli napospol prípady, v ktorých bolo veľké k , t.j. veľa dvojíc s_i, t_i . Zoberme si niektorý z testovaných grafov a skúsme vyriešiť inštanciu, do ktorej zahrnieme všetky dvojice vrcholov vo vzdialnosti aspoň 4.



Vľavo je graf s 20 vrcholmi a 30 hranami (všetky hrany majú váhu 1). Uvažujeme inšanciu MIN-MULTI-CUT, v ktorej treba oddeliť každú dvojicu vrcholov vo vzdialosti aspoň 4 (priemer grafu je 5, dvojíc vo vzdialosti aspoň 4 je 22). V strede je optimálne riešenie relaxovaného LP (čísla na hranach sú násobky $\frac{1}{15}$) s hodnotou 7. Kedže iba päť hrán v riešení je nulových, zaokrúhlením nahor získame riešenie s hodnotou 25. Vpravo je celočíselné riešenie s hodnotou 8: po odstránení 8 čiarkovaných hrán sa graf rozpadne na tri komponenty a každý z nich má priemer najviac 3.

Vidíme, že jednoduché zaokrúhlenie môže dávať zlé výsledky, a preto má zmysel rozmýšľať nad lepším "zaokrúhlovacím" algoritmom. Označme m^* optimálnu cenu riešenia problému MIN-MULTI-CUT: platí platí $m_Q^* \leq m^*$. Ukážeme si teraz, ako "zaokrúhlit" hodnoty x_e^* (t.j. vybrať hrany do rezu) tak, aby sme dostali riešenie problému MIN-MULTI-CUT s cenou najviac $4 \ln(2k)m_Q^*$.

Náš "zaokrúhľovací" algoritmus bude postupne z grafu "vyhrýzať" množiny V_1, V_2, \dots, V_k . V prvej iterácii nájde množinu V_1 tak, že $s_1 \in V_1$, $t_1 \notin V_1$ a zároveň žiadna dvojica s_i, t_i nie je vo V_1 . Algoritmus vyberie do rezu hrany oddelujúce V_1 od zvyšku grafu a pokračuje s grafom na vrcholoch $V - V_1$. V i -tej iterácii, ak s_i alebo t_i už nie sú v grafe (aspôň jeden z nich tam určite bude), tak $V_i = \emptyset$, ináč sa nájde V_i tak, aby $s_i \in V_i$ a $t_i \notin V_i$ a aby žiadna dvojica s_j, t_j nebola vo V_i ; V_i sa odstráni z grafu (t.j. vyberú sa do rezu všetky hrany, ktoré oddelujú V_i). Algoritmus takto postupne vytvorí rez, ktorý odseparuje každú dvojicu s_i, t_i . Ako vyberať množiny V_i a využiť pri tom optimálne riešenie relaxovaného problému?

Zoberme si hodnoty x_e^* a predstavme si optimálne riešenie (26) ako sieť potrubí, kde hrana e má prierez ω_e a dĺžku x_e^* , t.j. má objem $\omega_e x_e^*$. Pre hrano $e = (u, v)$ označme $d(u, v) = x_e^*$ a prirodzeným spôsobom môžeme merať vzdialenosť ľuboľovných dvoch vrcholov $d(v_1, v_2)$. Keď algoritmus vyberie nejakú množinu V_i , musí prerezať všetky potrubia, ktoré ju oddelujú od zvyšku grafu, pričom za prerezanie potrubia e zaplatí jeho prierez, t.j. ω_e . Z prípustnosti riešenia vieme, že pre každú dvojicu s_i, t_i je $d(s_i, t_i) \geq 1$ a optimálne riešenie (26) minimalizuje celkový objem hrán, t.j.

$$\Psi := \sum_{e=(u,v) \in E} \omega_e d(u, v)$$

Označme $G'_r = (V'_r, E'_r)$ graf, ktorý vznikne po odstránení množín V_1, \dots, V_{r-1} z grafu G . Prirodzeným spôsobom si zadefinujeme pojmy gule s polomerom ρ :

$$\mathcal{B}_\rho(v) = \{u \in V'_r \mid d(u, v) \leq \rho\}$$

Ak G'_r obsahuje s_r aj t_r , množina V_r bude guľa vhodného polomeru ρ okolo vrchola s_r . Ostáva nám určiť, ako vybrať ρ . V prvom rade chceme, aby $\rho < 1/2$, čo nám zaručí, že žiadna dvojica s_j, t_j nebude ležať vo V_r (vzdialenosť každej dvojice je aspoň 1). Nech vnútorné hrany gule sú

$$\mathcal{E}_\rho(v) = \{(w, z) \in E'_r \mid w, z \in \mathcal{B}_\rho(v)\}$$

a hranová hranica gule je

$$\bar{\mathcal{E}}_\rho(v) = \{(w, z) \in E'_r - \mathcal{E}_\rho(v) \mid w \in \mathcal{B}_\rho(v) \vee z \in \mathcal{B}_\rho(v)\}$$

Objem gule definujeme (mierne neštandardne) tak, že okrem objemu hrán pridáme z technických dôvodov člen Ψ/k :

$$V_\rho(v) = \frac{\Psi}{k} + \sum_{(w,z) \in \mathcal{E}_\rho(v)} \omega_{(w,z)} d(w, z) + \sum_{(w,z) \in \bar{\mathcal{E}}_\rho(v)} \omega_{(w,z)} (\rho - \min(d(v, w), d(v, z)))$$

Vnútorné hrany gule prispievajú do jej objemu celým svojím objemom, hrany z hranovej hranice iba príslušnou časťou. Najviac nás, pochopiteľne, zaujíma cena gule, t.j. veľkosť jej hranového rezu:

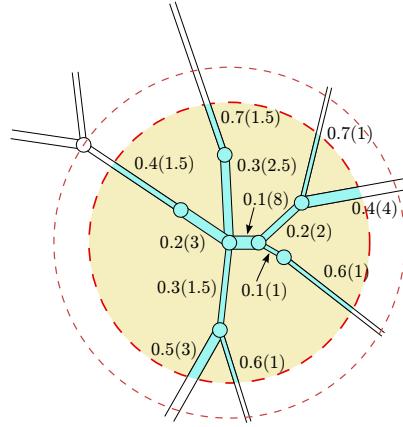
$$C_\rho(v) = \sum_{(w,z) \in \bar{\mathcal{E}}_\rho(v)} \omega_{(w,z)}$$

Konečne sa dostávame k tomu, ako zvliet polomer ρ v množine V_r . Chceme, aby guľa, ktorú odseparujeme mala čo najmenší jednotkovú cenu, t.j.

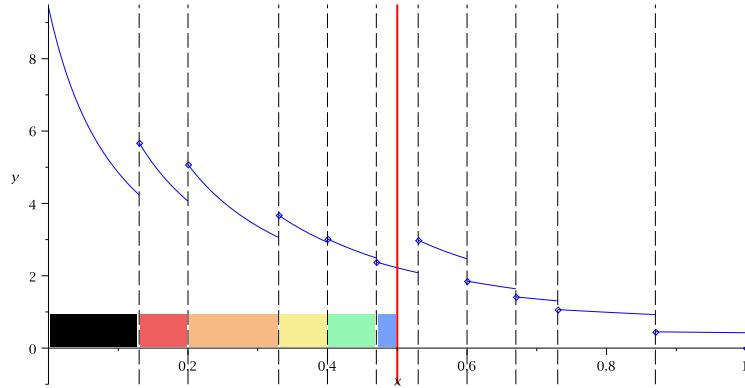
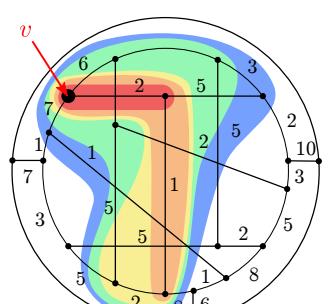
$$F_\rho(v) = \frac{C_\rho(v)}{V_\rho(v)}$$

Funkcia $F : \rho \mapsto F_\rho(v)$ pre daný vrchol v má body nespojitosti pre tie hodnoty ρ , pre ktoré existuje vrchol w vo vzdialosti ρ od v . Na každom intervalu medzi bodmi nespojitosti je F diferencovateľná a klesajúca. Preto môžeme ľahko nájsť minimum $F_\rho(s_i)$ na intervale $(0, 1/2)$: vyberieme minimum z hodnôt $F_\rho(s_i)$ pre $\rho = \delta(s_i, u) - \varepsilon$. Ostáva nám ukázať, že takto získaný rez je naozaj dobrou approximáciou. Na to ukážeme nasledovné tvrdenie:

$$\forall v \exists \rho < 1/2 : F_\rho(v) \leq 2 \ln(2k) \quad (28)$$



Predstava relaxovaného riešenia ako siete potrubí. Pri hranách je napísaná ich dĺžka (x_e^*) a v závitke prierez (ω_e). Guľa s polomerom 0.4 má cenu 13 a objem $\frac{\Psi}{k} + 4.65$. Ak polomer vzrastie na 0.6, cena ostane rovnaká, ale objem stúpne.



Funkcia F pre jeden vrchol v z predchádzajúceho príkladu. Optimum je $\Psi = 7$ a $k = 22$. Do vzdialenosť $\frac{2}{15} \approx 0.13$ je vnútro gule prázdnne a veľkosť rezu je 3, preto pre $0 \leq \rho < \frac{2}{15}$ je $F_\rho(v) = \frac{3}{\frac{7}{22} + 3\rho}$. Pre $\frac{2}{15} \leq \rho < \frac{3}{15} = 0.2$ vnútro gule obsahuje dva vrcholy a celú hranu dĺžky $\frac{2}{15}$; veľkosť rezu je 4, preto na tomto intervale $F_\rho(v) = \frac{4}{\frac{7}{22} + \frac{2}{15} + 2\rho + 2(\rho - \frac{2}{15})} \approx \frac{4}{0.18 + 4\rho}$. Ďalej až po vzdialenosť $\frac{5}{15} = \frac{1}{3}$ vnútro obsahuje tri vrcholy a dve hrany s celkovým objemom 0.2, pričom v reze je 5 hrán, atď.

Z tvrdenia (28) vyplynie požadovaná aproximácia: nech algoritmus vyberie gule $\mathcal{B}_{\rho_1}(s_{i_1}), \dots, \mathcal{B}_{\rho_h}(s_{i_h})$. Cena rezu je

$$m = \sum_{j=1}^h C(s_{i_j}, \rho_j) \leq 2 \ln(2k) \sum_{j=1}^h V(s_{i_j}, \rho_j)$$

V poslednej sume sa členy Ψ/k z definície objemu zosumujú najviac do Ψ a zvyšné objemy opäť najviac do Ψ , a teda $m \leq 2 \ln(2k)2\Psi$.

Dôkaz dokončíme dôkazom tvrdenia (28): Fixujme si vrchol v a uvažujme všetky funkcie ako funkcie vzdialenosťi ρ . Ak $V(\rho)$ je diferencovateľná, platí $V'(\rho) = C(\rho)$. Preto

$$F(\rho) = \frac{V'(\rho)}{V(\rho)} = [\ln V(\rho)]'$$

Tvrdenie (28) dokážeme sporom: predpokladajme, že pre všetky $\rho < 1/2$ platí $F(\rho) > 2 \ln(2k)$. Ak F je diferencovateľná na intervale $(0, 1/2)$, máme

$$[\ln V(\rho)]' > 2 \ln(2k)$$

obe strany zintegrujeme určitým integrálom od 0 po $1/2$ a dostaneme

$$\ln \frac{V(\frac{1}{2})}{V(0)} > \ln 2k$$

a odtiaľ

$$V\left(\frac{1}{2}\right) > 2kV(0) = 2\Psi$$

čo je spor, lebo $V(\rho) \leq \Psi + \Psi/k$. Ak F nie je diferencovateľná na intervale $(0, 1/2)$, zopakujeme predchádzajúcu úvahy na každom inetrvale spojitosti a výsledky sčítame.

Predcchádzajúce úvahy môžme zhrnúť do tvrdenia:

Veta 3.15. Existuje algoritmus, ktorý pre problém MIN-MULTI-CUT vráti v polynomiálnom čase riešenie s veľkosťou nanajvýš $4 \ln(2k)$ -násobku optimálneho.

Garancia, ktorú sme práve dokázali, sa zdá byť pomerne slabá a čakali by sme, že vo veľa prípadoch budú skutočné výsledky nášho algoritmu lepšie. Je to naozaj tak? Môžme prípadne dokázať silnejšiu garanciu? Ukážeme si, že to nie je možné. Zoberme si regulárny graf G , ktorý má n vrcholov a každý vrchol má stupeň $d \geq 3$. Zvolme si nejaký parameter α a uvažujme inštanciu MIN-MULTI-CUT na grafe G , kde všetky hrany majú váhu 1 a dvojice s_i, t_i , ktoré treba rozpojiť, sú všetky dvojice vrcholov vo vzdialenosťi aspoň α . Keď pre každú hranu e zvolíme $x_e = \frac{1}{\alpha}$, dostaneme prípustné riešenie programu (26), a preto $m_Q^* \leq \frac{n}{\alpha}$. Ukážeme teraz, ako dosiahliť, aby ľubovoľné celočíselné riešenie malo veľkosť aspoň n . Nech M je optimálne celočíselné riešenie, t.j. množina hrán, ktorá rozpojí všetky vrcholy vo vzdialenosťi aspoň α . Odstránením M z G dostaneme graf $G' = (V, E - M)$ s niekoľkými komponentami súvislosti, pričom platí:

Lema 3.16. Každý komponent súvislosti G' má najviac d^α vrcholov.

Dôkaz: Predpokladajme sporom, že by existoval komponent K s viac ako d^α vrcholmi. Zoberme si hocjaký vrchol $v \in K$. Vrchol v má v G stupeň d , teda aj v K má najviac d susedov. Každý z nich má opäť najviac d susedov, preto v celom G (a teda aj v K) môže byť najviac $d + d(d-1) < d + d^2$ vrcholov vo vzdialenosťi najviac 2 od v . Pokračujme indukciou a dostaneme, že môže byť najviac $d + d^2 + \dots + d^{\alpha-1}$ vrcholov vo vzdialenosťi nanajvýš α od v . Lenže $1 + d + d^2 + \dots + d^{\alpha-1} = \frac{d^\alpha - 1}{d-1} < d^\alpha$, preto v K existuje vrchol w , ktorý je od v ďalej ako α . Vrcholy v, w ale tvorili nejakú dvojicu s_i, t_i , preto nesmú byť v jednom komponente, čo je spor. \square

Nech teraz S_1, \dots, S_h sú komponenty súvislosti G' . Označme $\delta(S)$ hranovú hranicu množiny S , t.j. tie hrany, ktoré majú jeden koniec v S a druhý mimo S . Pretože každá hrana má dva konce, platí $2|M| = \sum_{i=1}^h |\delta(S_i)|$. Pretože každý vrchol je v nejakom komponente súvislosti (vyhadzovali sa iba hrany), máme $\sum_{i=1}^h |S_i| = n$. V ďalších úvahách použijeme užitočný nástroj: expandéry. Sú to grafy, v ktorých z každej množiny vrcholov odchádza “veľa” hrán.

Definícia 3.17. Graf $G = (V, E)$ nazveme *expandérom*, ak pre každú množinu vrcholov $S \subset V$ platí

$$|\delta(S)| \geq \min\{|S|, |\bar{S}|\},$$

kde $\bar{S} = V - S$.

Predpokladajme teraz, že náš graf G je expandér a zvolme si $\alpha = \lfloor \log_d n/2 \rfloor$. Pretože $d^\alpha \leq n/2$, každý komponent súvislosti v $(V, E - M)$ má najviac $n/2$ vrcholov, a teda dostávame

$$|M| = \frac{1}{2} \sum_{i=1}^h |\delta(S_i)| \geq \frac{1}{2} \sum_{i=1}^h |S_i| = \frac{n}{2}.$$

Ked si to zhrnieme, máme inštanciu na grafe stupňa d s n vrcholmi, kde optimálny multirez má $\Omega(n)$ hrán, ale existuje relaxované riešenie s cenou $O\left(\log d \frac{n}{\log n}\right)$. Zároveň máme $k = \Theta(n^2)$ dvojíc s_i, t_i , lebo pre každý vrchol je aspoň $n/2$ vrcholov vo vzdialosti aspoň α . Ak chceme docieliť, aby parametre n a k boli nezávislé, stačí pre nejaké ℓ nahradíť každú hranu cestou dĺžky ℓ a ponechať dvojice s_i, t_i : počet vrcholov narastie, ale approximačný faktor ostane rovnaký. Vidíme teda, že náš zaokrúhľovací algoritmus nemôže dať v najhoršom prípade lepsí výsledok ako $O(\log k)$ -násobok optima.

Ostáva posledná otázka, či vôbec expandéry existujú a či sú zriedkavé. Existujú a vyskytujú sa často, ale sú plaché. Nájsť deterministické konštrukcie, ktoré by zaručene vyrobili expandér je pomerne náročné. Na druhej strane, nie je príliš fažké ukázať (aj keď to tu neurobíme), že väčšina regulárnych grafov sú expandéry.

Iterované zaokrúhľovanie

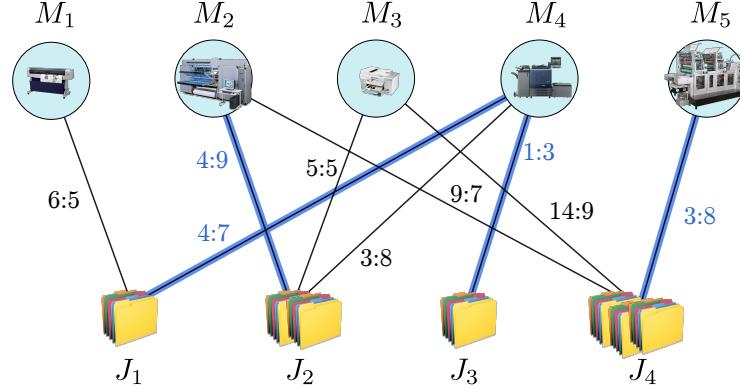
V prípade MIN-VERTEX-COVER sme vyriešili relaxovaný program a podarilo sa nám ukázať, že v bázovom riešení relaxovaného programu má každá premenná nejakú dobrú vlastnosť (konkrétnie, že $x_i \in \{0, \frac{1}{2}, 1\}$), na základe ktorej sme mohli odhadnúť zaokrúhľovaci chybu. Teraz si ukážeme, čo sa dá robiť, ak dobrú vlastnosť vieme ukázať iba o niektorých hodnotách relaxovaného riešenia. Pri použití techniky iterovaného zaokrúhľovania vyriešime relaxovaný lineárny program, potenciálne vyberieme niekoľko premenných, ktorých hodnoty zafixujeme a zo zvyšku vyrobíme menší lineárny program, na ktorom celý postup opakujeme.

Ukážeme si túto techniku na príklade. Predstavme si typografickú firmu, ktorá má k dispozícii rôzne tlačiarne s rôznymi technológiami. Firma dostane zakázku, ktorá pozostáva z niekoľkých úloh. Každá úloha sa potenciálne dá vytlačiť na rôznych tlačiarňach, s rôznym časom spracovania a nákladmi (napríklad čiernobiely text sa dá tlačiť na offsetovej tlačiarni, alebo laserovej tlačiarni, alebo aj na farebnom plotri – tam to bude ale dlho trvať a bude to drahé). Zakázka má na spracovanie časový limit a cielom je naplánovať úlohy na tlačiarne tak, aby sa stihol termín a zároveň náklady boli čo najmenšie. To nás vedie k nasledujúcej formulácii:

Definícia 3.18. Majme množinu procesorov M a množinu úloh J , kde $|M| = m$ a $|J| = n$.

Pre každú dvojicu $i \in M, j \in J$ máme daný čas $t_{ij} \geq 0$ a cenu $c_{ij} \geq 0$ spracovania úlohy j na procesore i . Zároveň je dané číslo T . Cielom problému GENERALIZED-ASSIGNMENT je priradiť každej úlohe $j \in J$ procesor $u_j \in M$ tak, aby sa minimalizovala celková cena priradenia, t.j. $\sum_{j \in J} c_{u_j j}$. Zároveň sa každý procesor musí zmestíť do časového limitu T , t.j. pre každé $i \in M$ platí $\sum_{j: u_j = i} t_{ij} \leq T$.

Prirodzená vizualizácia je pomocou bipartitného grafu $G = (V, E)$ s bipartíciou $V = M \uplus J$, pričom pre dvojicu $i \in M, j \in J$ je $(i, j) \in E$, ak $t_{ij} \leq T$.



Optimálne priradenie s cenou 12 pre limit $T = 10$. Prvé číslo na hrane udáva cenu, druhé čas.

Problém GENERALIZED-ASSIGNMENT je algoritmicke ťažko riešiteľný. Aj keď uvažujeme veľmi špeciálny prípad dvoch procesorov s tým, že pre každú úlohu $p_{1j} = p_{2j} = p_j$ (t.j. časy spracovania sú na obidvoch procesoroch rovnaké) a $\sum_{j \in J} p_j = 2T$, rozhodnút, či sa dajú úlohy podeliť tak, aby na každom procesore bol čas najviac T je ekvivalentné známemu NP -úplnému problému MIN-PARTITION⁷. Neočakávame preto, že by sme ho vedeli efektívne riešiť. Ukážeme riešenie trochu jednoduchejšej úlohy. Povedzme, že naša typografická firma pri prevzatí zakázky slúbila dodaciu dobu "5 až 10 dní". Budeme sa snažiť riešiť úlohu s termínom $T = 5$ dní, ale ak ho trochu zmeškáme, nič zlé sa nestane, kým to nebude viac ako 10 dní. Algoritmus, ktorý ukážeme, bude myšlený na práve takúto situáciu. Bude riešiť problém GENERALIZED-ASSIGNMENT s tým, že nájde priradenie s optimálnou cenou. Čas môže byť aj väčší ako T , ale nikdy neprekročí $2T$ (tu si treba povšimnúť, že to nie je to isté, ako riešiť priamo problém s limitom $2T$, lebo ten môže mať menšiu optimálnu cenu).

Problém GENERALIZED-ASSIGNMENT si najprv formulujeme ako celočíselný lineárny program. Pre každú dvojicu $i \in M, j \in J$ takú, že $(i, j) \in E$ si zavedieme premennú $x_{ij} \in \{0, 1\}$, ktorá vyjadruje, či procesor i má pridelenú úlohu j . Definícia 3.18 nám priamočiaro dáva ILP:

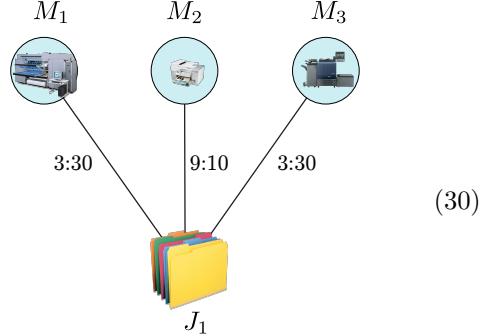
$$\begin{aligned}
 & \text{minimalizovať} \quad \sum_{(i,j) \in E} x_{ij} c_{ij} \\
 & \text{pri obmedzeniach} \quad \sum_{i \in M} x_{ij} = 1 \quad \forall j \in J \\
 & \quad \sum_{j \in J} t_{ij} x_{ij} \leq T \quad \forall i \in M \\
 & \quad x_{ij} \in \{0, 1\} \quad \forall i \in M, \forall j \in J
 \end{aligned} \tag{29}$$

Relaxácia programu (29) sa dá rozumieť tak, že úlohu nemusíme celú priradiť jednému procesoru, ale rôzne časti môžme dať rôznym procesorom. Keď skúsime takto vyriešiť inštanciu z príkladu vyššie, na chvíľu by mohla skrsnúť nádej, pretože optimálne riešenie je celočíselné. Stačí však, aby sme namiesto $T = 10$ zobrali $T = 7$ a optimálne priradenie bude $J_1 \mapsto M_1$, $J_2 \mapsto M_3$, $J_3 \mapsto M_4$ a $J_4 \mapsto M_2$ s cenou 21, pričom existuje relaxované riešenie

$$x_{11} = \frac{3}{7} \quad x_{22} = \frac{49}{72} \quad x_{24} = \frac{1}{8} \quad x_{32} = \frac{23}{72} \quad x_{34} = 0 \quad x_{41} = \frac{4}{7} \quad x_{42} = 0 \quad x_{43} = 1 \quad x_{54} = \frac{7}{8}$$

⁷Problém MIN-PARTITION je definovaný nasledovne: pre množinu čísel $A = \{a_1, \dots, a_n\}$ treba rozhodnúť, či existuje partícia $A = B \uplus C$ taká, že $\sum_{x \in A} x = \sum_{x \in B} x$.

s cenou $\frac{7019}{504} \approx 13.9$. Vidno aj, v čom je problém: napríklad M_5 je cenovo výhodná, ale pomalá. V celočíselnom riešení ju nemôžeme použiť, ale v relaxovanom jej môžme prideliť takú porciu úlohy J_4 , aby sa presne vyplnil limit. Napriek tomu vidíme, že aj v tomto relaxovanom riešení sú niektoré hodnoty celočíselné a na tom by sme chceli založiť našu stratégiju: zafixovať celočíselné hodnoty a zvyšok riešiť nejakým iným lineárny programom. Zaujíma nás preto, či môže existovať inštancia, ktorá nemá v optimálnom relaxovanom riešení žiadne celočíselné hodnoty. Obrázok vpravo ukazuje, že môže. Zdá sa, že náš pôvodný plán využíta celočíselné hodnoty z relaxovaného riešenia sa dostal do slepej uličky. Ako sa však ukáže, optimálne riešenia, ktoré nemajú žiadne celočíselné hodnoty majú veľmi špeciálny tvar, a tak sa nám predsalen podarí situáciu zachrániť. Pri návrhu iteratívneho algoritmu by sa neskôr ukázalo, že potrebujeme riešiť jemne všeobecnejšiu verziu relaxovaného programu. Ušetríme si preto robotu a uvedieme ju hneď od začiatku. Namiesto všetkých procesorov budeme vyžadovať splnenie termínu iba od nejakej podmnožiny $M' \subseteq M$; navyše, každý procesor bude mať potenciálne iný termín. Dostávame teda program



Optimálne priradenie pre limit $T = 10$ je M_2 s cenou 9. Optimum relaxovaného riešenia je $x_{11} = x_{21} = x_{31} = \frac{1}{3}$, s cenou 5.

$$\begin{aligned}
 & \text{minimalizovať} \quad \sum_{(i,j) \in E} x_{ij} c_{ij} \\
 & \text{pri obmedzeniach} \quad \begin{aligned}
 & \sum_{i \in M} x_{ij} = 1 \quad \forall j \in J \\
 & \sum_{j \in J} t_{ij} x_{ij} \leq T_i \quad \forall i \in M' \\
 & x_{ij} \geq 0 \quad \forall i \in M, \forall j \in J
 \end{aligned} \tag{31}
 \end{aligned}$$

Relaxácia programu (29) je špeciálnym prípadom (31) pre $M' = M$ a $T_i = T$. V ďalšom texte budeme symbolom $\deg(k)$, pre $k \in J \cup M$ označovať stupeň procesora alebo úlohy v grafe G (t.j. počet hrán, ktoré z neho vychádzajú). Pre celý algoritmus je klíčová nasledovná charakterizácia optimálnych riešení:

Lema 3.19. Nech \mathbf{x} je optimálne bázové riešenie programu (31), kde pre všetky i, j je $0 < x_{ij} < 1$. Potom existuje procesor $i \in M'$, pre ktorý buď $\deg(i) \leq 1$, alebo $\deg(i) = 2$ a $\sum_{j \in J} x_{ij} \geq 1$.

Lema 3.19 hovorí, že ak sa po vyriešení programu (31) ocitneme v situácii, kde nie je žiadna celočíselná premenná, tak buď máme procesor, ktorý dokáže spracovať najviac jednu úlohu ($d(i) \leq 1$), alebo procesor, ktorý dokáže spracovať práve dve úlohy a z každej z nich spracovať kus (navyše, v súčte sú tie kusy ≥ 1). Pri dôkaze lemy 3.19 využijeme nasledujúce pomocné tvrdenie:

Lema 3.20. Nech \mathbf{x} je bázové riešenie programu (31), v ktorom všetky $x_{ij} > 0$. Potom existuje množina $M'' \subseteq M'$ velkosti $|M''| = |E| - |J|$ taká, že pre všetky $i \in M''$ platí $\sum_{j \in J} t_{ij} x_{ij} = T_i$.

Dôkaz: Pripomeňme si definíciu bázového riešenia (Definícia 2.2). Najprv požadujeme program v normálnom tvare $\max_{\mathbf{x}} \{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$, kde A má plnú hodnosť (t.j. neobsahuje lineárne závislé riadky). Program (31) upravíme tak, že zmeníme minimalizáciu na maximalizáciu a pre každé obmedzenie $\sum_{j \in J} t_{ij} x_{ij} \leq T_i$ zavedieme rezervnú premennú s_i a dostaneme $\sum_{j \in J} t_{ij} x_{ij} + s_i = T_i$. Získame tak maticu obmedzení rozmerov $(n + m') \times (|E| + m')$ v tvare

$$A = \left(\begin{array}{c|c} B & 0 \\ \hline C & I \end{array} \right)$$

kde $B \in \mathbb{R}^{n \times |E|}$ je matica obmedzení pre úlohy, $C \in \mathbb{R}^{m' \times |E|}$ je matica obmedzení pre procesory a I je identická matica $m' \times m'$ zahŕňajúca rezervné premenné s_i . Čitateľ sa ľahko presvedčí, že riadky matice A sú lineárne nezávislé (každý riadok v hornej časti obsahuje iné premenné).

Napríklad pre zadanie (30) dostávame program

$$\max_{\mathbf{x} \in \mathbb{R}^6} \{\mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$$

kde

$$\mathbf{c} = \begin{pmatrix} -3 \\ -9 \\ -3 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} x_{11} \\ x_{21} \\ x_{31} \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} \quad A = \left(\begin{array}{ccc|ccc} 1 & 1 & 1 & 0 & 0 & 0 \\ 30 & 0 & 0 & 1 & 0 & 0 \\ 0 & 10 & 0 & 0 & 1 & 0 \\ 0 & 0 & 30 & 0 & 0 & 1 \end{array} \right) \quad \mathbf{b} = \begin{pmatrix} 1 \\ T \\ T \\ T \\ T \end{pmatrix}$$

Podľa Definície 2.2, báza má veľkosť $n + m'$ a všetky nebázové zložky musia byť nulové. Pretože predpokladáme, že všetky $x_{ij} > 0$, v bázovom riešení musí byť $|E| + m' - (n + m') = |E| - n$ nulových hodnôt s_i . Každá nulová rezervná premenná s_i dáva jeden procesor, pre ktorý platí $\sum_{j \in J} t_{ij} x_{ij} + s_i = T_i$. \square

Dôkaz Lemy 3.19: Majme bázové riešenie programu (31), kde pre všetky i, j je $0 < x_{ij} < 1$ a navyše pre všetky $i \in M'$ je $\deg(i) \geq 2$. Pre každú úlohu $j \in J$ musí platiť $\deg(j) \geq 2$: pretože $\sum_{i \in M} x_{ij} = 1$, keby $\deg(j) = 1$, musela by nejaká hodnota $x_{ij} = 1$. Zoberme si množinu M'' z Lemy 3.20. Platí

$$|J| + |M''| = |E| = \frac{\sum_{j \in J} \deg(j) + \sum_{i \in M} \deg(i)}{2} \geq \frac{\sum_{j \in J} \deg(j) + \sum_{i \in M'} \deg(i)}{2} \stackrel{(\clubsuit)}{\geq} |J| + |M'| \geq |J| + |M''|$$

Nerovnosť (\clubsuit) platí preto, lebo pre $i \in M'$ je $\deg(i) \geq 2$ z predpokladu a $\deg(j) \geq 2$ sme ukázali pred chvíľou pre všetky $j \in J$. Pretože prvý a posledný člen v sérii nerovností sa rovnajú, musia platiť rovnosti všade. Pretože $M'' \subseteq M'$, platí $M' = M''$. Zároveň, pretože všetky vrcholy z J a M' sú stupňa aspoň 2, je $\deg(j) = 2$ pre všetky $j \in J$ a $\deg(i) = 2$ pre všetky $i \in M'$ jediná možnosť, ako uchovať v platnosti rovnosť v (\clubsuit). Aby mohla platiť aj predchádzajúca rovnosť, musí byť $\deg(i) = 0$ pre všetky $i \in M \setminus M'$.

Z toho ale vyplýva, že G sa skladá z izolovaných vrcholov a vrcholov stupňa 2, čiže je tvorený dizjunktnými cyklami. Keďže G je bipartitný, každý cyklus C má párnú dĺžku, a preto obsahuje rovnaký počet k procesorov aj úloh. Keďže pre každú úlohu j je $\sum_{i \in M'} x_{ij} = 1$, je $\sum_{(i,j) \in C} x_{ij} = k$ a nutne musí v C existovať procesor i , pre ktorý je $\sum_{j \in J} x_{ij} \geq 1$. \square

S charakterizáciou bázových riešení pomocou Lemy 3.19 sa nám začína črtať algoritmus riešenia. Keď v riešení programu (31) je nejaká premenná $x_{ij} = 1$, priradíme úlohu j procesoru i ; úloha j tým bude vybavená a procesoru i sa zmenší jeho termín T_i (preto sme v programe (31) chceli mať rôzne limity pre rôzne procesory). Ak je nejaká premenná $x_{ij} = 0$, vyhodíme z grafu príslušnú hranu, čím dostaneme menší graf (a teda menší problém). Ak je nejaký procesor $i \in M'$, pre ktorý $\deg(i) = 1$, nemusíme pri ňom uvažovať žiadnen termín: keďže je iba jedna úloha, ktorá mu potenciálne môže byť priradená (a na začiatku sme bez ujmy na všeobecnosti predpokladali, že každá hrana má čas nanajvýš T), i nikdy neprekročí termín. Posledná vec, čo sa môže po vyriešení programu (31) stat, je že máme v M' procesor i , ktorý má stupeň 2 a $\sum_{j \in J} x_{ij} \geq 1$. V tomto prípade tiež nemusíme uvažovať žiadnen termín: i má stupeň 2, takže najväčšia možná hodnota $\sum_{j \in J} x_{ij} \leq 2$. Keďže v našom riešení je $\sum_{j \in J} x_{ij} \geq 1$ a termín je stále splnený, aj bez akýchkoľvek obmedzení termín nemôže byť prekročený viac ako o dvojnásobok. V oboch prípadoch teda dostaneme problém, ktorý je menší v tom, že sa zmenšila množina M' . Podme teraz túto tému rozvíest detailnejšie. Algoritmus na riešenie problému GENERALIZED-ASSIGNMENT bude vyzerať nasledovne

-
- 1 $M' := M$, $\forall i : T_i := T$, $F := \emptyset$ (F je množina priradených hrán)
 2 kým $J \neq \emptyset$
 3 nech \mathbf{x} je optimálne riešenie programu (31),
 vykonaj jednu z nasledovných možností:
 4a ak existuje $x_{ij} = 0$,
 vyhod hranu (i, j) z G
 4b ak existuje $x_{ij} = 1$,
 $F := F \cup \{(i, j)\}$, $J := J \setminus \{j\}$, $T_i := T_i - t_{ij}$
 4c inak nech $i \in M'$ také, že $\deg(i) \leq 1$ alebo $\deg(i) = 2$ a $\sum_{j \in J} x_{ij} \geq 1$
 $M' := M' \setminus \{i\}$
-

Každá iterácia cyklu na riadku 2 zmenší hodnotu $|E| + |J| + |M'|$, preto algoritmus po polynomiálnom počte iterácií skončí. Lahko vidno, že po skončení je každá úloha priradená nejakému procesoru. Ostáva nám ukázať, že toto priradenie má optimálnu cenu a termín je prekročený najviac o dvojnásobok:

Veta 3.21. *Uvedený algoritmus vyrieší v polynomiálnom čase problém GENERALIZED-ASSIGNMENT. Nájdené riešenie má optimálnu cenu a každý procesor skončí najneskôr v čase $2T$.*

Dôkaz: Už sme nahliadli, že algoritmus v polynomiálnom čase vráti nejaké riešenie. Teraz ukážeme, že toto riešenie má optimálnu cenu. Nech c je cena optimálneho riešenia programu (31) pre $M' = M$ a $T_i = T$. Zjavne $c \leq OPT$. Indukciou ukážeme, že v každej iterácii cena už priradených úloh F , spolu s cenou riešenia aktuálneho programu (31) z riadku 3 je dokopy nanajvýš c . V prvej iterácii tvrdenie platí triviálne. Ak sa vykoná riadok 4a, nezmení sa ani cena F , ani cena optima (31). Ak sa vykoná riadok 4b, cena F stúpne o c_{ij} a cena optima (31) klesne aspoň o $c_{ij}x_{ij} = c_{ij}$. Ak sa vykoná riadok 4c, cena F sa nezmiení a cena optima (31) nestúpne.

Teraz ukážeme, že čas každého procesora je najviac $2T$. Označme $F_{i,\ell}$ čas, ktorý na procesore i indukujú úlohy priradené do F počas prvých $\ell - 1$ iterácií, a podobne $T_{i,\ell}$ hodnotu T_i na začiatku ℓ -tej iterácie. Zafixujme si procesor i . Počas prvých iterácií, kým $i \in M'$, ostáva v platnosti $T_{i,\ell} + F_{i,\ell} \leq T$. Uvažujme iteráciu ℓ , v ktorej sa procesor i vyhodí z M' v riadku 4c. To sa môže stať dvoma spôsobmi:

1. $\deg(i) \leq 1$. Nech j je jediná úloha, ktorá je spojená⁸ v ℓ -tej iterácii s i . Koľko času spotrebuje i vo finálnom riešení? Čas, ktorý už má priradený, t.j. $T_{i,\ell}$, plus možno t_{ij} , ak sa mu j -ta úloha priradí. Viac spotrebovať nemôže, lebo žiadna iná úloha s ním už nie je spojená. Preto výsledný čas je nanajvýš $F_{i,\ell} + t_{ij} \leq T_{i,\ell} + F_{i,\ell} + t_{ij} \leq 2T$; posledná nerovnosť vyplýva z toho, že pre všetky hrany je $t_{ij} \leq T$ a až po ℓ -tú iteráciu bol $i \in M'$, takže $T_{i,\ell} + F_{i,\ell} \leq T$.

2. $\deg(i) = 2$ a $\sum_{j \in J} x_{ij} \geq 1$ Nech j_1, j_2 sú jediné dve úlohy, ktoré sú spojené s i . Opäť sa pýtame, koľko času môže spotrebovať i vo výslednom riešení. Zjavne najviac $F_{i,\ell} + t_{ij_1} + t_{ij_2}$. Na začiatku ℓ -tej iterácie platilo $F_{i,\ell} + T_{i,\ell} \leq T$. Pri riešení programu (31) v ℓ -tej iterácii je $T_{i,\ell}$ limit pre procesor i , preto ak \mathbf{x} je riešenie z riadku 3, platí $t_{ij_1}x_{ij_1} + t_{ij_2}x_{ij_2} \leq T_{i,\ell}$. Preto platí $F_{i,\ell} \leq T - t_{ij_1}x_{ij_1} + t_{ij_2}x_{ij_2}$ a odtiaľ dostaneme, že i určite nespotrebuje viac času ako

$$\begin{aligned}
 T - t_{ij_1}x_{ij_1} + t_{ij_2}x_{ij_2} + t_{ij_1} + t_{ij_2} &\leq \\
 T + (1 - x_{ij_1})t_{ij_1} + (1 - x_{ij_2})t_{ij_2} &\leq \\
 T + (1 - x_{ij_1})T + (1 - x_{ij_2})T &\leq \\
 T(3 - x_{ij_1} - x_{ij_2}) &\leq 2T
 \end{aligned}$$

pričom posledná nerovnosť vyplýva z predpokladu, že $\sum_{j \in J} x_{ij} \geq 1$. □

⁸Podľa správnosti by sme mali aj graf G indexovať číslom iterácie ℓ , lebo sa G sa v priebehu výpočtu mení. Spoľahlame sa ale na to, že z kontextu je zjavné, o ktorý graf G sa jedná.

3.3 Randomizované zaokrúhľovanie

Budeme ďalej pokračovať v rovnakej schéme ako doteraz: máme riešiť optimalizačný problém. Zapišeme si ho ako celočíselný program. Vyriešime jeho relaxovanú verziu. Rozmýšlame, čo s tým. Ak sme napríklad v pôvodnom celočíselnom programe mali premenné $x_i \in \{0, 1\}$, ktoré sme si interpretovali ako indikátory, či je i -ta vec vybraná, v relaxovanom programe máme $0 \leq x_i \leq 1$, čo nás láka predstavovať si hodnoty x_i ako pravdepodobnosť, že je i -ta vec vybraná. V tejto časti ukážeme, ako túto intuícii využiť na navrhovanie algoritmov. Všeobecný postup bude taký, že zaokrúhľovací algoritmus nebude deterministický, ale pravdepodobnostný. V prvom kroku ukážeme, že v očakávanom prípade dostaneme dobré riešenie a v druhom kroku ukážeme, ako vyrobiť deterministický zaokrúhľovací algoritmus, ktorý vždy vráti riešenie rovnako dobré, ako randomizovaný algoritmus v očakávanom prípade (tento proces budeme volať *derandomizácia*). Ako vždy, aj teraz použijeme konkrétny príklad:

MAX-SAT

V časti o celočíselnom lineárnom programovaní sme spomínali rozhodovací problém SAT (Definícia 3.2). Uvažujme teraz jeho optimalizačnú verziu:

Definícia 3.22. Majme n logických premenných $x_1, \dots, x_n \in \{\text{true}, \text{false}\}$. Literál je premenná x_i alebo jej negácia \bar{x}_i . Klauzula je dizjunkcia literálov $C = l_1 \vee l_2 \vee \dots \vee l_{k_C}$. Formula v konjunktívnej normálnej forme (CNF) je konjunkcia klauzúl $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$. Problém MAX-SAT je optimalizačný problém, v ktorom je na vstupe daná formula F v CNF, pričom každá klauzula C_i má nezápornú cenu ω_i . Úlohou je nájsť priradenie pravdivostných hodnôt premenným x_1, \dots, x_n tak, aby súčet váh splnených klauzúl bol maximálny.

Ak je F splnitelná a všetky váhy sú jednotkové, optimum je m , takže keby sme vedeli riešiť MAX-SAT, vedeli by sme riešiť aj SAT. V ďalšom označme celkovú váhu všetkých klauzúl Z , t.j.

$$Z = \sum_{i=1}^m \omega_i.$$

Lahko vidno, že v každom prípade môžme dosiahnuť váhu $Z/2$: najprv nastavíme všetky premenné na 0. Ak sme dosiahli aspoň $Z/2$, všetko je v poriadku. Ak nie, nastavíme všetky premenné na 1. Každá klauzula, ktorá predtým nebola splnená, teraz splnená bude, takže máme garantované váhu aspoň $Z/2$. Skúsme teraz uvažovať, či vieme spraviť algoritmus s lepšou garanciou; ukážeme, ako garantovať $3/4$ celkovej váhy.

Začnime najprv s úplne jednoduchým randomizovaným algoritmom A1, ktorý vôbec lineárne programovanie nepoužíva, ale nastaví každú premennú na 1 nezávisle s pravdepodobnosťou $1/2$. Aká je váha splnených klauzúl v očakávanom prípade? Označme túto váhu W : W je náhodná premenná a zaujíma nás jej stredná hodnota $E[W]$. Nech X_i je indikátorová náhodná premenná, ktorá je 1, ak je C_i splnená a 0 inak. Platí

$$W = \sum_{i=1}^m \omega_i X_i$$

a preto z linearity strednej hodnoty

$$E[W] = E \left[\sum_{i=1}^m \omega_i X_i \right] = \sum_{i=1}^m \omega_i E[X_i] = \sum_{i=1}^m \omega_i \Pr[X_i = 1].$$

Pre klauzulu C_i označme $s(C_i)$ jej veľkosť, t.j. počet literálov v nej. Aká je pravdepodobnosť, že konkrétna klauzula C_i je splnená? Klauzula je splnená vtedy, keď je splnený aspoň jeden z jej literálov. Kedže každá premenná x_i je nastavená nezávisle⁹ s pravdepodobnosťou $1/2$, pravdepodobnosť, že C_i je nesplnená je $2^{-s(C_i)}$, a teda $\Pr[X_i = 1] = 1 - 2^{-s(C_i)}$. Predpokladajme teraz,

⁹zdôrazňujeme, že náhodné premenné X_i nie sú nezávislé

že každá klauzula má aspoň k literálov. Potom algoritmus A1 v očakávanom prípade získa váhu aspoň

$$\mathbb{E}[W] = \sum_{i=1}^m \omega_i \Pr[X_i = 1] = \sum_{i=1}^m \omega_i (1 - 2^{-s(C_i)}) \geq (1 - 2^{-k})Z. \quad (32)$$

Vo všeobecnom prípade, keď formula môže mať klauzuly veľkosti 1, sme si nijak nepomohli – stále máme garanciu $Z/2$. Pre väčšie hodnoty k však dostávame lepšie výsledky.

Derandomizácia

Z predchádzajúcej časti máme randomizovaný algoritmus, ktorý v očakávanom prípade splní klauzuly s váhou $(1 - 2^{-k})Z$, ak každá klauzula má aspoň k literálov. Otázka znie, ako zostrojiť deterministický algoritmus, ktorý by mal rovnakú garanciu, t.j. vždy dosiahol váhu aspoň $(1 - 2^{-k})Z$. Algoritmus A1 používa n náhodných bitov. V procese derandomizácie ho budeme postupne modifikovať tak, že prvých i bitov (postupne pre $i = 1, 2, \dots$) zafixujeme a zvyšné ponecháme náhodné. Dostaneme algoritmus používajúci $n - i$ náhodných bitov a chceme, aby jeho očakávaná hodnota neklesla. Keď zafixujeme všetkých n bitov, dostaneme deterministický algoritmus.

Ako zafixovať prvý bit? Ak nastavíme napr. $x_1 = 1$, pre očakávanú hodnotu platí

$$\mathbb{E}[W | x_1 = 1] = \sum_{i=1}^m \omega_i \Pr[X_i = 1 | x_1 = 1].$$

Zoberme si i -tu klauzulu. Aká je pravdepodobnosť, že C_i je splnená, ak $x_1 = 1$? Ak C_i obsahuje literál x_i , je pravdepodobnosť 1, ak obsahuje literál \bar{x}_i , je táto pravdepodobnosť $1 - 2^{1-s(C_i)}$ a inak ostala rovnaká $1 - 2^{-s(C_i)}$. Rovnako sa dá zrátať hodnota $\mathbb{E}[W | x_1 = 0]$. Pretože x_1 bolo nastavené na 1 s pravdepodobnosťou 1/2, je

$$\mathbb{E}[W] = \frac{1}{2} \mathbb{E}[W | x_1 = 0] + \frac{1}{2} \mathbb{E}[W | x_1 = 1],$$

a teda jedna z $\mathbb{E}[W | x_1 = 0]$, $\mathbb{E}[W | x_1 = 1]$ je aspoň tak veľká ako $\mathbb{E}[W]$. Dostali sme algoritmus, ktorý používa $n - 1$ náhodných bitov a má v očakávanom prípade cenu aspoň $\mathbb{E}[W]$.

Derandomizovaný algoritmus A1 det bude pracovať v n iteráciách, pričom v t -tej iterácii si bude pamätať konštanty c_1, \dots, c_{t-1} . Pre všetky C_i zrátať $\Pr[X_i = 1 | x_1 = c_1, \dots, x_{t-1} = c_{t-1}, x_t = 0]$ a $\Pr[X_i = 1 | x_1 = c_1, \dots, x_{t-1} = c_{t-1}, x_t = 1]$. Na základe toho toho zráta očakávané hodnoty $\mathbb{E}[W | x_1 = c_1, \dots, x_{t-1} = c_{t-1}, x_t = 0]$ a $\mathbb{E}[W | x_1 = c_1, \dots, x_{t-1} = c_{t-1}, x_t = 1]$ a podľa toho, ktorá je väčšia, nastaví c_t . Po n iteráciach budú hodnoty c_1, \dots, c_n tvoriť riešenie.

Rovnaký proces derandomizácie (nazývaný *metóda podmienených pravdepodobností*) sa dá použiť vždy, ak vieme algoritmicky zrátať očakávanú hodnotu pôvodného randomizovaného algoritmu pri zafixovaných prvých i bitoch, aj keď náhodné rozhodnutia môžu mať rôzne pravdepodobnosti:

Veta 3.23. Majme randomizovaný algoritmus \mathbf{A}_R riešiaci v polynomiálnom čase optimalizačný problém \mathcal{P} , pričom na vstupe \mathbf{x} urobí $R(\mathbf{x})$ náhodných rozhodnutí $r_1, r_2, \dots, r_{R(\mathbf{x})}$; rozhodnutia sú nezávislé binárne náhodné premenné s $\Pr[r_i = 1] = p_i$. Predpokladajme, že existuje polynomiálny algoritmus, ktorý pre ľubovoľné i a konštanty $c_1, \dots, c_i \in \{0, 1\}$ vyráta $\mathbb{E}[\mathbf{A}_R(\mathbf{x}) | b_1 = c_1, \dots, b_i = c_i]$. Potom existuje deterministický algoritmus, ktorý v polynomiálnom čase nájde riešenie aspoň tak dobré, ako $\mathbb{E}[\mathbf{A}_R(\mathbf{x})]$.

Vo všeobecnom prípade sa v jednej iterácii derandomizovaného algoritmu využije

$$\begin{aligned} \mathbb{E}[W | r_1 = c_1, \dots, r_{i-1} = c_{i-1}] &= p_i \mathbb{E}[W | r_1 = c_1, \dots, r_{i-1} = c_{i-1}, r_i = 1] + \\ &\quad (1 - p_i) \mathbb{E}[W | r_1 = c_1, \dots, r_{i-1} = c_{i-1}, r_i = 0], \end{aligned}$$

odkial vyplýva, že bez ohľadu na p_i je aspoň jedna z očakávaných hodnôt pre $r_i = 1, r_i = 0$ aspoň tak veľká ako pôvodná očakávaná hodnota.

Algoritmus pre krátke klauzuly

Vráťme sa k problému MAX-SAT. Keby každá klauzula mala aspoň dva literály, algoritmus A1det dosiahne aspoň $3/4$ celkovej váhy Z . Problémom ostávajú jednoliterálové klauzuly. Pre čitatela ználeho rozhodovacieho problému SAT to na prvý pohľad môže vyzerať iba ako kozmetický problém: ak máme rozhodnúť splniteľnosť, jednoliterálové klauzuly nám iba pomáhajú, pretože určujú hodnotu, ktorú príslušná premenná musí mať v každom splňujúcom ohodnení. Ked ale formula splniteľná nie je a chceme splniť čo najväčšiu váhu klauzúl, táto dobrá vlastnosť sa stráca a jednoliterálové klauzuly môžu spôsobovať problémy.

Tu príde do hry lineárne programovanie. Skúsme zapísat problém MAX-SAT ako celočíselný lineárny program. Celkom prirodzene majme pre každú premennú x_i zo vstupnej formuly premennú $p_i \in \{0, 1\}$. Cielom je maximalizovať váhu splnených klauzúl, a preto si spravme pre každú klauzulu C_i premennú $z_i \in \{0, 1\}$, ktorá bude určovať, či je C_i splnená. Cielom je maximalizovať $\sum_{i=1}^m \omega_i z_i$ a obmedzenia musia zabezpečiť, aby z_i bolo 1 iba vtedy, keď je C_i splnená, t.j. ak žiadenny literál z C_i nie je splnený, z_i musí byť 0. Označme C_i^+ indexy premenných, ktoré sa v klauzule C_i vyskytujú v pozitívnej forme a C_i^- indexy tých premenných, ktoré sú v C_i negované. Dostávame nasledovný program:

$$\begin{aligned} & \text{maximalizovať } \sum_{i=1}^m \omega_i z_i \\ \text{pri obmedzeniach } & \sum_{j \in C_i^+} p_j + \sum_{j \in C_i^-} (1 - p_j) \geq z_i \quad \forall i = 1, \dots, m \\ & z_i, p_i \in \mathbb{Z} \end{aligned} \tag{33}$$

a jeho relaxovanú verziu

$$\begin{aligned} & \text{maximalizovať } \sum_{i=1}^m \omega_i z_i \\ \text{pri obmedzeniach } & \sum_{j \in C_i^+} p_j + \sum_{j \in C_i^-} (1 - p_j) \geq z_i \quad \forall i = 1, \dots, m \\ & z_i, p_i \geq 0 \\ & z_i, p_i \leq 1 \end{aligned} \tag{34}$$

Konečne sa dostávame k jadru tejto kapitoly – randomizovanému zaokrúhľovaniu. Algoritmus A2 vyrieší program (34), čím dostane optimálne hodnoty p_i^* , z_i^* a každú premennú x_i nastaví na 1 nezávisle s pravdepodobnosťou p_i^* . Aká je očakávaná hodnota algoritmu A2? Rovnako ako pri algoritme A1 označme W celkovú cenu algoritmu a X_i indikátor, či je splnená i -ta klauzula a odhadujme

$$E[W] = \sum_{i=1}^m \omega_i \Pr[X_i = 1].$$

Klauzula C_i je nesplnená, ak nie je splnený žiadnen z jej literálov. Pozitívny literál x_j je nesplnený s pravdepodobnosťou $1 - p_j^*$ a negatívny \bar{x}_j s pravdepodobnosťou p_j^* , takže

$$\Pr[X_i = 1] = 1 - \prod_{j \in C_i^+} (1 - p_j^*) \cdot \prod_{j \in C_i^-} p_j^* \tag{35}$$

$$\geq 1 - \left(\frac{\sum_{j \in C_i^+} (1 - p_j^*) + \sum_{j \in C_i^-} p_j^*}{s(C_i)} \right)^{s(C_i)} \tag{36}$$

$$\begin{aligned} & = 1 - \left(1 - \frac{\sum_{j \in C_i^+} p_j^* + \sum_{j \in C_i^-} (1 - p_j^*)}{s(C_i)} \right)^{s(C_i)} \\ & \geq 1 - \left(1 - \frac{z_i^*}{s(C_i)} \right)^{s(C_i)} \end{aligned} \tag{37}$$

kde nerovnosť (36) je aplikáciou aritmeticko-geometrickej nerovnosti

$$\frac{a_1 + \dots + a_n}{n} \geq \sqrt[n]{a_1 \cdots a_n}$$

a nerovnosť (37) vyplýva z obmedzení programu (34). Našim ďalším bezprostredným cieľom bude vyjadriť odhad na $\Pr[X_i = 1]$ z riadku (37) v lineárnom tvare, t.j. $\Pr[X_i = 1] \geq \beta z_i^*$ pre nejaké β , ktoré môže závisieť od C_i , ale nezávisí od z_i^* . Označme si

$$g_k(z) := 1 - \left(1 - \frac{z}{k}\right)^k$$

funkciu premennej z s parametrom $k \geq 1$. Máme $g_k(0) = 0$ a $g_k(k) = 1$. Priamočiaro zrátame

$$\begin{aligned} g'_k(z) &= \left(1 - \frac{z}{k}\right)^{k-1} \\ g''_k(z) &= -\frac{k-1}{k-z} \left(1 - \frac{z}{k}\right)^{k-2}, \end{aligned}$$

takže vidíme, že $g_k(z)$ je na intervale $[0, k]$ rastúca ($g'_k(z) > 0$) a konkávna ($g''_k(z) < 0$). Preto celý graf na intervale $[0, 1]$ leží nad priamkou prechádzajúcou bodmi $[0, 0]$ a $[1, g_k(1)]$. Preto pre $z \in [0, 1]$ platí $g_k(z) \geq g_k(1)z$. Keď sa vrátíme k nerovnosti (37), dostávame

$$\Pr[X_i = 1] \geq g_{s(C_i)}(1)z_i^*.$$

Označme

$$\beta(k) := g_k(1) = 1 - \left(1 - \frac{1}{k}\right)^k$$

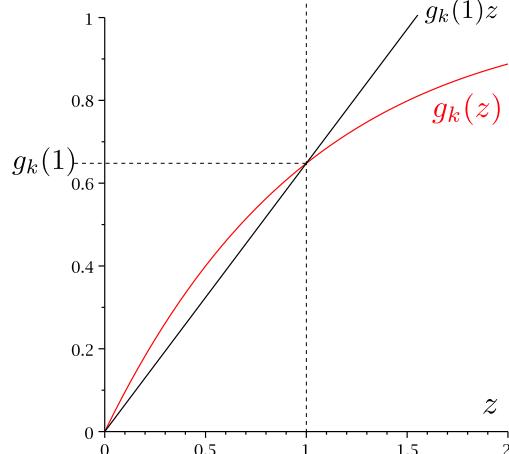
funkciu premennej k . Pre očakávanú cenu algoritmu A2 máme

$$\mathbb{E}[W] \geq \sum_{i=1}^m \omega_i \beta(s(C_i)) z_i^*. \quad (38)$$

S rastúcim k hodnota $\beta(k)$ klesá, preto ak všetky klauzuly majú najviac k literálov, máme

$$\mathbb{E}[W] \geq \beta(k) \sum_{i=1}^m \omega_i z_i^*.$$

Môžeme použiť vetu 3.23 a vyrobiť deterministický algoritmus **A2det**; príslušné podmienené pravdepodobnosti sa budú rátať analogicky, podľa riadku (35).



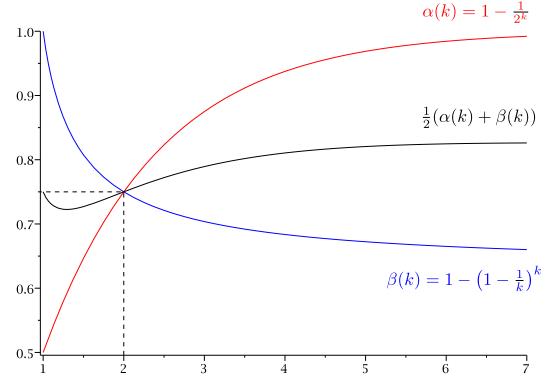
Výsledný algoritmus s garanciou 3/4

Máme teraz dva algoritmy: **A1det**, ktorý funguje dobre na inštanciach s dlhými klauzulami a **A2det**, ktorý funguje dobre na inštanciach s krátkymi klauzulami. Čo môžeme robiť na inštanciach, ktoré majú dlhé aj krátke klauzuly? Urobíme prirodzený vec: spustíme obidva algoritmy a z výsledkov vyberieme ten lepší. Z (32) a (38) a z vety 3.23 dostávame pre ceny algoritmov **A1det** a **A2det** na formule F :

$$\mathbf{A1det}(F) \geq \sum_{i=1}^m \omega_i \left(1 - \frac{1}{2^{s(C_i)}}\right) \quad \mathbf{A2det}(F) \geq \sum_{i=1}^m \omega_i \beta(s(C_i)) z_i^*$$

Označíme $\alpha(k) := 1 - 2^{-k}$ a lepší z nich (t.j. maximum) odhadneme zdola priemerom:

$$\begin{aligned}
& \max\{\text{A1det}(F), \text{A2det}(F)\} \\
& \geq \frac{1}{2}(\text{A1det}(F) + \text{A2det}(F)) \\
& \geq \frac{1}{2} \sum_{i=1}^m \omega_i (\beta(s(C_i)) z_i^* + \alpha(s(C_i))) \\
& \geq \sum_{i=1}^m \omega_i z_i^* \left(\frac{\alpha(s(C_i)) + \beta(s(C_i))}{2} \right)
\end{aligned}$$



kde posledná nerovnosť vyplýva z toho, že $z_i^* \leq 1$. Pozrime sa teraz na i -tu klauzulu; nech $s(C_i) = k$. Ako vyzerá priemer $\frac{1}{2}(\alpha(k) + \beta(k))$? Pre $k \in \{1, 2\}$ je $\frac{1}{2}(\alpha(k) + \beta(k)) = \frac{3}{4}$. Pre $k \geq 2$ je to rastúca funkcia, takže môžme uzavrieť, že

$$\max\{\text{A1det}(F), \text{A2det}(F)\} \geq \frac{3}{4} \sum_{i=1}^m \omega_i z_i^* \geq \frac{3}{4} OPT.$$

4

Dualita v lineárnom programovaní

4.1 Čo je dualita

V predchádzajúcich častiach sme sa snažili ukázať, že lineárne programovanie je silný a flexibilný nástroj pre riešenie optimalizačných problémov. Teraz sa zoznámime s ďalšou jeho podstatnou vlastnosťou, ktorá sa bude dať tiež mnohostranne využiť pri návrhu algoritmov. Táto vlastnosťou je tzv. *dualita*. Predpokladajme, že Róbert chce nájsť minimum funkcie

$$f(x_1, x_2, x_3) := 10x_1 + 3x_2 + 5x_3$$

spomedzi takých hodnôt x_1, x_2, x_3 , ktoré spĺňajú

$$6x_1 + x_2 - x_3 \geq 2 \quad (39)$$

$$2x_1 + 2x_2 + 6x_3 \geq 8 \quad (40)$$

$$6x_1 + 3x_2 + 5x_3 = 30$$

$$x_1, x_2, x_3 \geq 0$$

Angela vidí, že je to lineárny program a ľahko nájde minimum. Otázka je, ako má o tom presvedčiť Róberta, ktorý lineárne programovanie nepozná. O hornom odhade ho presvedčí ľahko: ukáže mu ako *svedka* nejaké konkrétné riešenie. Angela môže napríklad vybrať vektor $x_1 = 2, x_2 = 6, x_3 = 0$ a Róbert z toho ľahko zistí, že hľadané minimum je nanajvýš 38. Ako ale nájsť svedka pre odhad z opačnej strany? Angela môže napríklad skúsiť takýto argument: "Pozrime sa na riadok (42). Akékoľvek nezáporné čísla x_1, x_2, x_3 zvolíme, vždy bude platiť $6x_1 \leq 10x_1, x_2 \leq 3x_2$ a $-x_3 \leq 5x_3$. Preto $6x_1 + x_2 - x_3 \leq f(x_1, x_2, x_3)$. Lenže podľa (42) je $6x_1 + x_2 - x_3 \geq 2$, takže každá prípustná hodnota (a teda aj minimum) funkcie f je aspoň 2". Angela môže pokračovať napríklad tak, že scítá riadky (42) a (43) a použije rovnaký argument. Môže aj každý riadok prenásobiť nejakým číslom (v riadkoch (42) a (43) to musí byť nezáporné číslo) a scítať ich dokopy; ak platí, že výsledná lineárna kombinácia ľavých strán obmedzení je po zložkách menšia ako funkcia f , lineárna kombinácia pravých strán tvorí dolný odhad minima. Aby poskytla čo najlepší argument, Angela vlastne rieši lineárny program: nájsť maximum funkcie

$$g(y_1, y_2, y_3) := 2y_1 + 8y_2 + 30y_3$$

spomedzi takých hodnôt y_1, y_2, y_3 , ktoré spĺňajú

$$\begin{aligned} 6y_1 + 2y_2 + 6y_3 &\leq 10 \\ y_1 + 2y_2 + 3y_3 &\leq 3 \\ -y_1 + 6y_2 + 5y_3 &\leq 5 \\ y_1, y_2 &\geq 0 \end{aligned}$$

V tomto prípade je na jednej strane svedok $x_1 = 0, x_2 = 5, x_3 = 3$, že hľadané minimum f je najviac 30, na druhej strane svedok $y_1 = y_2 = 0, y_3 = 1$ ukazuje, že 30 je skutočná minimálna hodnota f . Je to náhoda, že sa podarilo touto metódou nájsť tesný odhad?

Pozrime sa všeobecnejšie na to, čo sa dialo. Majme nasledovný lineárny program, ktorý nazveme *primárny*

$$\begin{array}{lll} \text{minimalizovať} & c_1 x_1 & + c_2 x_2 + \cdots + c_n x_n \\ \text{pri obmedzeniach} & a_{1,1} x_1 & + a_{1,2} x_2 + \cdots + a_{1,n} x_n \geq b_1 \\ & a_{2,1} x_1 & + a_{2,2} x_2 + \cdots + a_{2,n} x_n \geq b_2 \\ & \vdots & \vdots \ddots \vdots \vdots \\ & a_{m,1} x_1 & + a_{m,2} x_2 + \cdots + a_{m,n} x_n \geq b_m \\ & x_1, \dots, x_n & \geq 0 \end{array}$$

alebo skrátene

$$(P) : \min_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0} \}.$$

Dolný odhad minima (P) hľadáme ako vhodnú lineárnu kombináciu obmedzení: j -te obmedzenie prenásobíme na oboch stranách koeficientom $y_j \geq 0$ a sčítame. Chceme nájsť koeficienty y_j tak, aby ľavá strana bola menšia ako minimalizovaná funkcia

$$\sum_{i=1}^n c_i x_i \geq \sum_{j=1}^m y_j (a_{j,1}x_1 + a_{j,2}x_2 + \dots + a_{j,n}x_n) \geq \sum_{j=1}^m y_j \cdot b_j.$$

Kedže $x_i \geq 0$, prvá nerovnosť platí, ak platia nerovnosti v koeficientoch pri každom x_i . Najlepší dolný odhad, aký touto metódou môžeme získať, je preto riešením *duálneho* lineárneho programu

$$\begin{array}{lll} \text{maximalizovať} & b_1 y_1 & +b_2 y_2 & +\dots & +b_m y_m \\ \text{pri obmedzeniach} & a_{1,1} y_1 & +a_{2,1} y_2 & +\dots & +a_{m,1} y_m & \leq c_1 \\ & a_{1,2} y_1 & +a_{2,2} y_2 & +\dots & +a_{m,2} y_m & \leq c_2 \\ & \vdots & \vdots & \ddots & \vdots & \vdots \\ & a_{1,n} y_1 & +a_{2,n} y_2 & +\dots & +a_{m,n} y_m & \leq c_n \\ & y_1, \dots, y_m & & & & \geq 0 \end{array}$$

alebo skrátene

$$(D) : \max_{\mathbf{y} \in \mathbb{R}^m} \{ \mathbf{b}^\top \mathbf{y} \mid A^\top \mathbf{y} \leq \mathbf{c}, \mathbf{y} \geq \mathbf{0} \}.$$

Z úvah, ktoré sme robili, by mala byť zrejmá nasledovná veta, ktorá sa zvykne volať aj *slabá veta o dualite*:

Veta 4.1. Ak \mathbf{x} je ľubovoľné prípustné riešenie primárnej úlohy (P) a \mathbf{y} je ľubovoľné prípustné riešenie duálnej úlohy (D) , potom platí

$$\mathbf{c}^\top \mathbf{x} \geq \mathbf{b}^\top \mathbf{y}$$

Dôkaz:

$$\mathbf{c}^\top \mathbf{x} \geq (A^\top \mathbf{y})^\top \mathbf{x} = \mathbf{y}^\top A \mathbf{x} \geq \mathbf{y}^\top \mathbf{b} \quad (41)$$

Prvá nerovnosť vyplýva z podmienok duálnej úlohy, pretože $\mathbf{c} \geq A^\top \mathbf{y}$ a druhá nerovnosť z podmienok primárnej úlohy, pretože $A\mathbf{x} \geq \mathbf{b}$. \square

Veta 4.1 hovorí len to, ako sme duálny program konštruovali: všetky prípustné riešenia (D) sú dolné odhady minima (P) . V našom príklade však navyše platilo, že optimálne riešenia primárnej a duálnej úlohy sa rovnali. Našim cieľom bude ukázať, že to nebola náhoda. Ešte predtým ale niekoľko poznámok o konštrukcii duálnej úlohy. V našom príklade sme vychádzali z minimalizačnej úlohy a hľadali sme najväčší dolný odhad minima, preto duálna úloha bola maximalizačná. Všetky naše úvahy ale boli z tohto hľadiska symetrické: keby sme vychádzali z maximalizačnej úlohy, rovnakým spôsobom by sme hľadali najmenší horný odhad maxima a dostali by sme duálnu minimalizačnú úlohu. Čitateľ sa lahko presvedčí, že v tomto všeobecnejšom chápání duality je pojem primárnej a duálnej úlohy symetrický: ak by sme za primárnu úlohu zobraли program (D) a konštruovali duálnu úlohu, dostali by sme (P) . Vo všeobecnosti teda platí, že duálna úloha k duálnej úlohe je primárna úloha.

Pozrime sa ešte na tvar obmedzení. V motivačnom príklade sme mali v primárnej úlohe obmedzenia v tvare nerovnosti (\geq) aj rovnosti. V prvom prípade museli byť príslušné multiplikátory y_i nezáporné, v prípade rovnosti mohli byť ľubovoľné. Zároveň všetky premenné x_i v primárnej úlohe boli nezáporné, a preto sme potrebovali, aby každé obmedzenie duálnej úlohy bolo v tvare nerovnosti. Čo by sa stalo, ak by niektorá premenná, napríklad x_1 mohla byť aj záporná? Podobne, ako keď sme odvodzovali normálny tvar lineárneho programu, mohli by sme nahradíť $x_1 = z_1 - z_2$, $z_1, z_2 \geq 0$ a dostali by sme primárnu úlohu minimalizovať

$$f(z_1, z_2, x_2, x_3) := 10z_1 - 10z_2 + 3x_2 + 5x_3$$

spomedzi takých hodnôt z_1, z_2, x_2, x_3 , ktoré spĺňajú

$$6z_1 - 6z_2 + x_2 - x_3 \geq 2 \quad (42)$$

$$2z_1 - 2z_2 + 2x_2 + 6x_3 \geq 8 \quad (43)$$

$$6z_1 - 6z_2 + 3x_2 + 5x_3 = 30$$

$$z_1, z_2, x_2, x_3 \geq 0$$

Príslušná duálna úloha by bola nájsť maximum funkcie

$$g(y_1, y_2, y_3) := 2y_1 + 8y_2 + 30y_3$$

spomedzi takých hodnôt y_1, y_2, y_3 , ktoré spĺňajú

$$6y_1 + 2y_2 + 6y_3 \leq 10$$

$$-6y_1 - 2y_2 - 6y_3 \leq -10$$

$$y_1 + 2y_2 + 3y_3 \leq 3$$

$$-y_1 + 6y_2 + 5y_3 \leq 5$$

$$y_1, y_2 \geq 0$$

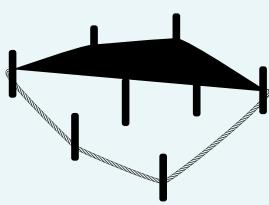
pričom prvé dve nerovnosti môžeme zlúčiť do jednej rovnosti $6y_1 + 2y_2 + 6y_3 = 10$. Ked zhrnieme naše doterajšie úvahy, konštrukciu duálnej úlohy môžeme opísť nasledovným receptom:

primárna úloha	duálna úloha
minimalizovať	$\mathbf{c}^T \mathbf{x}$
i -te obmedzenie tvaru	i -ta premenná
i -te obmedzenie tvaru	i -ta premenná
j -ta premenná	j -te obmedzenie tvaru
j -ta premenná	j -te obmedzenie tvaru

Skôr, ako pokročíme v našich úvahách, urobme malú odbočku:

Malá odbočka ku konvexným obalam

Čitateľ sa už možno stretol s konvexným obalom v dvoch rozmeroch: pre dané body v rovine je ich konvexný obal najmenší konvexný mnogouholník, ktorý ich všetky obsahuje.



Ak si body predstavíme ako kolíky, ktoré omotáme špagátom, tak kolíky, na ktorých sa špagát zachytí, tvoria vrcholy konvexného obalu. Takáto predstava ďalej celkom dobre zafunguje v troch rozmeroch, kde ako keby sme balili body do baliaceho papiera. V n rozmeroch a s $n - 1$ -rozmerným papierom je to ale komplikovanejšie; nie všetky vlastnosti, na ktoré sme z 2 rozmerov zvyknutí, platia aj v n rozmeroch, a preto si treba v úvahách dávať obzvlášť pozor na argumenty typu "je jasné, že...".

Pripomienieme, že konvexné teleso \mathcal{T} je také, že pre každé dva body $\mathbf{x}, \mathbf{y} \in \mathcal{T}$ je celá úsečka medzi nimi v \mathcal{T} , t.j. všetky body tvaru $t\mathbf{x} + (1-t)\mathbf{y}$ pre $0 \leq t \leq 1$ sú v \mathcal{T} . Definujme konvexný obal takto:

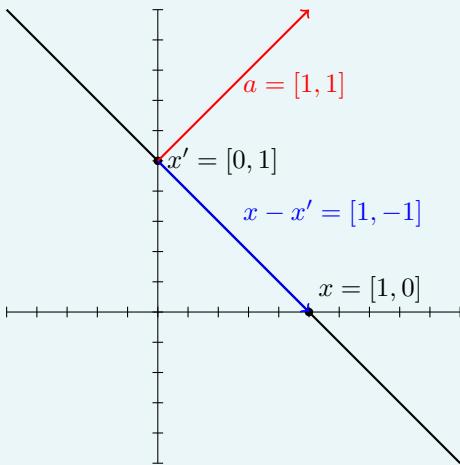
Definícia 4.2. Konvexný obal n bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$ je prienik všetkých konvexných telies, ktoré obsahujú body $\mathbf{a}_1, \dots, \mathbf{a}_n$.

Konvexných telies, ktoré obsahujú dané body je sice nespočítateľne veľa, ale ich prienik je dobre definovaný, takže nás to netrápi. Navyše, prienik ľubovoľných konvexných telies je zjavne opäť konvexné telo, takže naša definícia je dobrá v tom, že konvexný obal je, tak ako sa patrí, konvexný. Bude sa nám hodí aj nasledovná charakterizácia:

Lema 4.3. Konvexný obal n bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$ je tvorený bodmi konvexnými kombináciami bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$, t.j. bodmi $z_1\mathbf{a}_1 + \dots + z_n\mathbf{a}_n$, kde $z_1, \dots, z_n \in \mathbb{R}^+$ a $z_1 + \dots + z_n = 1$.

Dôkaz: Označme K množinu všetkých konvexných kobniacií bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$. Lahko sa overí, že K je konvexné telo: pre konvexné kombinácie $\sum z_i\mathbf{a}_i$ a $\sum z'_i\mathbf{a}_i$ je aj $t\sum z_i\mathbf{a}_i + (1-t)\sum z'_i\mathbf{a}_i = \sum(tz_i + (1-t)z'_i)\mathbf{a}_i$ konvexná kombinácia, a teda patrí do K . Pretože K je konvexné telo a obsahuje body $\mathbf{a}_1, \dots, \mathbf{a}_n$, z definície K je nadmnožina konvexného obalu. Na dôkaz lemy nám stačí ukázať, že každá konvexná kombinácia patrí do konvexného obalu.

Dokážeme to indukciou na počet bodov n . Pre $n = 1$ je to jasné priamo z definície konvexného obalu, pre $n = 2$ sú konvexné kombinácie $z_1\mathbf{a}_1 + (1-z_1)\mathbf{a}_2$ a tieto z definície konvexnosti ležia v každom konvexnom telese, ktoré obsahuje \mathbf{a}_1 a \mathbf{a}_2 (a teda ležia aj v ich prieniku). Nech teraz $n \geq 3$ a majme konvexnú kombináciu $\mathbf{x} = \sum z_i\mathbf{a}_i$. Ak $z_n = 1$, $\mathbf{x} = \mathbf{a}_n$ a z definície \mathbf{x} je v konvexnom obale. Nech teda $z_n < 1$. Označme $z'_i := \frac{z_i}{1-z_n}$ a zoberme bod $\mathbf{x}' = \sum z'_i\mathbf{a}_i$. \mathbf{x}' je konvexná kombinácia bodov $\mathbf{a}_1, \dots, \mathbf{a}_{n-1}$, a preto podľa indukčného predpokladu každé konvexné telo, ktoré obsahuje body $\mathbf{a}_1, \dots, \mathbf{a}_{n-1}$, obsahuje aj \mathbf{x}' . Zároveň $\mathbf{x} = (1-z_n)\mathbf{x}' + z_n\mathbf{a}_n$, a preto každé konvexné telo, ktoré obsahuje \mathbf{x}' a \mathbf{a}_n , obsahuje aj \mathbf{x} . \square



Ešte pripomeňme, že ak sme v m -rozmernom priestore, nadrovina \mathcal{H} je množina bodov \mathbf{x} spĺňajúcich rovnosť $a_1x_1 + \dots + a_mx_m = 1$ (jednotka na pravej strane je takmer všeobecná: ľubovoľná nadrovina, ktorá neprechádza bodom $[0, \dots, 0]$ sa dá normovaním upraviť na tento tvar). Ak označíme $\langle \cdot, \cdot \rangle$ skalárny súčin, je \mathcal{H} definovaná ako $\langle \mathbf{x}, \mathbf{a} \rangle = 1$. Body, pre ktoré $\langle \mathbf{x}, \mathbf{a} \rangle < 1$ sú pod \mathcal{H} , a body $\langle \mathbf{x}, \mathbf{a} \rangle > 1$ nad \mathcal{H} . Vektor \mathbf{a} je *normálkový vektor* \mathcal{H} a pre ľubovoľné dva body \mathbf{x}, \mathbf{x}' z \mathcal{H} platí,

$$\langle \mathbf{x} - \mathbf{x}', \mathbf{a} \rangle = \sum_{i=1}^m (x_i - x'_i)a_i = \langle \mathbf{x}, \mathbf{a} \rangle - \langle \mathbf{x}', \mathbf{a} \rangle = 0.$$

V dvoch rozmeroch je nadrovinou priamka, tá na obrázku vľavo je daná rovnicou $x_1 + x_2 = 1$.

Naše úvahy o konvexných obaloch zavŕšíme nasledovným pozorovaním, ktoré je pre dva a tri rozmery úplne zjavné:

Lema 4.4. Majme body $\mathbf{x}, \mathbf{a}_1, \dots, \mathbf{a}_n$ v m -rozmernom priestore. Nech K je konvexný obal $\mathbf{a}_1, \dots, \mathbf{a}_n$. Potom platí:

- Ak \mathbf{x} je vo vnútri^a K , potom neexistuje nadrovina \mathcal{H} , ktorá prechádza cez \mathbf{x} a všetky body z K ležia na jednej strane \mathcal{H} .
- Ak $\mathbf{x} \notin K$, potom existuje nadrovina \mathcal{H} , ktorá prechádza cez \mathbf{x} a všetky body z K ležia na jednej strane \mathcal{H} .

Dôkaz: Prvá časť je ľahká: nech \mathbf{x} leží vo vnútri K nech existuje taká nadrovina \mathcal{H} , že všetky body z K sú na jednej strane \mathcal{H} . Pretože \mathbf{x} je vo vnútri K , existuje nejaký bod $\mathbf{x}' \in K$, ktorý leží na opačnej strane \mathcal{H} ako všetky body z K , špeciálne ako body \mathbf{a}_i . Lenže polpriestor ohraničený \mathcal{H} , ktorý obsahuje všetky \mathbf{a}_i je konvexné telo, a preto z definície konvexného obalu $\mathbf{x}' \notin K$.

Ideme dokázať druhú časť. Majme fixovaný bod \mathbf{x} a pre body $\mathbf{y} \in K$ uvažujme euklidovskú vzdialenosť $f(\mathbf{y}) := \|\mathbf{y} - \mathbf{x}\|$. Pretože K je kompaktná množina a f je spojité funkcia nadobúdajúca kladné hodnoty, existuje bod $\mathbf{z} \in K$, na ktorom f nadobúda minimum. Označme

$$\mathbf{t} = \mathbf{x} - \mathbf{z} \quad \kappa = \langle \mathbf{t}, \mathbf{x} \rangle$$

Nech \mathcal{H} je nadrovina tvorená bodmi \mathbf{y} , pre ktoré $\langle \mathbf{t}, \mathbf{y} \rangle = \kappa$. Zjavne $\mathbf{x} \in \mathcal{H}$. Platí

$$0 \leq \sum_{i=1}^m (x_i - z_i)^2 = \langle \mathbf{x}, \mathbf{x} \rangle - 2 \langle \mathbf{x}, \mathbf{z} \rangle - \langle \mathbf{z}, \mathbf{z} \rangle$$

a preto

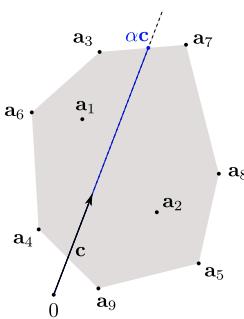
$$\begin{aligned} \langle \mathbf{t}, \mathbf{z} \rangle &= \langle \mathbf{x} - \mathbf{z}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle - \langle \mathbf{z}, \mathbf{z} \rangle \leq \\ &\leq \langle \mathbf{x}, \mathbf{x} \rangle - \langle \mathbf{x}, \mathbf{z} \rangle = \langle \mathbf{x} - \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{t}, \mathbf{x} \rangle = \kappa \end{aligned}$$

Teraz stačí ukázať, že pre všetky body $\mathbf{y} \in K$ tiež platí $\langle \mathbf{t}, \mathbf{y} \rangle \leq \kappa$, a teda ležia na rovnakej strane \mathcal{H} ako \mathbf{z} .

^at.j. K obsahuje \mathbf{x} aj nejaké jeho malé okolie

Majme teda $\mathbf{z} \in K$. Keďže \mathbf{z} minimalizuje vzdialenosť od \mathbf{x} a K je konvexné telo, pre všetky $0 \leq \lambda \leq 1$ platí $\|\mathbf{z} - \mathbf{x}\|^2 \leq \|(1 - \lambda)\mathbf{z} + \lambda\mathbf{y} - \mathbf{x}\|^2 = \|(1 - \lambda)(\mathbf{z} - \mathbf{x}) + \lambda(\mathbf{y} - \mathbf{x})\|^2$ a odtiaľ po priamočiarych úpravách $0 \leq \lambda(\lambda - 2)\|\mathbf{z} - \mathbf{x}\|^2 + 2\lambda(1 - \lambda) \langle \mathbf{z} - \mathbf{x}, \mathbf{y} - \mathbf{x} \rangle + \lambda^2 \|\mathbf{y} - \mathbf{x}\|^2$. Pretože $\lambda \geq 0$, máme $0 \leq (\lambda - 2)\|\mathbf{z} - \mathbf{x}\|^2 + 2(1 - \lambda) \langle \mathbf{z} - \mathbf{x}, \mathbf{y} - \mathbf{x} \rangle + \lambda \|\mathbf{y} - \mathbf{x}\|^2$ a odtiaľ v limite pre $\lambda \mapsto 0$ máme $0 \leq \langle \mathbf{z} - \mathbf{x}, \mathbf{y} - \mathbf{x} \rangle - \|\mathbf{z} - \mathbf{x}\|^2 = \langle \mathbf{z} - \mathbf{x}, \mathbf{y} - \mathbf{x} \rangle - \langle \mathbf{z} - \mathbf{x}, \mathbf{z} - \mathbf{x} \rangle = \langle \mathbf{z} - \mathbf{x}, \mathbf{y} \rangle - \langle \mathbf{z} - \mathbf{x}, \mathbf{z} \rangle = \langle \mathbf{t}, \mathbf{z} \rangle - \langle \mathbf{t}, \mathbf{y} \rangle$. Preto $\langle \mathbf{t}, \mathbf{y} \rangle \leq \langle \mathbf{t}, \mathbf{z} \rangle \leq \kappa$, čo sme chceli dokázať. \square

Vráťme sa teraz k nášmu cieľu – ukázať, že rovnaká hodnota optimálneho riešenia v primárnej aj duálnej úlohe nie je náhoda. Pozrime sa na nasledovný príklad: Majme m -rozmerný priestor \mathbb{R}^m a v ňom n bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$ (t.j. bod \mathbf{a}_i má súradnice $(a_{i1}, a_{i2}, \dots, a_{im})$) a vektor \mathbf{c} . Cieľom je nájsť čo najväčšie číslo $\alpha \in \mathbb{R}^+$ tak, že bod $\alpha\mathbf{c}$ leží v konvexnom obale K bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$.



Predpokladajme pre jednoduchosť, že úloha má riešenie, t.j. polpriamka generovaná vektorom \mathbf{c} pretína K . Z konvexitu vyplýva, že prienik K a polpriamky je úsečka. Optimálne riešenie je preto bod $\alpha\mathbf{c}$, v ktorom polpriamka opúšta K . Podľa Lemy 4.3 sa body konvexného obalu sa dajú vyjadriť ako konvexná kombinácia bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$, t.j. nás hľadaný bod $\alpha\mathbf{c}$ sa musí dať napísať v tvare $z_1\mathbf{a}_1 + \dots + z_n\mathbf{a}_n$ pre nejaké $z_1, \dots, z_n \geq 0$, kde $\sum_{i=1}^n z_i = 1$. Pre lubovoľný prípustný bod $\alpha\mathbf{c}$ si označme $y_j = \frac{z_j}{\alpha}$, potom platí

$$y_j \geq 0 \quad \sum_{j=1}^n y_j = \frac{1}{\alpha} \quad \alpha \mathbf{c} = \alpha \sum_{j=1}^n \mathbf{a}_j y_j.$$

Nájsť bod, pre ktorý je α maximálne, znamená nájsť bod, pre ktorý je $1/\alpha$ minimálne. Snažíme sa teda minimalizovať hodnotu $\sum_{j=1}^n y_j$ spomedzi takých $y_1, \dots, y_n \geq 0$, pre ktoré $\sum_{j=1}^n \mathbf{a}_j y_j = \mathbf{c}$; stačí si uvedomiť, že z daných hodnôt \mathbf{y} vieme zrekonštruovať \mathbf{z} a α . Našu úlohu teda vieme zapísť ako úlohu lineárneho programovania:

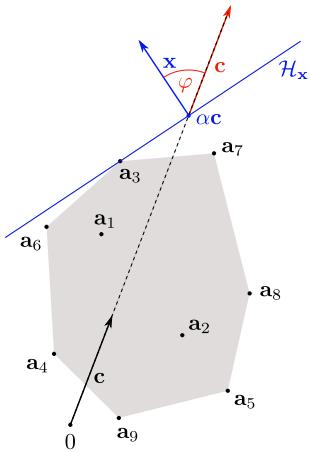
$$\begin{array}{lll} \text{minimalizovať} & y_1 & + y_2 + \cdots + y_n \\ \text{pri obmedzeniach} & a_{1,1}y_1 + a_{2,1}y_2 + \cdots + a_{n,1}y_n = c_1 \\ & a_{1,2}y_1 + a_{2,2}y_2 + \cdots + a_{n,2}y_n = c_2 \\ & \vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \\ & a_{1,m}y_1 + a_{2,m}y_2 + \cdots + a_{n,m}y_n = c_m \\ & y_1, \dots, y_n \geq 0 \end{array} \quad (44)$$

alebo skrátene

$$(P) \quad \min_{\mathbf{y} \in \mathbb{R}^n} \{ \mathbf{1}^T \mathbf{y} \mid A^T \mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \},$$

kde A je matica rozmerov $n \times m$, ktorej riadky sú tvorené súradnicami bodov $\mathbf{a}_1, \dots, \mathbf{a}_n$. Na program (44) použijeme náš dualizačný recept a dostaneme duálny program

$$(D) \quad \max_{\mathbf{x} \in \mathbb{R}^m} \{ \mathbf{c}^T \mathbf{x} \mid A \mathbf{x} \leq \mathbf{1} \} \quad (45)$$



Ako môžeme program (45) interpretovať? Zoberme si ľubovoľné prípustné riešenie \mathbf{x} . Nech $\mathcal{H}_{\mathbf{x}}$ je nadrovina daná bodmi \mathbf{y} , pre ktoré $\langle \mathbf{y}, \mathbf{x} \rangle = 1$. Obmedzenia programu (45) hovoria, že pre každý bod \mathbf{a}_i je $\langle \mathbf{a}_i, \mathbf{x} \rangle \leq 1$, a z konvexity je preto celý konvexný obal K pod $\mathcal{H}_{\mathbf{x}}$. Ďalej ak označíme $\alpha = \frac{1}{\mathbf{c}^T \mathbf{x}}$, tak bod $\alpha \mathbf{c}$ leží v nadrovine $\mathcal{H}_{\mathbf{x}}$, lebo $\langle \alpha \mathbf{c}, \mathbf{x} \rangle = 1$. Každému prípustnému riešeniu (45) teda zodpovedá bod $\alpha \mathbf{c}$ na polpriamke generovanej vektorom \mathbf{c} , ktorý (podľa Lemy 4.4) neleží vo vnútri K . Naviac je zrejmé, že optimálne riešenie je nezáporné, a preto môžeme bez ujmy na všeobecnosti predpokladať, že $\langle \mathbf{x}, \mathbf{c} \rangle = \|\mathbf{x}\| \cdot \|\mathbf{c}\| \cos \varphi \geq 0$, kde φ je uhol, ktorý zvierajú vektoru \mathbf{x} a \mathbf{c} . Preto nás zaujímajú iba tie prípustné riešenia, ktorým zodpovedajú body $\alpha \mathbf{c}$ ležiace na polpriamke generovanej \mathbf{c} za K .

Platí to aj naopak. Zoberme si hocjaký bod $\alpha \mathbf{c}$ ležiaci za K . Podľa Lemy 4.4 existuje nadrovina \mathcal{H} taká, že všetky body z K ležia pod ňou. Nech \mathcal{H} je tvorená bodmi \mathbf{y} splňajúcimi $\langle \mathbf{x}, \mathbf{y} \rangle = 1$ pre nejaký vektor \mathbf{x} , potom platí $A \mathbf{x} \leq \mathbf{1}$. Zároveň, pretože \mathcal{H} prechádza cez $\alpha \mathbf{c}$, platí $\langle \mathbf{x}, \alpha \mathbf{c} \rangle = 1 = \alpha \langle \mathbf{x}, \mathbf{c} \rangle$, a teda $\alpha = \frac{1}{\mathbf{c}^T \mathbf{x}}$. Pre maximálnu hodnotu $\mathbf{c}^T \mathbf{x}$ je príslušná α minimálna, a preto program (45) vyžaduje nájsť minimálne α tak, aby bol $\alpha \mathbf{c}$ ležal za K na polpriamke generovanej vektorom \mathbf{c} . To je ale zjavne bod, v ktorom polpriamka opúšta K , a teda programy (44) a (45) majú rovnaké optimálne riešenie. Dokázali sme teda tvrdenie

Lema 4.5. Ak primárny program $\min_{\mathbf{y} \in \mathbb{R}^n} \{ \mathbf{1}^T \mathbf{y} \mid A^T \mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \}$ má prípustné riešenie, potom aj duálny program $\max_{\mathbf{x} \in \mathbb{R}^m} \{ \mathbf{c}^T \mathbf{x} \mid A \mathbf{x} \leq \mathbf{1} \}$ má prípustné riešenie a optimálne hodnoty oboch programov sa rovnajú.

Keby sa nám podarilo Lemu 4.5 zovšeobecniť na programy tvaru $\min_{\mathbf{y} \in \mathbb{R}^n} \{ \mathbf{b}^T \mathbf{y} \mid A^T \mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \}$ pre ľubovoľný vektor \mathbf{b} , boli by sme spokojní, lebo každý lineárny program sa dá napísat v takomto tvare. Majme teda primárny program

$$(P) \quad \min_{\mathbf{y} \in \mathbb{R}^n} \{ \mathbf{b}^T \mathbf{y} \mid A^T \mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \} \quad (46)$$

a k nemu duálny program

$$(D) \max_{\mathbf{x} \in \mathbb{R}^m} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b} \} \quad (47)$$

Uvažujme najprv vektory \mathbf{b} také, že všetky zložky $b_i > 0$. Označme

$$\mathbf{a}'_i = \frac{1}{b_i} \mathbf{a}_i \quad y'_j = b_j y_j \quad (48)$$

Platí $\mathbf{b}^\top \mathbf{y} = \sum_{j=1}^n b_j y_j = \mathbf{1}^\top \mathbf{y}'$ a pre každé $i \in \{1, \dots, m\}$ $\sum_{j=1}^m a_{ji} y_j = \sum_{j=1}^m a'_{ji} b_i \frac{y'_j}{b_j}$. Pretože $b_j > 0$, program (46) je ekvivalentný¹s programom

$$\min_{\mathbf{y}' \in \mathbb{R}^n} \{ \mathbf{1}^\top \mathbf{y}' \mid A'\mathbf{y}' = \mathbf{c}, \mathbf{y}' \geq \mathbf{0} \} \quad (49)$$

kde stĺpce matice A' sú vektory \mathbf{a}'_i . Podľa Lemy 4.5, ak má program (49) prípustné riešenie, tak jeho optimum je rovnaké ako optimum programu

$$\max_{\mathbf{x}' \in \mathbb{R}^m} \{ \mathbf{c}^\top \mathbf{x}' \mid A'^\top \mathbf{x}' \leq \mathbf{1} \} \quad (50)$$

Obmedzenia programu (50) sú v tvare $\sum_{j=1}^m a'_{ji} x'_j \leq 1$ a tak s použitím označenia (48) dostávame, že programy (50) a (47) sú ekvivalentné, preto ak (46) má prípustné riešenie, má ho aj (47) a hodnoty optima sú rovnaké.

Cvičenie. Upravte predchádzajúci postup tak, aby platil pre $b_i \geq 0$.

Nech teraz \mathbf{b} nadobúda ľubovoľné hodnoty a nech \mathbf{x} je prípustné riešenie (47). Označme

$$\mathbf{x}' = \mathbf{x} - \mathbf{x} \quad b'_i = b_i - \mathbf{a}_i^\top \mathbf{x} \quad (51)$$

Ako sa v tomto označení zmenia programy (46) a (47)? Pre prípustné riešenia \mathbf{x}, \mathbf{y} platí

$$\begin{aligned} \mathbf{c}^\top \mathbf{x} &= \mathbf{c}^\top \mathbf{x}' + \mathbf{c}^\top \mathbf{x} \\ \mathbf{b}^\top \mathbf{y} &= \sum_{i=1}^n b_i y_i = \sum_{i=1}^n b'_i y_i + \sum_{i=1}^n y_i (\mathbf{a}_i^\top \mathbf{x}) = \mathbf{b}'^\top \mathbf{y} + \sum_{i=1}^n y_i \sum_{j=1}^m a_{ij} \tilde{x}_j = \\ &= \mathbf{b}'^\top \mathbf{y} + \sum_{j=1}^m \tilde{x}_j (\sum_{i=1}^n y_i a_{ij}) = \mathbf{b}'^\top \mathbf{y} + \mathbf{c}^\top \mathbf{x} \end{aligned}$$

kde posledná rovnosť vyplýva z toho, že $A^\top \mathbf{y} = \mathbf{c}$. Pretože $\mathbf{c}^\top \mathbf{x}$ je konštanta, programy (46) a (47) vieme ekvivalentne zapísat ako

$$(P') \min_{\mathbf{y} \in \mathbb{R}^n} \{ \mathbf{b}'^\top \mathbf{y} \mid A^\top \mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \} \quad (D') \max_{\mathbf{x}' \in \mathbb{R}^m} \{ \mathbf{c}^\top \mathbf{x}' \mid A\mathbf{x}' \leq \mathbf{b}' \}$$

Navýše z prípustnosti \mathbf{x} pre (47) vyplýva, že $\mathbf{b}' \geq \mathbf{0}$ a preto môžeme aplikovať predchádzajúci prípad.

Načrtli sme² dôkaz fundamentálnej vety v teórii lineárneho programovania:

Veta 4.6 (Silná veta o dualite). *Pre dvojicu duálnych lineárnych programov*

$$(P) \min_{\mathbf{y} \in \mathbb{R}^n} \{ \mathbf{b}^\top \mathbf{y} \mid A^\top \mathbf{y} = \mathbf{c}, \mathbf{y} \geq \mathbf{0} \} \quad (D) \max_{\mathbf{x} \in \mathbb{R}^m} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b} \}$$

Platí práve jedna zo štyroch možností:

1. (P) ani (D) nemajú žiadne prípustné riešenie
2. (P) je neohraničený a (D) nemá prípustné riešenie
3. (D) je neohraničený a (P) nemá prípustné riešenie
4. (P) aj (D) majú prípustné riešenie. V tom prípade sa optimálne hodnoty (P) a (D) rovnanjú.

¹v tom zmysle, že každému riešeniu programu (46) prislúcha nejaké riešenie programu (49) s rovnakou hodnotou a naopak

²pre kompletnejší dôkaz treba ošetriť ešte niekoľko špeciálnych prípadov, ktoré prenechávame na čitateľa

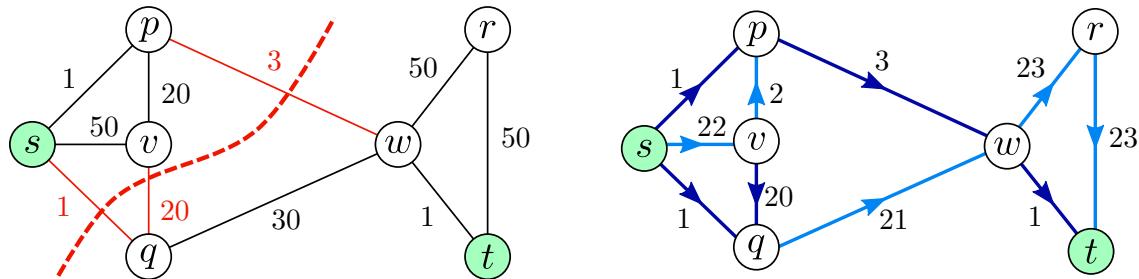
4.2 MAX-FLOW–MIN-CUT očami duality

Dualita lineárneho programovania, ktorú sme predstavili v predchádzajúcej časti, často pomáha lepšie pochopiť min–max charakterizácie výpočtových problémov. V tejto časti si z pohľadu duality priblížime známe tvrdenie, že veľkosť maximálneho toku sa rovná kapacite minimálneho rezu.

Začneme tým, že predstavíme problém maximálneho toku. Keďže predpokladáme, že čitateľ sa s ním už stretol, odpustíme si tentokrát motivačné rozprávky a zopakujeme len definíciu. Majme daný (neorientovaný) graf $G = (V, E)$ s nezápornými váhami hrán, t.j. funkciou $c : E \mapsto \mathbb{R}^+$; namiesto $c(\{u, v\})$ budeme skrátene písat $c_{uv} = c_{vu}$. V grafe sú dva význačné vrcholy s a t . V probléme MAX-FLOW je cieľom nájsť čo najväčší tok z s do t . Tok je funkcia $f : V^2 \mapsto \mathbb{R}^+$, ktorej interpretácia je taká, že $f(u, v)$ je množstvo kvapaliny, ktorá tečie z u do v . Funkcia f musí splňať tieto vlastnosti:

1. ak $f(u, v) \neq 0$, potom $(u, v) \in E$, t.j. tiečť musí po hranách grafu
2. $f(u, v) = -f(v, u)$, t.j. znamienko udáva, ktorým smerom tok ide
3. pre každé $v \notin \{s, t\}$ platí $\sum_{u \in V} f(u, v) = 0$, t.j. čo do vrchola vteká, to z neho aj vyečie

Veľkosť toku je $\sum_{v \in V} f(s, v)$. Rez v grafe G je množina hrán, ktorej odobratie oddeli vrcholy s a t . Cena rezu je súčet váh prerezaných hrán. Problém MIN-CUT je nájsť rez minimálnej ceny. Na nasledovnom obrázku je graf s kapacitami hrán a minimálnym rezom (vľavo) a maximálnym tokom (vpravo; tmavo sú označené hrany, ktorých kapacita je naplnená tokom) s hodnotou 24.



Problém MAX-FLOW sa dá prirodzene formulovať ako úloha lineárneho programovania, a to hned niekoľkými spôsobmi. Z dôvodov, ktoré, ako dúfame, budú jasné na konci tejto časti, si zvolíme takúto formuláciu: pre každú hranu $(u, v) \in E$ budeme mať dve nezáporné premenné x_{uv} a x_{vu} , ktoré budú udávať množstvo toku z u do v (všimnite si, že sme tým povolili situáciu, že po tej istej hrane tečie tok oboma smermi: x_{uv} v smere z u do v a x_{vu} v smere z v do u ; ničmenej, tieto dve hodnoty stačí odčítať a dostaneme riešenie v zmysle našej definície). Ďalej budeme mať jednu premennú f , ktorá bude udávať veľkosť toku (a tú sa budeme snažiť maximalizovať): $f = \sum_{u:(s,u) \in E} x_{su} - \sum_{u:(s,u) \in E} x_{us}$. Zákony zachovania toku aj obmedzenia kapacít ľahko zapíšeme pomocou lineárnych nerovností a dostaneme program

$$\begin{aligned}
& \text{maximalizovať} && f \\
& \text{pri obmedzeniach} && \\
& \sum_{u:(s,u) \in E} x_{su} - \sum_{u:(s,u) \in E} x_{us} - f = 0 \\
& \sum_{u:(t,u) \in E} x_{ut} - \sum_{u:(s,u) \in E} x_{tu} + f = 0 \\
& \sum_{u:(u,v) \in E} x_{vu} - \sum_{u:(u,v) \in E} x_{uv} = 0 \quad \forall v \in V - \{s, t\} \\
& x_{uv} \leq c_{uv} \quad \forall (u, v) \in E \\
& x_{vu} \leq c_{uv} \quad \forall (u, v) \in E \\
& x_{uv} \geq 0 \quad \forall (u, v) \in E
\end{aligned} \tag{52}$$

Prvé a druhé obmedzenie definujú veľkosť toku f ako to, čo vyteká z s , resp. čo vteká do t . Vo všetkých ostatných vrcholoch platí zákon zachovania toku.

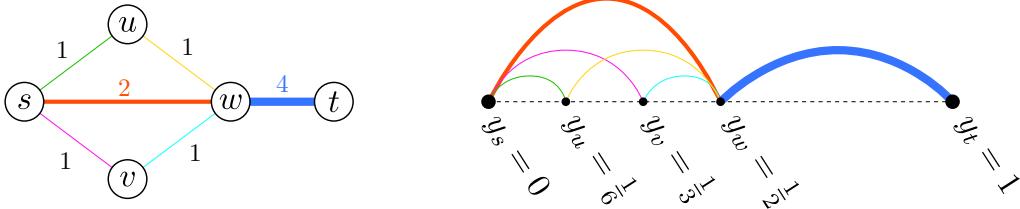
Použime teraz nás dualizačný recept a zostrojme k programu (52) duálny minimalizačný program. Každému obmedzeniu v primárnom programe zodpovedá premenná v duálnom programe. V programe (52) máme dva druhy obmedzení: obmedzenia v tvare $\dots = 0$ pre každý vrchol a obmedzenia $x_{uv} \leq c_{uv}$. Zavedieme si preto duálne premenné $y_v \in \mathbb{R}$ pre každý vrchol $v \in V$ (y_s zodpovedá prvému obmedzeniu, y_t druhému a ostatné y_v v poradí ďalším) a premenné z_{uv} a z_{vu} pre každú hranu $(u, v) \in E$, pričom $z_{uv}, z_{vu} \geq 0$. Duálny program sa vyrába tak, že i -ta duálna premenná je multiplikátor, ktorým prenasobíme i -te obmedzenie a výsledky sčítame; súčet pravých strán je príslušný odhad a vyžadujeme, aby súčet ľavých strán bol v každej zložke väčší ako maximalizovaná funkcia. Súčet pravých strán je v našom prípade $\sum_{(u,v) \in E} (z_{uv} + z_{vu})c_{uv}$. Maximalizovaná funkcia

má pri všetkých premenných okrem f hodnotu 0. Premenná f vystupuje iba v prvom a druhom obmedzení (52), preto dostávame v duálnom programe obmedzenie $y_t - y_s = 1$ (je jasné, že tok je nezáporný, ale formálne sme v (52) nevyžadovali, aby $f \geq 0$). Každá premenná x_{uv} sa vyskytuje v dvoch obmedzeniach: v obmedzení zodpovedajúcom vrcholu u s kladným znamienkom, v obmedzení zodpovedajúcom vrcholu v so záporným znamienkom a v obmedzení $x_{uv} \leq c_{uv}$. Dostávame tak duálny program

$$\begin{aligned}
& \text{minimalizovať} && \sum_{(u,v) \in E} (z_{uv} + z_{vu}) c_{uv} \\
& \text{pri obmedzeniach} && \\
& y_t - y_s = 1 \\
& y_v - y_u + z_{uv} \geq 0 \quad \forall (u, v) \in E \\
& y_u - y_v + z_{vu} \geq 0 \quad \forall (u, v) \in E \\
& z_{uv}, z_{vu} \geq 0 \quad \forall (u, v) \in E
\end{aligned} \tag{53}$$

Z vety o dualite vieme, že programy (52) a (53) majú rovnakú hodnotu optima. Ako môžeme program (53) interpretovať? Bez ujmy na všeobecnosť môžeme predpokladať, že aspoň jedna z dvojice premenných z_{uv}, z_{vu} je nulová: jediné obmedzenia na z_{uv} a z_{vu} sú $z_{uv} \geq y_u - y_v$ a $z_{vu} \geq y_v - y_u$. Zjavne aspoň jedna z hodnôt $y_u - y_v$ a $y_v - y_u$ nie je kladná, a teda ak máme ľubovoľné prípustné riešenie, tak aspoň jednu z premenných z_{uv}, z_{vu} môžeme nastaviť na 0 a nezväčšíme hodnotu minimalizovanej funkcie. Môžeme si preto označiť $\bar{z}_{uv} = \max\{z_{uv}, z_{vu}\}$; obmedzenia potom hovoria, že $\bar{z}_{uv} \geq |y_u - y_v|$ a z rovnakých dôvodov ako pred chvíľou môžeme bez ujmy na všeobecnosť predpokladať, že $\bar{z}_{uv} = |y_u - y_v|$. Môžeme si predstaviť, že vrcholy grafu ukladáme na priamku: vrchol v je uložený v bode y_v , pričom, opäť bez ujmy na všeobecnosť, je s uložený v bode 0 a t v bode 1. \bar{z}_{uv} je dĺžka hrany (u, v) v našom uložení. Keď to zhrnieme, optimálne riešenie programu

(53) nám dá také uloženie vrcholov grafu na úsečku dĺžky 1, že s a t sú na krajoch a minimalizuje sa váhovaná dĺžka hrán.



Graf s váhami hrán a jedno z možných optimálnych riešení programu (53): $\bar{z}_{su} = \bar{z}_{vw} = \frac{1}{6}$, $\bar{z}_{sv} = \bar{z}_{uw} = \frac{1}{3}$, $\bar{z}_{sw} = \bar{z}_{wt} = \frac{1}{2}$. Výsledná hodnota je

$$\sum_{(u,v) \in E} c_{uv} \bar{z}_{uv} = \frac{1}{6} + \frac{1}{3} + 2 \frac{1}{2} + \frac{1}{3} + \frac{1}{6} + 4 \frac{1}{2} = 4.$$

Podme teraz prepísat program (53) do normálneho tvaru $\max\{-\mathbf{c}^T \boldsymbol{\beta} \mid A\boldsymbol{\beta} = 0, \boldsymbol{\beta} \geq 0\}$. Ku každej premennej z_{uv} zavedieme premennú $\hat{z}_{uv} \geq 0$, aby sme získali obmedzenia tvaru $y_v - y_u + z_{uv} - \hat{z}_{uv} = 0$. Ak vektor $\boldsymbol{\beta}$ pozostáva zaradom z hodnôt $y_s, y_t, y_{v_1}, \dots, z_{uv}, z_{vu}, \dots, \hat{z}_{uv}, \hat{z}_{vu}, \dots$, matica A má takúto štruktúru:

$$A = \begin{array}{|ccc|ccc|ccc|} \hline & \overbrace{y_s \quad y_t \quad \dots \quad y_u \quad \dots \quad y_v} & & \overbrace{z_{uv} \quad z_{vu} \quad \dots \quad \hat{z}_{uv} \quad \hat{z}_{vu} \quad \dots} & \\ \hline -1 & 1 & & & & & & & & \\ & & 1 & & & & -1 & & & \\ & & & 1 & & & & -1 & & \\ & & & & 1 & & & & -1 & \\ & & & & & 1 & & & & -1 \\ & & & & & & 1 & & & \\ & & & & & & & -1 & & \\ & & & & & & & & -1 & \\ & & & & & & & & & -1 \\ \hline \end{array} \quad \left. \right\} (u, v) \in E$$

Nasledujúcu priamočiaru povinnú jazdu preneháme na čitateľa:

Cvičenie. S pomocou viet 3.6 a 3.8 ukážte, že matica A je TUM.

Z Vety 3.5 vyplýva, že existuje celočíselné optimálne riešenie programu (53). To znamená, že všetky vrcholy majú $y_v \in \{0, 1\}$ (t.j. sú uložené na niektorom konci úsečky) a každá hrana má dĺžku 0 alebo 1. Preto každá cesta z s do t musí obsahovať aspoň jednu hranu s dĺžkou 1, a teda ak hrany dĺžky 1 z grafu odstránime, dostaneme rez, ktorý oddeluje s od t . Môžeme teda povedať

Veta 4.7 (MAX-FLOW-MIN-CUT veta). *Veľkosť maximálneho toku sa rovná kapacite minimálneho rezu.*

Dôkaz: Veľkosť maximálneho toku je optimum programu (52), ktoré je rovnaké, ako optimum programu (53). Existuje celočíselné optimum programu (53), a zároveň je bijektia medzi $s - t$ rezmi a celočíselnými riešeniami (53): celočíselnému riešeniu (53) prislúcha rez a každému rezu vieme prirodzeným spôsobom priradiť riešenie (53). \square

Je dosť možné, že čitateľ možno pozná oveľa jednoduchší dôkaz tohto tvrdenia. Dôvod, prečo uvádzame tento dôkaz je (okrem toho, že chceme ilustrovať dualitu lineárnych programov) ten, že ukazuje dvojicu MAX-FLOW-MIN-CUT ako špeciálny prípad všeobecnejších problémov; zároveň dáva nový pohľad na otázku, prečo rovnaký výsledok neplatí pre iné podobné problémy (ak napríklad máme viaceru zdrojov a ústí a cielom je maximalizovať tok rôznych komodít v spoľočných potrubiacach, máme duálany problém k problému MIN-MULTI-CUT z Definície 3.14, avšak analogický výsledok neplatí) cez unimodularitu príslušných matíc.

4.3 Primárno-duálna metóda

Dúfame, že sme čitateľa presvedčili o tom, že dualita je zaujímavá vlastnosť lineárnych programov. Teraz je načase ukázať, ako sa dá využiť pri návrhu algoritmov. Pozrime sa na dvojicu duálnych programov

$$(P) : \min_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq 0 \}$$

$$(D) : \max_{\mathbf{y} \in \mathbb{R}^m} \{ \mathbf{b}^\top \mathbf{y} \mid A^\top \mathbf{y} \leq \mathbf{c}, \mathbf{y} \geq 0 \}$$

Z vety o dualite vieme, že majú rovnakú hodnotu optimu, t.j. že existujú vektory $\mathbf{x}^* \geq 0$ a $\mathbf{y}^* \geq 0$, že $A\mathbf{x}^* \geq \mathbf{b}$, $A^\top \mathbf{y}^* \leq \mathbf{c}$ a $\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*$. Pripomeňme si nerovnosti z dôkazu Vety 4.1, ktoré platia pre všetky dvojice prípustných riešení primárnej a duálnej úlohy, a teda aj pre \mathbf{x}^* , \mathbf{y}^* :

$$\mathbf{c}^\top \mathbf{x}^* \stackrel{(\clubsuit)}{\geq} (A^\top \mathbf{y}^*)^\top \mathbf{x}^* = \mathbf{y}^{*\top} A\mathbf{x}^* \stackrel{(\diamondsuit)}{\geq} \mathbf{y}^{*\top} \mathbf{b} \quad (54)$$

Kedže $\mathbf{c}^\top \mathbf{x}^* = \mathbf{b}^\top \mathbf{y}^*$, (\clubsuit) aj (\diamondsuit) musia byť rovnosti. Pozrime sa na (\clubsuit) a rozpišme vektorový zápis pomocou sumy:

$$\sum_{j=1}^n c_j x_j^* = \sum_{j=1}^n [A^\top \mathbf{y}^*]_j x_j^*, \quad (55)$$

kde symbol $[.]_j$ označuje j -ty prvok vektora. Z prípustnosti duálneho riešenia vieme, že $\mathbf{c} \geq A^\top \mathbf{y}^*$; nerovnosť vektorov platí v každej zložke, takže pre každé j je $c_j \geq [A^\top \mathbf{y}^*]_j$. Pre každé j je $x_j^* \geq 0$, a teda aj $c_j x_j^* \geq [A^\top \mathbf{y}^*]_j x_j^*$. Z toho vidieť, že ak má platiť rovnosť (55), musí platiť rovnosť v každej zložke.

Rovnaké úvahy sa dajú urobiť pre nerovnosť (\diamondsuit) a dostaneme tak nasledovnú charakterizáciu optimálneho riešenia:

Veta 4.8 (podmienky komplementarity). *Nech \mathbf{x} , \mathbf{y} sú prípustné riešenia primárnej a duálnej úlohy. Potom \mathbf{x} , \mathbf{y} sú obidve optimálne vtedy a len vtedy, ak sú splnené obe nasledujúce podmienky:*

- primárna podmienka komplementarity:

$$\forall 1 \leq j \leq n : \text{ bud } x_j = 0 \text{ alebo } \sum_{i=1}^m a_{ij} y_i = c_j$$

- duálna podmienka komplementarity:

$$\forall 1 \leq i \leq m : \text{ bud } y_i = 0 \text{ alebo } \sum_{j=1}^n a_{ij} x_j = b_i$$

Edmondsov algoritmus pre MIN-1-FACTOR

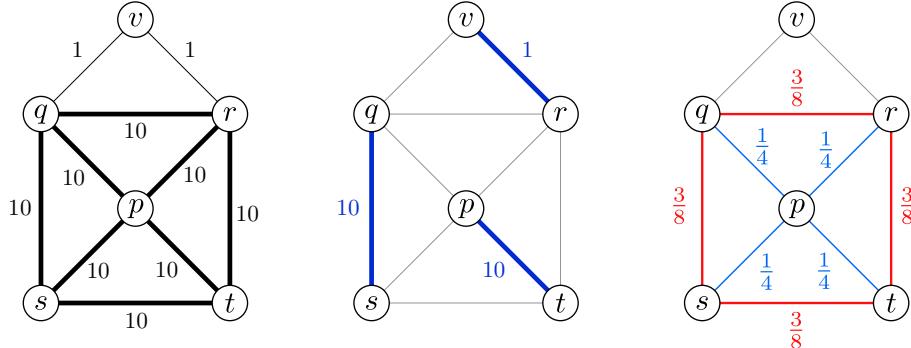
Podmienky komplementarity sú užitočný nástroj pri analýze algoritmov, lebo poskytujú jednoduchý invariant, ktorý charakterizuje optimum. Pripomeňme si definíciu problému MAX-WEIGHTED-BIPARTITE-MATCHING z časti o ILP:

Definícia 3.3. Majme daný bipartitný graf s hranami ohodnotenými nezápornými reálnymi číslami. Cieľom problému MAX-WEIGHTED-BIPARTITE-MATCHING je nájsť množinu hrán s najväčším súčtom váh tak, aby žiadne dve vybraté hrany nezdieľali vrchol.

Problém sme vtedy formulovali ako celočíselný program

$$\begin{aligned} \text{maximalizovať } & \sum_{e \in E} \omega_e x_e \\ \text{pri obmedzeniach } & \sum_{\substack{e \in E \\ e=(v,w)}} x_e \leq 1 \quad \forall v \in V \\ & x_e \geq 0 \quad \forall e \in E \\ & x_e \in \mathbb{Z} \end{aligned}$$

kde $\omega \in \mathbb{R}^n$ je vektor váh hrán. Ukázali sme, že matica obmedzení je TUM, a preto stačí vyriešiť relaxovaný program a podmienky celočíselnosti máme zadarmo. Celkom prirodzene sa natíska otázka: "A čo keby ten graf neboli bipartitný?" Formulácia ILP bude stále v poriadku, ale už nebude platit, že musí existovať optimálne celočíselné riešenie.



Vľavo je graf s váhami hrán. Maximálne párovanie má hodnotu 21: hrany váhy 10 sú iba medzi vrcholmi $\{p, q, r, s, t\}$, t.j. môžu byť v riešení najviac dve. Zo zvyšných hrán môže byť najviac 1. Vpravo riešenie relaxovaného programu s hodnotou 25.

Prv, než budeme pokračovať, upravíme trocha formuláciu nášho problému. Párovaniu, ktoré po krýva všetky hrany, t.j. takej množine hrán $E' \subseteq E$, že každý vrchol susedí s práve jednou hranou z E' , budeme hovoriť *perfektné párovanie*, respektívne 1-faktor (pochopiteľne, aby graf mal 1-faktor, musí mať párný počet vrcholov). Namiesto hľadania najtažšieho párovania nám stačí vedieť hľadať najtažší 1-faktor: z grafu G vyrábime G' tak, že ak má G nepárný počet vrcholov, pridáme k nemu jeden izolovaný vrchol a potom všetky vrcholy, ktoré nie sú spojené hranou spojíme hranou váhy 0. Čitateľ sa ľahko presvedčí, že párovania v G zodpovedajú 1-faktorom v G' .

Ďalej nech ω_{\max} je maximálna váha hrany v G' . Ak vyrábime G'' tak, že nastavíme nové váhy $\omega''_e = \omega_{\max} - \omega_e$, tak vidno, že najľahší 1-faktor v G'' je najtažší 1-faktor v G'' . Dostali sme sa tak k nasledovnej definícii:

Definícia 4.10. Majme daný úplný ohodenotený graf $G = (V, E)$ na $n = 2k$ vrcholoch, pričom hrana $e = (u, v) \in E$ má váhu $\omega_e \in \mathbb{R}^+$. Problém MIN-1-FACTOR je nájsť 1-faktor v G , pre ktorý je súčet váh hrán minimálny, t.j. $\min_{E'} \sum_{e \in E'} \omega(e)$, kde minimum je brané cez všetky 1-faktory E' .

Čitateľ, ktorý s nami vydržal až potialto, určite nebude mať ľahkosti zapísat MIN-1-FACTOR ako celočíselný lineárny program: pre každú hranu $e \in E$ zavedieme premennú $x_e \in \{0, 1\}$, ktorá vyjadruje, či je hrana e vybraná do párovania alebo nie. Vybraná množina hrán je 1-faktor práve vtedy, ak s každým vrcholom susedí práve jedna vybraná hrana, čo priamočiaro zapíšeme

$$\begin{aligned} \text{minimalizovať } & \sum_{e \in E} \omega_e x_e \\ \text{pri obmedzeniach } & \sum_{\substack{e \in E \\ e=(u,v)}} x_e = 1 \quad \forall v \in V \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{aligned} \tag{56}$$

Ak tento program relaxujeme a namiesto $x_e \in \{0,1\}$ budeme požadovať iba $x_e \geq 0$ (čitateľ si všimne, že $x_e \leq 1$ vyplýva z minimality), optimum už nemusí byť celočíselné. V nasledujúcich odstavcoch predstavíme primárno-duálny prístup podľa Edmondsa [?]. Najprv ale ešte jedno označenie, ktoré nám zjednoduší zápis:

Definícia 4.11 (hranová hranica množiny). Majme graf $G = (V, E)$ a množinu vrcholov $S \subseteq V$. Hranová hranica množiny S , označovaná $\delta(S)$, je množina hrán s jedným koncom v S a druhým mimo S , t.j.:

$$\delta(S) := \{e \in E \mid e = (u, v), u \in S, v \in V \setminus S\}$$

Ako sa vysporiadaj s tým, že relaxácia programu (56) nemá celočíselné riešenie? Trik je v tom, že pridáme (vela) dodatočných obmedzení, ktoré zabezpečia celočíselnosť optimu. Nech \mathcal{S} sú všetky aspoň trojprvkové množiny s nepárnym počtom vrcholov, t.j.

$$\mathcal{S} := \{S \subseteq V \mid |S| > 1, |S| \text{ nepárne}\}$$

Každá hrana má dva konce, a preto vrcholy z $S \in \mathcal{S}$ nemôžu byť popárované medzi sebou, takže v každom 1-faktore musí aspoň 1 hrana odchádzať z S . Pridáme tieto obmedzenia k relaxovanému programu (56), čím dostaneme

$$\begin{aligned} \text{minimalizovať} \quad & \sum_{e \in E} \omega_e x_e \\ \text{pri obmedzeniach} \quad & \sum_{e \in \delta(\{v\})} x_e = 1 \quad \forall v \in V \\ & \sum_{e \in \delta(S)} x_e \geq 1 \quad \forall S \in \mathcal{S} \\ & x_e \geq 0 \quad \forall e \in E \end{aligned} \tag{57}$$

...a voilà! Máme lineárny program, ktorý má celočíselné optimálne riešenie. Dostali sme sa však z dažďa pod odkvap: po prvej potrebujeme dokázať, že naozaj existuje celočíselné optimum programu (57) a podruhé potrebujeme riešiť program, ktorý má exponenciálne vela obmedzení. Ani jeden z týchto problémov nie je neprekonateľný, ale my to spravíme elegantne: využijeme dualitu a vyhneme sa riešeniu (57); a navyše dôkaz celočíselnosti vypadne zadarmo. Spôsob návrhu algoritmov, ktorý tu ukážeme, sa zvykne nazývať *primárno-duálna metóda*.

Použime náš dualizačný recept a zostrojme duálny program k (57). Duálny program bude maximalizačný a bude mať premennú pre každé obmedzenie. Máme dva typy obmedzení: pre vrcholy a pre množiny, takže si zavedieme duálne premenné $r_v \in R$ pre $v \in V$ a premenné $w_S \in \mathbb{R}^+$ pre $S \in \mathcal{S}$. Každá primárna premenná x_e má vklad $x_e \omega_e$ do minimalizačnej funkcie a vyskytuje sa v dvoch obmedzeniach pre vrcholy (konkrétnie pre dva koncové vrcholy e) a v obmedzeniach pre tie množiny $S \in \mathcal{S}$, kde $e \in \delta(S)$. Dostali sme duálny program (všimnite si, že r_v môže byť aj záporné):

$$\begin{aligned} \text{maximalizovať} \quad & \sum_{v \in V} r_v + \sum_{S \in \mathcal{S}} w_S \\ \text{pri obmedzeniach} \quad & r_u + r_v + \sum_{\substack{S \in \mathcal{S} \\ e \in \delta(S)}} w_S \leq \omega_e \quad \forall e = (u, v) \in E \\ & w_S \geq 0 \quad \forall S \in \mathcal{S} \end{aligned} \tag{58}$$

Z didaktických dôvodov, a aj preto, že v našej dvojici programov nie sú všetky premenné nezáporné, prepíšme nerovnosť (54) v našom značení:

$$\sum_{e \in E} x_e \omega_e \stackrel{\clubsuit}{\geq} \sum_{\substack{e \in E \\ e = (u, v)}} x_e (r_u + r_v + \sum_{\substack{S \in \mathcal{S} \\ e \in \delta(S)}} w_S) \stackrel{\heartsuit}{=} \sum_{v \in V} (r_v \sum_{\substack{e \in \delta(\{v\})}} x_e) + \sum_{S \in \mathcal{S}} w_S \left(\sum_{e \in \delta(S)} x_e \right) \stackrel{\diamondsuit}{\geq} \sum_{v \in V} r_v + \sum_{S \in \mathcal{S}} w_S$$

Rovnosť (\heartsuit) platí preto, lebo na ľavej strane je pre každý vrchol $v \in V$ hodnota r_v zarátaná s koeficientom x_e pre všetky hrany incidentné s v a hodnota w_S je zarádaná s koeficientom x_e pre všetky hrany z hranice S . Z obmedzení programu (57) vyplýva, že modrá suma $\sum_{e \in \delta(\{v\})} x_e = 1$ a červená suma $\sum_{e \in \delta(S)} x_e \geq 1$, preto podmienky komplementarity majú tvar

$$\mathbf{S1}(\clubsuit) \quad \forall e = (u, v) \in E : \quad x_e > 0 \Rightarrow r_u + r_v + \sum_{\substack{S \in \mathcal{S} \\ e \in \delta(S)}} w_S = \omega(e)$$

$$\mathbf{S2}(\diamondsuit) \quad \forall S \in \mathcal{S} : \quad w_S > 0 \Rightarrow \sum_{e \in \delta(S)} x_e = 1$$

Pozrime sa teraz na program (58) a skúsme mu dať nejakú intuitívnu interpretáciu. Predstavme si, že okolo každého vrchola $v \in V$ (resp. okolo každej množiny $S \in \mathcal{S}$) môže byť bublina s nábojom r_v (resp. w_S). Samozrejme, množiny s \mathcal{S} sa môžu rôzne prekrývať, a tak predstava bublín okolo každej z nich nie je úplne intuitívna, ničmenej nakoniec budeme využívať iba systémy do seba zapadajúcich bublín. Program (58) nám hovorí, že chceme maximalizovať celkový náboj. Váhy hrán si teraz vieme predstaviť ako kapacitu a obmedzenia hovoria, že žiadnu hranu e nesmieme „pretrhnúť“: celkový náboj na všetkých bublinách, ktoré pretínajú e nesmie prekročiť jej kapacitu. Hranu e , pre ktorú platí $r_u + r_v + \sum_{\substack{S \in \mathcal{S} \\ e \in \delta(S)}} w_S = \omega(e)$, nazveme *plná*. Podmienky **S1** a **S2** spolu s vetou o dualite vieme interpretovať takto:

Lema 4.12. Ak (ľubovoľným spôsobom) nájdeme 1-faktor M a hodnoty bublín \mathbf{r} a \mathbf{w} tak, že

(I1) žiadna hrana nie je preplnená,

(I2) hodnoty $w_S \geq 0$,

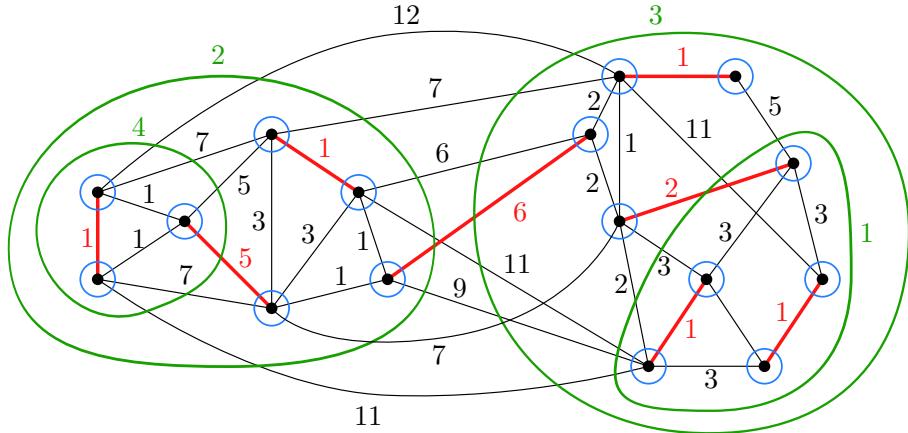
(I3) všetky hrany z M sú plné a

(I4) z každej nenulovej bubliny $S \in \mathcal{S}$ odchádza práve jedna hrana z M ,

tak máme optimálne riešenie dvojice programov (57) a (58), ktoré má celočíselné hodnoty \mathbf{x} , a teda tvorí minimálny 1-faktor.

Dôkaz: Podmienky (I1) a (I2) zaručujú prípustnosť duálneho programu (58). Fakt, že M je 1-faktor zaručuje prípustnosť primárneho programu (57). Podmienky (I3) a (I4) sú vporadí podmienky **S1** a **S2**, takže tvrdenie je dôsledok Vety 4.8. \square

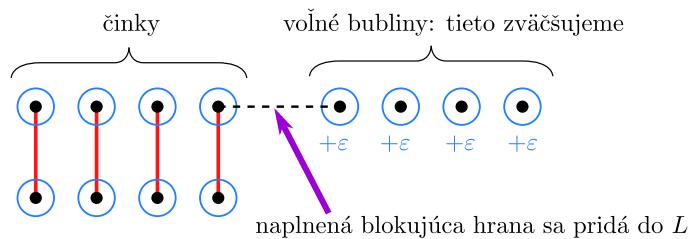
Od tohto momentu môžeme zabudnúť na celé lineárne programovanie a ideme sa snažiť nájsť algoritmus, ktorý vyrobí hľadané objekty M , \mathbf{r} a \mathbf{w} . Ako sme už naznačili, keďže \mathbf{w} je príliš veľké, budeme si pamätať iba jeho nenulové zložky; zároveň budeme hľadať iba také riešenia, v ktorých sa množiny s nenulovými bublinami nepretínajú. Chceli by sme dostať nejakú takúto štruktúru:



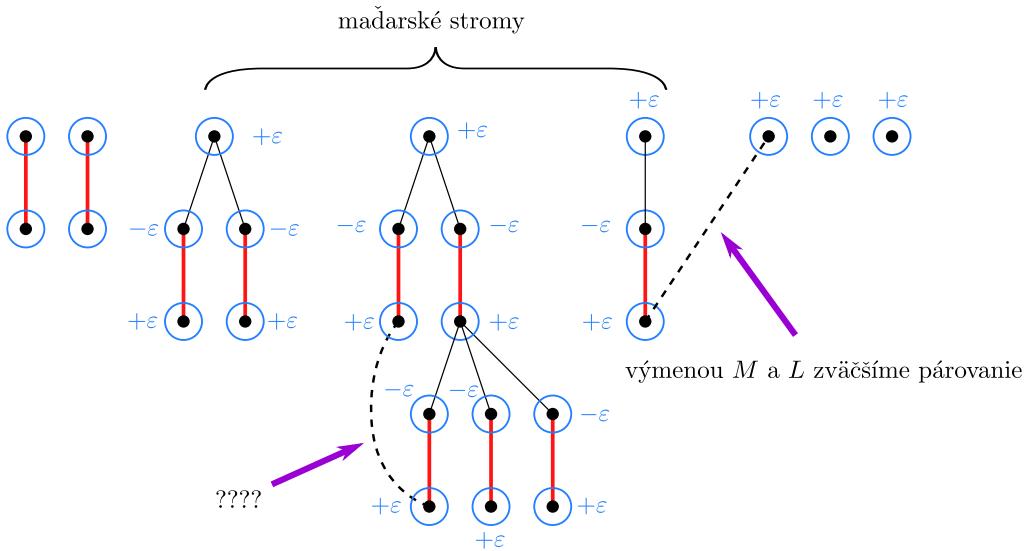
Modré bubliny majú všetky hodnotu $1/2$, hrany, ktoré nie sú zakreslené, majú váhu ∞ . Červené hrany tvoria 1-faktor. Žiadna hrana nie je preplnená, všetky červené hrany sú plné a z každej zelenej bubliny odchádza práve jedna červená hrana; preto vieme, že červený 1-faktor je minimálny.

Samozrejme, nikde zatiaľ nemáme zaručené, že takáto konfigurácia vždy existuje. Ale ak to dokážeme a nájdeme algoritmus, ktorý ju vyrobí, môžeme zajasať a úlohu označiť za splnenú.

Začnime neformálnym opisom činnosti algoritmu po štarte. Počas celého algoritmu budú podmienky **(I1)**, **(I2)** a **(I3)** udržiavané v platnosti. Algoritmus začne s tým, že párovanie M bude prázdne a všetky bubliny budú nulové. Postupne sa bude snažiť pridávať náboj do bublín a hrany do párovania tak, aby nakoniec M bol 1-faktor a boli splnená podmienka **(I4)**; potom M bude minimálny 1-faktor. V prvom kroku začne pridávať náboj na všetky bubliny r_v (v ďalšom budeme tieto bubliny volať *modré*, kým bubliny w_S budú *zelené*). Časom sa stane, že dve modré bubliny r_u a r_v naplnia nejakú hranu $e = (u, v)$. Hrana e sa pridá do M a bublinám r_u a r_v sa nebude v ďalšom kroku pridávať náboj (budú tvoriť *činku*). Raz sa ale naplní aj nejaká hraná e , ktorej jeden vrchol už patrí hrane z M . Táto hrana je plná, a preto jej koncové bubliny sa nemôžu zväčšiť, a zároveň sa nemôže pridať do M . Nazveme ju *blokujúca* hrana a algoritmus si bude udržiavať množinu blokujúcich hrán L .

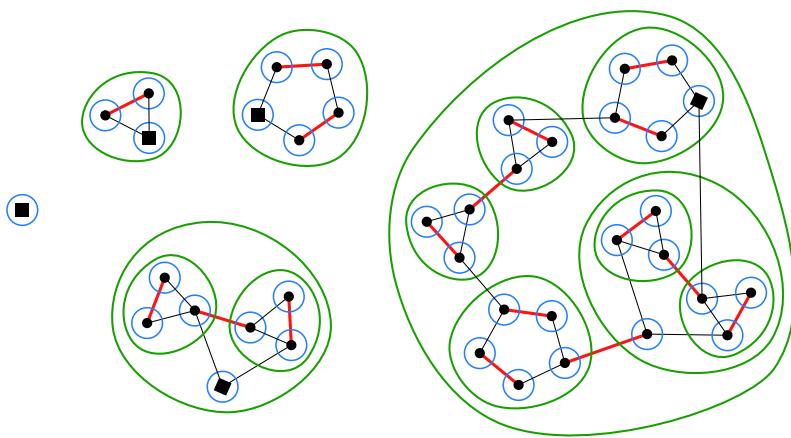


Okrem voľných bublín a činek vznikol ďalší útvar: cesta dĺžky 3 zložená z hrany z L a hrany z M ; cesty, na ktorých sa striedajú hrany z L a M budeme volať *alternujúce*. Keď sa v ďalšom bude voľným bublinám pridávať náboj $+\varepsilon$, bublinám z alternujúcej cesty sa bude striedavo pridávať $+\varepsilon$ a $-\varepsilon$, aby sa žiadna hrana nepreplnila (pripomíname, že na rozdiel od zelených bublín, modré bubliny môžu mať záporný náboj). V ďalšom sa naplní hrana medzi dvoma bublinami, ktorým sa náboj pridáva, a môžu tak vznikať stromy z alternujúcich ciest (tzv. *maďarské stromy*). Ak naplnenou hranou vznikne alternujúca cesta s nepárnym počtom hrán, je to dobré, lebo sa môže zväčšiť párovanie tak, že na tejto ceste sa vymení príslušnosť hrán medzi M a L a z alternujúcej cesty ostane sada činek.



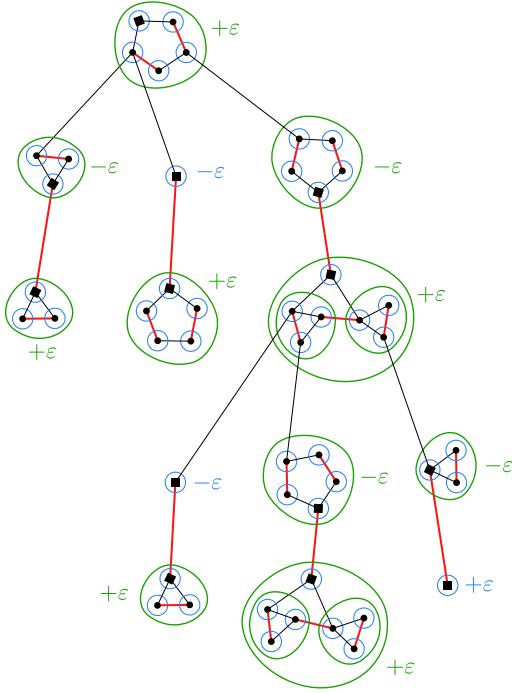
Čo ale máme robiť, ak sa napríklad naplní hrana medzi dvoma bublinami v jednom strome? Toto je moment, v ktorom dozrel čas na zelené bublinky a ich hierarchické štruktúry.

Základnou dátovou štruktúrou algoritmu je *kvet*. Každý kvet má vonkajšiu bublinu a v nej jeden význačný vrchol zvaný *stopka*. Najjednoduchší kvet je jeden vrchol s modrou bublinou okolo neho. Zložitejšie kvety vznikajú rekurzívne: majme nepárny počet kvetov $K_1, K_2, \dots, K_{2r+1}$, $r \geq 1$ (t.j. aspoň tri kvety) tak, že stopky kvetov K_{2i} a K_{2i+1} pre $i = 1, \dots, r$ sú spojené hranou z M , a zároveň pre každú dvojicu kvetov $A := K_{2i-1}$ a $B := K_{2i}$ pre $i = 1, \dots, r$ a $A := K_{2r+1}$ a $B := K_1$ existujú vrcholy $u \in A$ a $v \in B$ také, že hrana $(u, v) \in L$. Potom bublina ohraňujúca K_1, \dots, K_{2r+1} vytvorí nový kvet, ktorého stopka bude stopka kvetu K_1 . Kvety, ktoré nie sú súčasťou iného kvetu, budeme volať *vonkajšie kvety*.



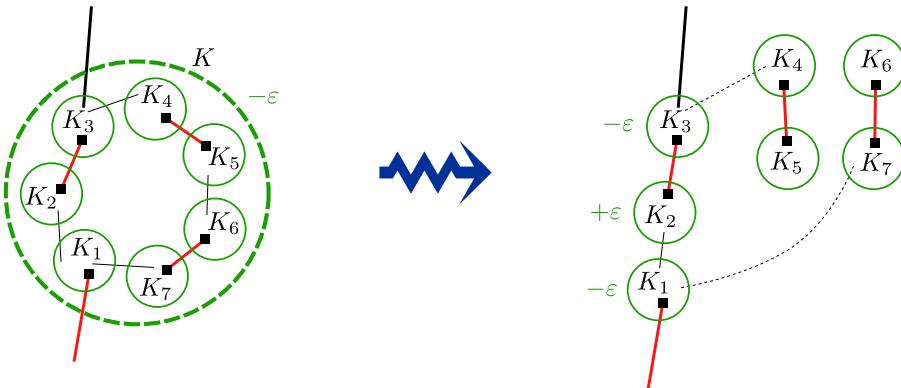
Päť rôznych vonkajších kvetov so stopkami označenými štvorcom. Červené hrany sú z M , čierne z L .

Za povšimnutie stojí, že kvet uzatvára časť grafu, ktorú máme "takmer hotovú": okrem stopky sú všetky vrcholy kvetu pospájané hranami z M a zároveň každú bublinu pretína práve jedna hrana z M , s výnimkou bubleí, ktoré ohraňujú stopku (toto je pre činnosť algoritmu klúčové pozorovanie a čitateľovi odporúčame si ho detailne indukciou dokázať). Ak je medzi stopkami dvoch kvetov plná hrana, jej pridaním do M vznikne činka. Algoritmus skončí, ak sú všetky vrcholy pokryté činkami.



Operácia *presun* na maďarskom strome.

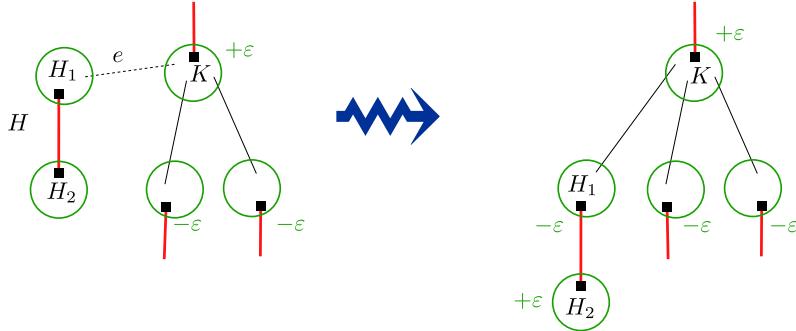
(P1) Zelenej bubline na nepárnej úrovni klesol náboj na 0. Nech K je kvet, ktorému patrí nulová bublina. Z definície kvetu, K obsahuje nepárny počet kvetov K_1, \dots, K_{2r+1} , pričom stopka je vo K_1 . Keďže K je na nepárnej úrovni, má jedného rodiča a jedného syna, ktorý je napojený na stopku. Nech hrana do rodiča ide z kvetu K_t a bez ujmy na všeobecnosti nech t je nepárne. Potom cesta K_1, K_2, \dots, K_t má nepárny počet kvetov a môže nahradit K v strome. Dvojice vrcholov K_{t+1}, K_{t+2} až K_{2r}, K_{2r+1} tvoria činky. Plné hrany medzi K_t a K_{t+1} a K_1, K_{2r+1} sa dajú odobrať z L , lebo v novom strome bude jedna z nich na nepárnej úrovni a druhá v činke, takže pri najbližšej operácii *posun* prestanú byť plné.



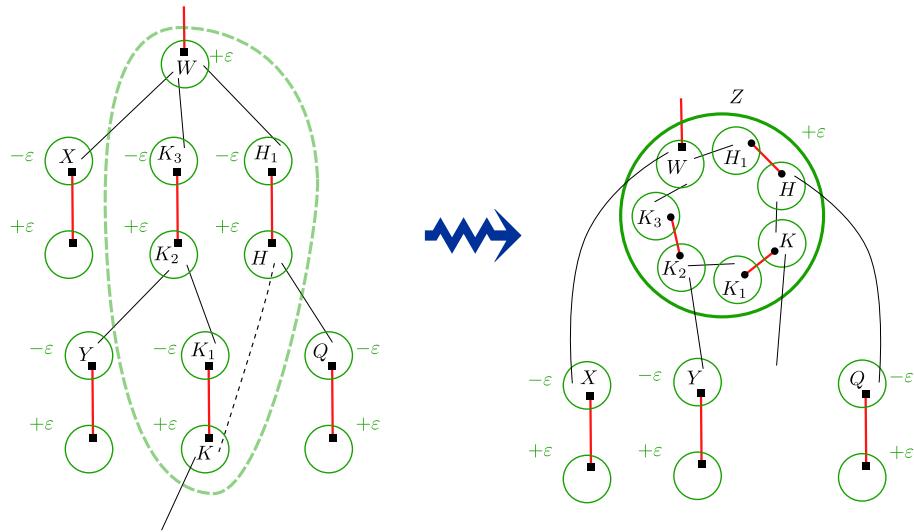
(P2) Naplnila sa hrana e medzi kvetom K na párnej úrovni a činkou H . Nech sa činka H skladá z kvetov H_1 a H_2 tak, že e vedie do nejakého vrchola vo w_1 . Hrana e sa pridá do L a činka H sa pripojí k príslušnému stromu tak, že K (na párnej úrovni) bude mať syna H_1 (na nepárnej úrovni) a ten bude mať jedného syna H_2 (na párnej úrovni).

Volné kvety, ktoré netvoria činky, sú organizované v maďarských stromoch. Kvet na úrovni 0 je koreň, stopka kvetu na úrovni $2h-1$ je spojená hranou z M so stopkou syna na úrovni $2h$. Ak má kvet K na úrovni $2h$ syna H , tak existuje hrana $z L$ medzi nejakým vrcholom z K a nejakým vrcholom z H . Intuitívna predstava je takáto: koreň stromu je kvet K , ktorý by sme chceli zapojiť do činky. Lenže nemôžeme, lebo hrany, ktoré z neho odchádzajú a dali by sa použiť, nie sú plné. Chceli by sme pridať náboj na vonkajšiu bublinu K , ale nemôžeme, lebo nám v tom bránia hrany z L , ktoré vedú do jeho synov.

Algoritmus si v každom momente udržiava sadu maďarských stromov, pričom zvyšok grafu je pokrytý činkami. Algoritmus pracuje v iteráciách, pričom v každej iterácii sa urobí operácia *presun*: všetkým volným kvetom na párnich úrovniach stromov sa k vonkajšej bubline pripočítá ε a od kvetov na nepárnich úrovniach sa ε odpocítia. Hodnota ε sa zvolí maximálna taká, kym sa nenaruší niektorá z podmienok **(I1)**, **(I2)**. To sa môže stať niekoľkými spôsobmi:



(P3) Naplnila sa hrana spájajúca kvety K a H v jednom strome. Zjavne K aj H sú na párnej úrovni. Nech W je najbližší spoločný predok K a H . Kedže W má aspoň dvoch synov, musí byť tiež na párnej úrovni. Nech $K, K_1, \dots, K_{2k+1}, W$ a $H, H_1, \dots, H_{2r+1}, W$ sú cesty v strome. Z parity vrcholov vyplýva, že ich môžeme obalíť novou bublinou a dostaneme kvet Z na párnej úrovni, ktorého stopka je stopka W . Synovia Z budú všetci synovia zahrnutých kvetov – títo ostanú na nepárnej úrovni.



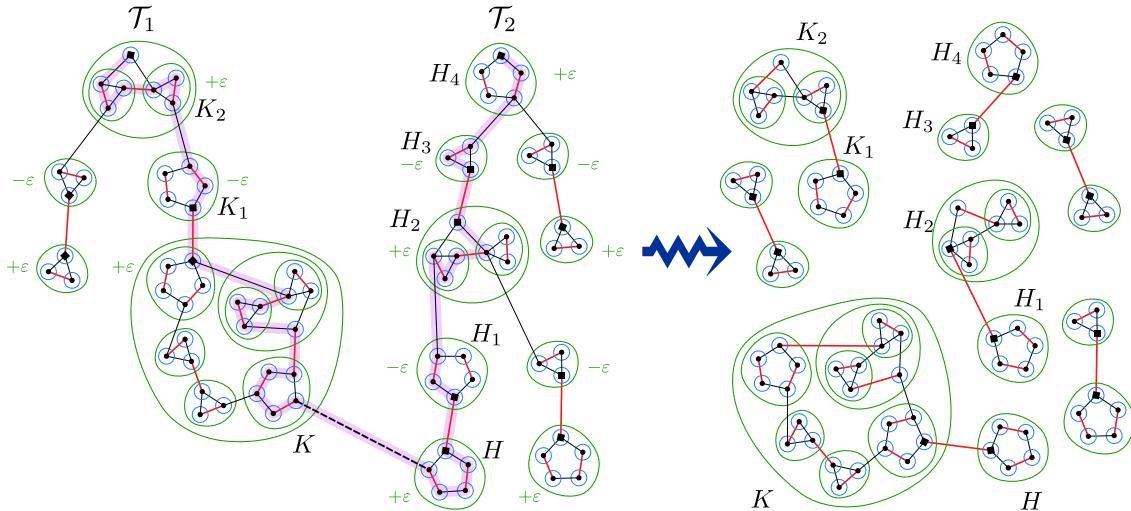
(P4) Naplnila sa hrana e spájajúca kvety K a H v dvoch rôznych stromoch \mathcal{T}_1 a \mathcal{T}_2 . Toto je vlastne jadro celého algoritmu, v ktorom zväčšíme párovanie M . Urobíme to tak, že nájdeme alternujúcu cestu, ktorá spája stopku koreňa stromu \mathcal{T}_1 so stopkou koreňa \mathcal{T}_2 . Kedže sa naplnila hrana, obidva kvety K a H museli byť na párnej úrovni, takže existuje cesta K, K_1, \dots, K_{2r} v strome \mathcal{T}_1 a H, H_1, \dots, H_{2q} v strome \mathcal{T}_2 , kde K_{2r} a H_{2q} sú korene príslušných stromov. Obidve cesty sú tvorené hranami z pôvodného grafu G a striedajú sa v nich hrany z M a L , pričom hrana z M spája stopky susedných kvetov. Aby sme cesty v stromoch mohli doplniť na alternujúce cesty v grafe G , stačí si uvedomiť nasledujúce tvrdenie:

Lema 4.13. Nech K je kvet, u je jeho stopka a v je jeho ľubovoľný vrchol. Potom existuje alternujúca cesta v G z u do v , ktorá je celá obsiahnutá v K a ak je neprázdna, tak začína hranou z L a končí hranou z M .

Dôkaz: Indukciou na hĺbkou vnorenia kvetu. Pre kvet s jedným vrcholom tvrdenie zrejme platí. Nech teda je kvet K tvorený kvetmi K_1, \dots, K_{2r+1} , pričom stopka K je v K_1 . Bez ujmy na všeobecnosti, nech $v \in K_{2t-1}$ pre nejaké t (keby $v \in K_{2t}$, zmeníme smer číslovania kvetov). Ak $t = 1$, použijeme indukciu na kvet K_1 . Nech teda $t > 1$. Z definície kvetu existuje hrana $(q, w) \in L$, kde $q \in K_1$ a $w \in K_2$. Z indukcie existuje alternujúca $u - q$ cesta v K_1 , ktorá končí hranou z M . Zároveň existuje alternujúca cesta v K_2 zo stopky do w , ktorá končí hranou z M . Spojením týchto

ciesť dostaneme alternujúcu cestu z u do stopky K_2 , ktorá končí hranou z L , a teda sa dá predĺžiť do stopky K_3 . Tento postup opakujeme, až kým cestu dostaneme do stopky kvetu K_{2t-1} a odtiaľ opäť z indukčného predpokladu predĺžime cestu do v . \square

S pomocou Lemy 4.13 teraz vieme nájsť alternujúcu cestu medzi stopkou K_{2r} a stopkou H_{2q} . Na tejto ceste vymeníme príslušnosť hrán medzi L a M , čím zvýšime počet hrán v párovaní M . Stromy T_1 a T_2 následne "rozoberieme": kvety K_{2i-1} a K_{2i} (a rovnako H_{2j-1} a H_{2j}) budú po výmene hrán na alternujúcej ceste tvoriť činky. Aby sme videli, prečo, stačí si uvedomiť, že cesta z dôkazu Lemy 4.13 pretína každý vnútorný kvet bud' ani raz, alebo dvakrát a to jednou hranou z M a jednou z L . Preto po výmene hrán z M a L ostane zachovaná podmienka, že každú bublinu pretína práve jedna hrana z M a stopka stromu sa presunie do vrcholu v z Lemy 4.13. Rovnako vytvorí činku kvety K a H . Zvyšné časti stromov tvoria činky prirodzeným spôsobom.



Celý algoritmus pracuje v cykle, v ktorom kým M nie je 1-faktor, nájde najmenšie ε , ktoré poruší niektorú z podmienok **(I1)**, **(I2)** Lemy 4.12. Podľa toho, aká situácia nastala, vykoná jednu z akcií **(P1)**, **(P2)**, **(P3)** alebo **(P4)** a pokračuje v hlavnom cykle. Podmienka **(I3)** ostáva splnená stále. Keď algoritmus skončí a M je 1-faktor, žiadnen kvet nemôže byť koreňom stromu (jeho stopka by bola nespárovaná), preto všetky vrcholy sú pokryté činkami a platí aj podmienka **(I4)**. Preto podľa Lemy 4.12 po skončení algoritmu je M minimálny 1-faktor.

Ostáva nám ukázať, že algoritmus naozaj skončí a, pokiaľ možno aj to, že skončí rýchlo. Klúčové je pri tom nasledovné pozorovanie:

Lema 4.14. *Popísaný algoritmus na riešenie problému MIN-1-FACTOR urobí maximálne $O(n^2)$ iterácií, kde n je počet vrcholov G .*

Dôkaz: Počet hrán v M nikdy neklesá a každé vykonanie akcie **(P4)** ho o jednotku zväčší. Preto sa v celom algoritme vykoná $O(n)$ akcií **(P4)**. Na dôkaz tvrdenia stačí ukázať, že medzi dvoma vykonaniami **(P4)** sa vykoná najviac $O(n)$ akcií **(P1)**, **(P2)** a **(P3)**.

Prvá vec, ktorú si všimneme, je, že v ktoromkoľvek okamihu je v celom algoritme $O(n)$ bublín (vrátane vnorených); to vyplýva z toho, že kvet obsahuje aspoň tri vnútorné kvety, a teda ak si nazveme *hlíbkou* kvety maximálnu úroveň do seba vnorených bublín, tak kedykoľvek môže byť najviac n kvetov hlíbky 0, $n/3$ kvetov hlíbky 1, $n/3^2$ kvetov hlíbky 2, atď., čo dáva geometrický rad. Uvažujme teraz výpočet algoritmu medzi dvoma akciami **(P4)**. Ďalšie dôležité pozorovanie je, že vonkajšej bubline kvetu, ktorý je na párnej úrovni v nejakom strome, nikdy nebude náboj ubúdať (buď ostane na párnej úrovni, alebo sa stane súčasťou inej bubliny na párnej úrovni v akcii **(P3)**). Bublinu, ktorá sa niekedy počas výpočtu vyskytovala ako vonkajšia bublina kvetu na párnej úrovni

budeme volať *bezpečná*. Na dôkaz tvrdenia stačí ukázať, že v akciách **(P1)**, **(P2)** a **(P3)** pribúdajú bezpečné bubliny. Keďže bezpečné bubliny sa nikdy nerozpadnú a všetkých bublín je lineárne veľa, dostaneme, že medzi dvoma akciami **(P4)** môže byť najviac lineárne veľa iterácií algoritmu.

Akcia **(P1)** rozbije bublinu B na nepárnej úrovni a aspoň jedna bublina B' z jej vnútra sa dostane na párnú úroveň. B' ale nemohla byť bezpečná, lebo bezpečná bublina sa nikdy nestane súčasťou bubliny na nepárnej úrovni. Akcia **(P2)** pridá novú bezpečnú bublinu z činky a akcia **(P3)** vyrobí novú bezpečnú bublinu. \square

Implementovať jednu iteráciu je možné priamočiaro v čase $O(nm)$, kde n je počet vrcholov a m je počet hrán: pre každú hranu prejdeme všetky bubliny a zistíme, aké veľké ε nám dovoľuje. Vyberieme hranu s najmenším ohraničením a implementujeme príslušnú akciu. Teda môžeme povedať

Veta 4.15. *Problém MIN-1-FACTOR je riešiteľný v čase $O(n^3m)$.*

Pre hľbavého čitateľa poznamenáme, že s rafinovanejšími dátovými štruktúrami sa výsledný čas dá podstatne zlepšiť; nie je to však cieľom tohto textu.

Relaxované podmienky komplementarity

V predchádzajúcej časti sme ukázali primárno-duálnu metódu založenú na charakterizácii optimálnych riešení pomocou podmienok komplementarity. Pre dvojicu duálnych programov

$$(P) : \min_{\mathbf{x} \in \mathbb{R}^n} \{ \mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq 0 \}$$

$$(D) : \max_{\mathbf{y} \in \mathbb{R}^m} \{ \mathbf{b}^\top \mathbf{y} \mid A^\top \mathbf{y} \leq \mathbf{c}, \mathbf{y} \geq 0 \}$$

sú vektory \mathbf{x} a \mathbf{y} optimálnym riešením (P) a (D) práve vtedy, keď

$$\forall 1 \leq j \leq n : \text{ bud } x_j = 0 \text{ alebo } \sum_{i=1}^m a_{ij} y_i = c_j$$

$$\forall 1 \leq i \leq m : \text{ bud } y_i = 0 \text{ alebo } \sum_{j=1}^n a_{ij} x_j = b_i$$

Môže nám táto charakterizácia pomôcť aj v situácii, keď máme skromnejší cieľ nájsť riešenie blízke optimu? Ukážeme, že keď sú podmienky komplementarity porušené iba "trochu", máme riešenia, ktoré sú "blízko" optimálnym. Vetu 4.8 modifikujeme takto:

Veta 4.16. *Nech \mathbf{x} a \mathbf{y} sú prípustné riešenia úloh (P) a (D) a nech pre nejaké $\alpha, \beta \geq 1$ platí*

$$\forall 1 \leq j \leq n : \text{ bud } x_j = 0 \text{ alebo } c_j/\alpha \leq \sum_{i=1}^m a_{ij} y_i \leq c_j$$

$$\forall 1 \leq i \leq m : \text{ bud } y_i = 0 \text{ alebo } b_i \leq \sum_{j=1}^n a_{ij} x_j \leq \beta b_i$$

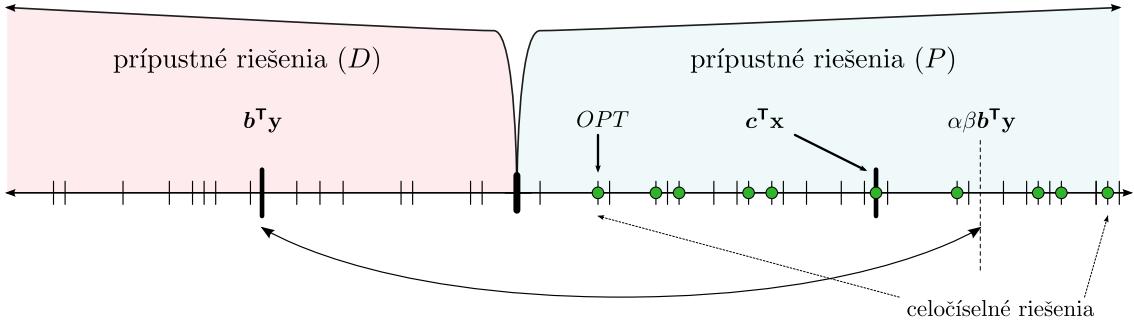
Potom $\mathbf{c}^\top \mathbf{x} \leq \alpha \beta \mathbf{b}^\top \mathbf{y}$.

Dôkaz:

$$\sum_{j=1}^n c_j x_j \leq \sum_{j=1}^n \alpha \left(\sum_{i=1}^m a_{ij} y_i \right) x_j \leq \alpha \sum_{i=1}^m y_i \left(\sum_{j=1}^n a_{ij} x_j \right) \leq \alpha \beta \sum_{i=1}^m y_i b_i$$

\square

Typické použitie vyzerá takto: hľadáme najmenšie celočíselné riešenie (P). Ak hocjakým algoritmom nájdeme celočíselné riešenie \mathbf{x} a k nemu nejaké \mathbf{y} splňajúce podmienky Vety 4.16 budeme mať situáciu



OPT aj nájdené riešenie $c^T \mathbf{x}$ ležia medzi $b^T \mathbf{y}$ a $\alpha\beta b^T \mathbf{y}$, preto nájdené riešenie je nanajvýš $\alpha\beta$ -násobok optima.

Ako konkrétny príklad navrhнемe (opäť) 2-aproximačný algoritmus na problém MIN-VERTEX-COVER (Definícia 3.10). Oproti algoritmu, ktorý sme už navrhli v časti o celočíselných programoch sa bude lísiť tým, že bude omnoho rýchlejší, lebo nebude potrebovať riešiť žiaden relaxovaný lineárny program. Začiatok je rovnaký: chceme riešiť celočíselný program

$$\begin{aligned} & \text{minimalizovať} \quad \sum_{v \in V} \omega_v x_v \\ \text{pri obmedzeniach} \quad & \begin{aligned} x_u + x_v &\geq 1 & \forall e = (u, v) \in E \\ x_v &\geq 0 & \forall v \in V \\ x_v &\in \mathbb{Z} \end{aligned} \end{aligned} \tag{22}$$

urobíme relaxovaný program:

$$\begin{aligned} & \text{minimalizovať} \quad \sum_{v \in V} \omega_v x_v \\ \text{pri obmedzeniach} \quad & \begin{aligned} x_u + x_v &\geq 1 & \forall e = (u, v) \in E \\ x_v &\geq 0 & \forall v \in V \end{aligned} \end{aligned} \tag{23}$$

Teraz ale namiesto toho, aby sme riešili (23), zostrojíme duálny program:

$$\begin{aligned} & \text{maximalizovať} \quad \sum_{e \in E} y_e \\ \text{pri obmedzeniach} \quad & \begin{aligned} \sum_{\substack{e \in E \\ e=(u,v)}} y_e &\leq \omega_u & \forall u \in V \\ y_e &\geq 0 & \forall e \in E \end{aligned} \end{aligned} \tag{59}$$

Kým program (22) vyžaduje, aby sme vybrali vrcholy s najmenšou váhou tak, aby z koncových vrcholov každej hrany bol aspoň jeden vrchol vybratý, program (59) môžeme interpretovať tak, že každej hrane chceme priradiť nezáporný náboj y_e , pričom ω_v bude kapacita vrchola. Cieľom je napumpovať do grafu čo najviac náboja, ale žiaden vrchol sa nesmie preťažiť: súčet nábojov incidentných hrán musí byť najviac kapacita vrchola. Napíšeme si podmienky komplementarity:

$$\mathbf{S1} \quad \forall v \in V : \quad x_v > 0 \Rightarrow \sum_{\substack{e \in E \\ e=(u,v)}} y_e = \omega_u$$

$$\mathbf{S2} \quad \forall e = (u, v) \in E : \quad y_e > 0 \Rightarrow x_u + x_v = 1$$

Z Vety 4.8 vieme povedať toto: keby sme vedeli vybrať množinu vrcholov (t.j. celočíselné hodnoty \mathbf{x}) a dali hranám náboj tak, že žiaden vrchol nie je preťažený (prípustné duálne riešenie), každý

vybratý vrchol je plný (podmienky **S1**) a z každej hrany s nenulovým nábojom je vybratý práve jeden vrchol (podmienky **S2**), mali by sme optimálne riešenie. Zjavne najväčší problém robí splnenie podmienky **S2**. Ak si povieme, že sa o **S2** vôbec nebudeme starať, určite bude platiť $x_u + x_v \leq 2$, preto budú splnené podmienky Vety 4.16 pre $\alpha = 1$ a $\beta = 2$.

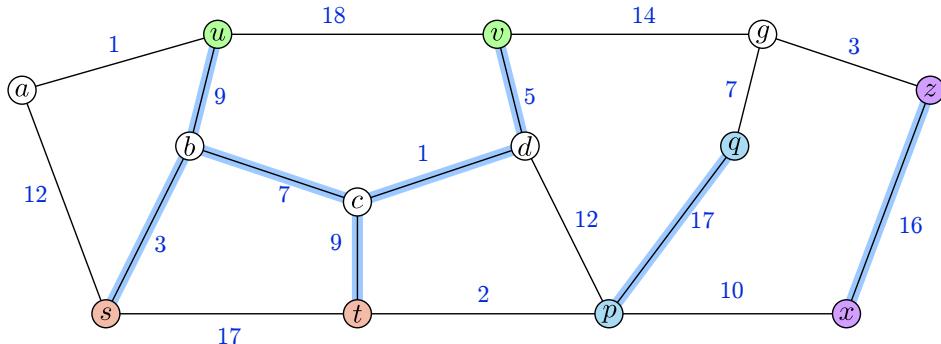
Zostrojíme jednoduchý greedy algoritmus: bude si pamätať množinu vybratých vrcholov C a pre každú hranu e jej aktuálne priradený náboj y_e . Začne s tým, že $C = \emptyset$ a $y_e = 0$ pre všetky e . Algoritmus bude pracovať v cykle: kým C netvorí pokrytie, vyberie ľubovoľnú nepokrytú hranu $e = (u, v)$ a zvýši y_e na maximálnu hodnotu, ktorá nepreťaží vrchol u ani v . Po tejto operácii je aspoň jeden z vrcholov u, v plný (môžu byť aj obidva) a algoritmus ho pridá (ak sú plné obidva, pridá ľubovoľný z nich) do vytváraného pokrycia C .

Po skončení algoritmu platí, že C je pokrytie (hlavný cyklus skončil až keď každá hraná bola pokrytá), žiadny vrchol nie je preťažený a každý vybratý vrchol je plný (obe tieto podmienky sa udržiavajú v platnosti počas celého behu), a preto podľa predchádzajúcich úvah máme garantované, že nájdené riešenie je najviac dvojnásobok optima. Navyše algoritmus pracuje v čase $O(m + n)$, kde n je počet vrcholov a m je počet hrán.

MIN-STEINER-FOREST

V tejto časti si ukážeme, ako využiť relaxované podmienky komplementarity aj v prípade, keď sú aj predpoklady Vety 4.16 príliš silné. Zoberme si nasledovný zjednodušený modelový príklad: máme daný neorientovaný graf, ktorý reprezentuje železničnú sieť medzi mestami: vrcholy zodpovedajú mestám a hrany železničným tratiam. Dopravca, ktorý chce poskytovať prepravné služby, si musí prenajať trate od vlastníka, pričom pre každú trať je daná cena za prenájom. Dopravca sa snaží prenajať trate tak, aby mohol splniť všetky svoje plánované spojenia a pritom zaplatil čo najmenej. Formálnejšie zapísané, chceme riešiť nasledovný problém:

Definícia 4.17. Majme daný graf $G = (V, E)$ a ceny hrán $\omega : E \mapsto \mathbb{R}^+$. Ďalej je daná (symetrická) funkcia požadovaných spojení $r : V \times V \mapsto \{0, 1\}$. Cielom problému MIN-STEINER-FOREST je vybrať množinu hrán $F \subseteq E$ tak, aby pre každú dvojicu vrcholov u, v takú, že $r(u, v) = 1$, existovala $u-v$ cesta používajúca iba hrany z F . Navyše požadujeme, aby celková cena hrán v F bola najmenšia možná.



Príklad grafu s váhami hrán (modré). Požadované prepojenia sú $u - v$, $s - t$, $p - q$ a $x - z$ (t.j. $r(u, v) = r(v, u) = r(s, t) = r(t, s) = r(p, q) = r(q, p) = r(x, z) = r(z, x) = 1$ a ostatné hodnoty $r(\cdot, \cdot)$ sú 0). Zvýraznené hrany tvoria optimálne riešenie, ktoré má cenu 51 a je tvorené tromi stromami (je dobré si uvedomiť, že komponenty súvislosti optimálneho riešenia budú vždy stromy).

Pre čitateľa oboznámeného s NP -úplnými problémami je jednoduchým cvičením ukázať, že problém MIN-STEINER-FOREST je NP -tažký, preto ani nebude očakávať, že prezentujeme algoritmus, ktorý ho optimálne rieši. Ukážeme 2-aproximačný algoritmus, t.j. ukážeme, že riešenie, ktoré algoritmus vráti nebude nikdy viac ako dvojnásobok optima. Začneme tradične – problém formulujeme ako celočíselný lineárny program. Celkom prirodzene pre každú hranu $e \in E$ zavedieme premennú $x_e \in \{0, 1\}$, ktorá udáva, či je e vybratá do riešenia. Potrebujeme ešte sformulovať požiadavky na existenciu spojenia formou lineárnych ohraničení. Keďže sa chystáme použiť primárno-duálnu metódu, neostýchame sa použiť exponenciálne vela obmedzení. Nejakú množinu $S \subseteq V$ nazveme *hladná*, ak z nej musí vo výslednom riešení odchádzať nejaká hrana, t.j. ak existuje $u \in S, v \in V \setminus S$, kde $r(u, v) = 1$. Naša formulácia bude založená na tomto pozorovaní:

Lema 4.18. *Ak z každej hladnej množiny odchádza aspoň jedna vybratá hrana, potom vybraté hrany tvoria prípustné riešenie.*

Dôkaz: Zoberme si ľubovoľné $u, v \in V$, také, že $r(u, v) = 1$. Treba nájsť $u - v$ cestu z vybratých hrán. Budeme indukciou konštruovať postupnosť množín $\{u\} = S_0 \subseteq S_1 \subseteq \dots$ s tou vlastnosťou, že pre každý vrchol $w \in S_i$ existuje $u - w$ cesta z vybratých hrán. Pre $\{u\} = S_0$ to zjavne platí. Zoberme si teraz nejakú množinu S_i . Ak $v \in S_i$, máme $u - v$ cestu z vybratých hrán a skončíme dôkaz. Ak nie, S_i je hladná a teda z nej odchádza vybratá hrana do nejakého vrchola $w \in V \setminus S$. Zoberme $S_{i+1} := S \cup \{w\}$ a vlastnosť, že z u existuje cesta z vybratých hrán do každého vrchola z S_{i+1} , ostane zachovaná. Keďže vrcholov je n , a v každom kroku pridávame do S_i jeden vrchol, časom musíme prísť do situácie, že $v \in S_i$. \square

Označme $f(S) = 1$, ak S je hladná a $f(S) = 0$ inak. Podľa Definície 4.11 označme $\delta(S)$ hranovú hranicu množiny S . Problém MIN-STEINER-FOREST môžeme zapísť nasledovným celočíselným programom:

$$\begin{aligned} & \text{minimalizovať} \quad \sum_{e \in E} \omega_e x_e \\ & \text{pri obmedzeniach} \quad \begin{aligned} \sum_{e \in \delta(S)} x_e &\geq f(S) \quad \forall S \subseteq V \\ x_e &\in \{0, 1\} \quad \forall e \in E \end{aligned} \end{aligned} \tag{60}$$

ktorý následne relaxujeme tak, že $x_e \geq 0$; podmienka $x_e \leq 1$ je splnená v každom minimálnom riešení, lebo nemá zmysel zobrať $x_e > 1$. Dostaneme tak

$$\begin{aligned} & \text{minimalizovať} \quad \sum_{e \in E} \omega_e x_e \\ & \text{pri obmedzeniach} \quad \begin{aligned} \sum_{e \in \delta(S)} x_e &\geq f(S) \quad \forall S \subseteq V \\ x_e &\geq 0 \quad \forall e \in E \end{aligned} \end{aligned} \tag{61}$$

K programu (61) zostrojíme štandardným spôsobom duálny program: budeme mať premennú y_S pre každú množinu $S \subseteq V$ a zapíšeme

$$\begin{aligned} & \text{maximalizovať} \quad \sum_{S \subseteq V} y_S f(S) \\ & \text{pri obmedzeniach} \quad \begin{aligned} \sum_{S: e \in \delta(S)} y_S &\leq \omega_e \quad \forall e \in E \\ y_S &\geq 0 \quad \forall S \subseteq V \end{aligned} \end{aligned} \tag{62}$$

Programy s rovnakou štruktúrou sme tu už stretli niekoľkokrát a tak interpretácia bude celkom prirodzená: okolo každej množiny je bublina s nábojom. Cieľom je maximalizovať náboj na hladných množinách (množiny, ktoré nie sú hladné majú nulový vklad do účelovej funkcie, a tak im nikdy nebudeme žiaden náboj pridelovať) tak, aby sme žiadnu hranu nepretažili: súčet nábojov na bublinách, ktoré danú hranu pretínajú, musí byť najviac kapacita hrany. Podotýkame, že zvýšiť hodnotu y_S je to isté, ako zvýšiť hodnotu $y_{V \setminus S}$ (lebo pretína tie isté hrany). Napišme si ešte podmienky komplementarity:

$$\mathbf{S1} \quad \forall e \in E : \quad x_e > 0 \Rightarrow \sum_{S:e \in \delta(S)} y_S = \omega_e$$

$$\mathbf{S2} \quad \forall S \subseteq V : \quad y_S > 0 \Rightarrow \sum_{e \in \delta(S)} x_e = f(S)$$

Z podmienok komplementarity vidíme, že ak by existovalo optimálne celočíselné riešenie, tak každá vybratá hrana by bola plná a z každej nenulovej bubliny okolo hladnej množiny by odchádzala práve jedna hrana (z bublín okolo množín, ktoré nie sú hladné, nejde nič).

Algoritmus, ktorý zostrojíme, bude postupne vyberať hrany, až kým nebudú splnené všetky požiadavky na spojenia. Zároveň sa bude udržiavať invariant, že všetky vybrané hrany sú plné a žiadna hrana nie je preplnená. Preto keď algoritmus skončí, bude mať prípustné riešenie programu (61) a prípustné riešenie programu (62), ktoré zároveň spĺňajú podmienky **S1**. Keby sme chceli použiť Vetu 4.16 na dosiahnutie 2-aproximačného algoritmu, potrebovali by sme navyše zaručiť splnenie podmienok

$$\mathbf{S2'} \quad \forall S \subseteq V : \quad y_S > 0 \Rightarrow \sum_{e \in \delta(S)} x_e \leq 2f(S),$$

t.j. z každej nenulovej bubliny (okolo hladnej množiny) odchádzajú najviac dve vybrané hrany. Toto nebudeme schopní zaručiť, ale ako uvidíme neskôr, bude stačiť slabšie tvrdenie, že *v priemere* z nenulovej bubliny odchádzajú najviac dve vybrané hrany.

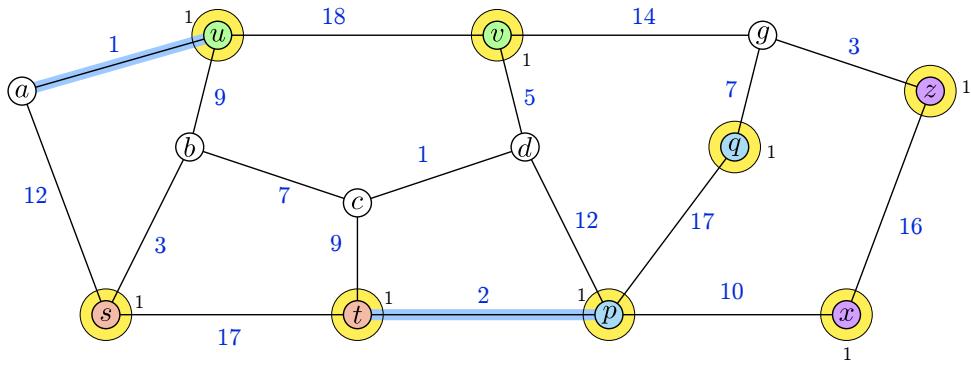
Štruktúra algoritmu bude podobná ako pri algoritme pre MIN-VERTEX-COVER z minulej časti: začne s prázdnou množinou hrán a nulovými bublinami. Kým ale algoritmus pre MIN-VERTEX-COVER v jednej iterácii vybral jednu duálnu premennú, zväčšil ju kolko sa dalo a z naplnených vrcholov vybral jeden, teraz algoritmus v jednej iterácii zvýši veľa duálnych premenných naraz a z naplnených hrán vyberie jednu do riešenia.

Aké duálne premenné chceme zväčšovať? Určite musia zodpovedať hladným množinám (dohodli sme sa, že množinám, ktoré nie sú hladné, žiadnená náboj nedávame). Zároveň, pretože všetky vybrané hrany sú plné, bublinu okolo hladnej množiny, z ktorej už odchádzajú vybrané hrany, nemôžeme zväčšiť. Zväčšovať teda môžeme bubliny okolo množín, ktoré sú *nespokojné*: sú hladné (t.j. vo výslednom riešení z nich musí odchádzať nejaká hrana), ale zatiaľ z nich nijaká vybraná hrana neodchádzajú (inými slovami, sú to množiny, ktoré porušujú prípustnosť programu (61)). Nespokojných množín môže byť potenciálne veľa, ale jednoduché pozorovanie nám povie, že

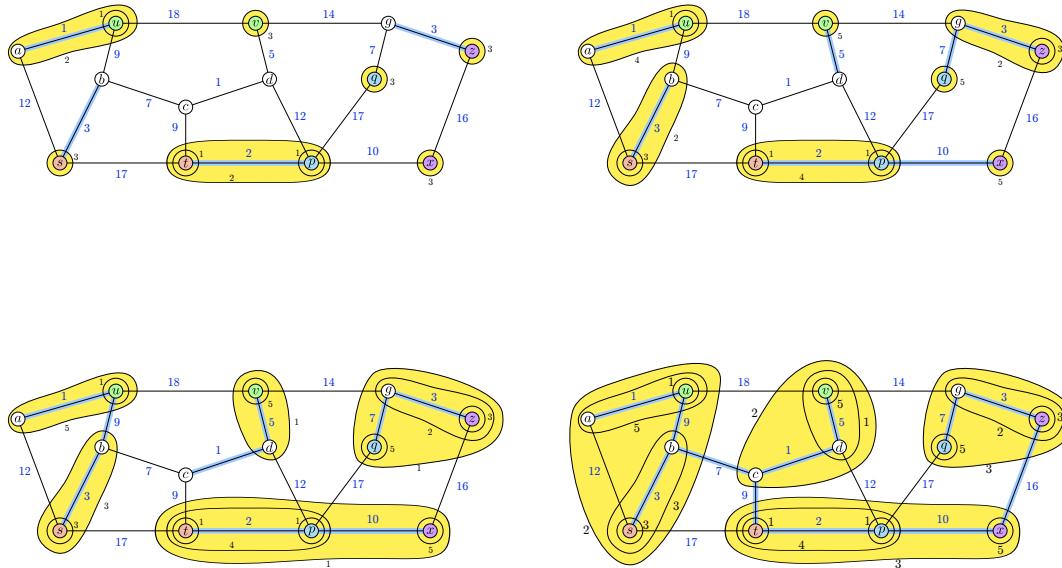
Lema 4.19. *Nespokojné množiny, ktoré sú minimálne vzhľadom na inklúziu, sú komponenty súvislosti podgrafa indukovaného doteraz vybratými hranami.*

Dôkaz: Množina je nespokojná, keď je hladná a neodchádzajú z nej vybraná hrana. Preto každá nespokojná množina je zjednotenie nejakých komponentov súvislosti grafu, indukovaného vybranými hranami. \square

Algoritmus bude v každom kroku zväčšovať duálne premenné, ktoré zodpovedajú nespokojným komponentom súvislosti. Na začiatku nie sú vybrané žiadne hrany, a teda nespokojné komponenty súvislosti sú jednoprvkové množiny okolo vrcholov, ktoré potrebujú byť spojené. Pre zadanie z úvodného príkladu sú to množiny $\{u\}, \{v\}, \{s\}, \{t\}, \{p\}, \{q\}, \{x\}, \{z\}$. Algoritmus zväčšuje príslušné duálne premenné, až kým sa niektoré hrany nenaaplnia. V našom prípade sa duálne premenné zvýšia na 1 a naplnia sa hrany $\{a, u\}$ a $\{t, p\}$.

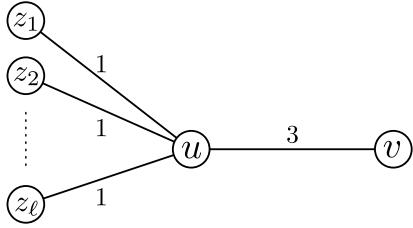


V nasledujúcej iterácii sú nespokojné komponenty $\{a, u\}, \{s\}, \{v\}, \{t, p\}, \{q\}, \{z\}, \{x\}$ a algoritmus zvýši ich hodnoty o 2, čím sa naplnia hrany $\{s, b\}$ a $\{g, z\}$, vzniknú nové nespokojné komponenty a výpočet pokračuje ako na nasledovných obrázkoch.



Algoritmus skončí, keď sú všetky komponenty súvislosti, tvorené vybranými hranami, spokojné. Z technických dôvodov nám bude príjemnejšie uvažovať, že v jednej iterácii sa vyberie iba jedna z naplnených hrán. Ak sa naplnilo viacero hrán, nasledujúce iterácie zvýšia duálne premenné o nulovú hodnotu a postupne popridávajú potrebné hrany.

Takto popísaný algoritmus má ešte jeden problém, a to, že nefunguje. Ako príklad zoberme nasledovný graf, v ktorom je jediná požiadavka $r(u, v) = 1$.



Na začiatku sú nespokojné komponenty $\{u\}$ a $\{v\}$ a algoritmus dá každému z nich hodnotu 1. Tým sa ale zaplnia hrany $\{z_i, u\}$, takže algoritmus na koniec vyberie všetky hrany, a teda vyprodukuje riešenie s cenou $\ell + 3$, ktoré ale nie je dobrá aproximácia optima s cenou 3. Pokúsime sa násť algoritmus zachrániť priamočiarou úpravou: po skončení algoritmu získané riešenie upravíme tak, že odstránieme prebytočné hrany,

t.j. hrany, po ktorých odobratí ostatné všetky požiadavky na spojenia splnené. Výsledný algoritmus bude vyzerať takto:

-
- 1 $F := \emptyset, y_{\{v\}} := 0$ pre všetky $v \in V$
 - 2 kým existuje nespokojný komponent súvislosti indukovaný hranami z F
 - 3 zvýš hodnoty y_S pre všetky množiny S zodpovedajúce nespokojným komponentom súvislosti z F tak, aby sa naplnila nejaká hrana e
 - 4 $F := F \cup e$
 - 5 $F' := F$
 - 6 pre každú hranu $e \in F$
 - 7 ak $F - \{e\}$ je prípustné riešenie, $F' := F' - \{e\}$
-

Najprv sa presvedčíme, že algoritmus je korektný:

Lema 4.20. Po skončení algoritmu tvoria hrany F' prípustné riešenie programu (60) a hodnoty y tvoria prípustné riešenie programu (62).

Dôkaz: Po skončení cyklu na riadku 2 neexistuje nespokojný komponent súvislosti tvorený hranami F . Pretože každá nespokojná množina je zjednotením komponentov súvislosti tvorených F , z každej hladnej množiny odchádza vybratá hrana, a teda F je prípustné riešenie (60). Ešte sa treba presvedčiť, že riadky 6 a 7 túto vlastnosť neporušia. Hranu $e \in F$, pre ktorú platí, že $F - \{e\}$ je prípustné riešenie, nazveme *zbytočná*. Ukážeme, že keď z F vyhodíme naraz všetky zbytočné hrany, dostaneme prípustné riešenie F' . Na to si treba uvedomiť, že hrany F indukujú acylický graf: na riadku 3 vždy zvyšujeme hodnoty pre komponenty súvislosti, preto sa nikdy nenaplní hrana vo vnútri komponentu; každá naplnená hrana spája dva komponenty súvislosti, takže nevytvorí cyklus. Preto ak máme dva vrcholy u, v , ktoré musia byť spojené, t.j. $r(u, v) = 1$, je v grafe indukovanom F práve jedna cesta z u do v , a všetky jej hrany sa preto ocitnú v F' .

Na druhej strane, hodnoty y_S sme vždy zvyšovali tak, aby sa žiadna hrana nepreplnila, takže počas celého algoritmu tvoria prípustné riešenie (62). \square

Vidíme, že po skončení algoritmu máme nejaké riešenie F' , a nejaké riešenie y duálneho programu. Na to, aby sme ohraničili approximačný pomer, potrebujeme porovnať hodnotu riešenia F' s hodnotou optima. Tú nepoznáme, ale vieme, že je určite väčšia (alebo rovná) ako hodnotu optima. Preto ak chceme dokázať, že algoritmus vždy vráti najviac dvojnásobok optima, stačí dokázať, že hodnota riešenia F' je najviac dvojnásobok hodnoty y , t.j.

Veta 4.21.

$$\sum_{e \in F'} \omega_e \leq 2 \sum_{S \subseteq V} y_S f(S)$$

Dôkaz: Pretože sa nikdy nezvyšovali hodnoty y_S pre množinu S , ktorá nie je hladná, $f(S) = 0$ implikuje $y_S = 0$, a teda chceme dokázať

$$\sum_{e \in F'} \omega_e \leq 2 \sum_{S \subseteq V} y_S.$$

Zavedme si teraz nasledovné označenie: pre nejaké množiny $W \subseteq E$ a $S \subseteq V$ označme $\deg_W(S) := |W \cap \delta(S)|$, t.j. počet hrán z W , ktoré majú jeden koniec v S a druhý mimo S . Každá hrana, ktorú sme vybrali do F (a teda aj do F') je plná (hodnoty y nikdy neznižujeme, teda ak hrana raz bola plná, je plná aj na konci), a preto na ľavej strane platí

$$\sum_{e \in F'} \omega_e = \sum_{e \in F'} \left(\sum_{S: e \in \delta(S)} y_S \right) = \sum_{S \subseteq V} \deg_{F'}(S) y_S.$$

Potrebujeme teda dokázať

$$\sum_{S \subseteq V} \deg_{F'}(S) y_S \leq 2 \sum_{S \subseteq V} y_S \quad (63)$$

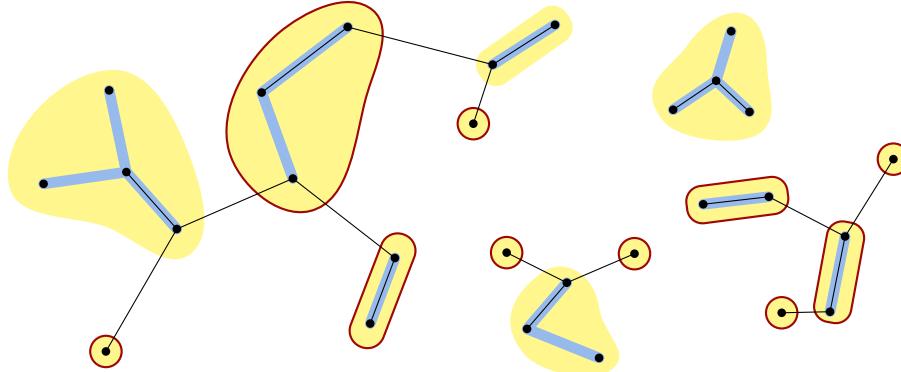
Ukážeme to indukciou na počet iterácií algoritmu. Na začiatku sú všetky hodnoty $y_S = 0$, takže (63) triviálne platí. Uvažujme teraz iteráciu ℓ , v ktorej sa na riadku 3 zvýšili y_S pre všetky nespokojné komponenty o hodnotu Δ . Ako sa zmení (63)? Pre každý nespokojný komponent S pribudne na ľavej strane $\Delta \deg_{F'}(S)$ a na pravej 2Δ . Aby sme dokázali (63), ukážeme, že na pravej strane pribudlo viac ako na ľavej, t.j.

$$\Delta \left(\sum_{S \in \mathcal{S}_\ell} \deg_{F'}(S) \right) \leq 2\Delta |\mathcal{S}_\ell|, \quad (64)$$

kde \mathcal{S}_ℓ je množina všetkých nespokojných komponentov v tejto iterácii. Nerovnosť (64) sa dá napísat ako

$$\frac{\sum_{S \in \mathcal{S}_\ell} \deg_{F'}(S)}{|\mathcal{S}_\ell|} \leq 2,$$

t.j. potrebujeme ukázať, že priemerný stupeň nespokojného komponentu vzhľadom na F' je najvyšší 2. Označme F_ℓ množinu algoritmom vybratých hrán na začiatku iterácie ℓ . Situácia na začiatku iterácie vyzerá ako na obrázku:



Čierne hrany sú výsledné riešenie F' . Modré hrany sú algoritmom doteraz vybraté hrany F_ℓ . Zvýraznené komponenty súvislosti F_ℓ sú nespokojné.

Hrany F' aj F_ℓ sú podmnožiny hrán F . Pretože F indukujú les (viď. dôkaz Lemy 4.20), aj $F' \cup F_\ell$ tvoria les. Ak skontrahujeme každý komponent súvislosti F_ℓ do jedného vrchola (žlté oblasti v predchádzajúcom obrázku), dostaneme les H , ktorého vrcholy zodpovedajú komponentom súvislosti F_ℓ a hrany sú tvorené hranami F' . Dokázať nerovnosť (64) znamená ukázať, že priemerný stupeň vrchola v H je nanajvýš dva, pričom priemer je braný cez vrcholy, ktoré zodpovedajú nespokojným komponentom. Keby sa priemer bral cez všetky vrcholy, mohli by sme dôkaz skončiť: les s n vrcholmi má najviac $n - 1$ hrán, takže súčet stupňov vrcholov je najviac $2(n - 1)$, a preto priemerný stupeň je menší ako 2. Na to, aby sme ohraničili priemerný stupeň nespokojných komponentov ukážeme, že spokojné komponenty nemajú stupeň 1: buď sú izolované, alebo majú stupeň aspoň 2. Keď to dokážeme, dôkaz dokončíme takto: žiadnený nespokojný komponent nemá v H stupeň 0, pretože z F' z neho odchádza aspoň jedna hrana. Izolované vrcholy v H preto všetky zodpovedajú spokojným komponentom. Keď z H vyhodíme izolované vrcholy, dostaneme nový les H' : stačí nám ukázať, že priemerný stupeň nespokojných komponentov v H' je nanajvýš 2. Lenže priemerný stupeň všetkých komponentov je nanajvýš 2 a každý spokojný komponent má stupeň aspoň 2, takže priemerný stupeň nespokojných komponentov nemôže byť väčší ako 2.

Na záver ukážeme, že spokojný komponent nemôže mať v H stupeň 1. Predpokladajme sporom, že existuje komponent súvislosti C v F_ℓ , z ktorého v F' odchádza práve jedna hrana e . Hrana e sa do F' dostala preto, lebo nie je zbytočná, t.j. leží na jedinej ceste, ktorá v F spája dva vrcholy u a v , kde $r(u, v) = 1$. Lenže ak e je jediná hrana, ktorá v F' odchádza z C , potom vrcholy u a v musia ležať jeden v C a druhý mimo C , a preto C nemôže byť spokojný v F_ℓ . \square

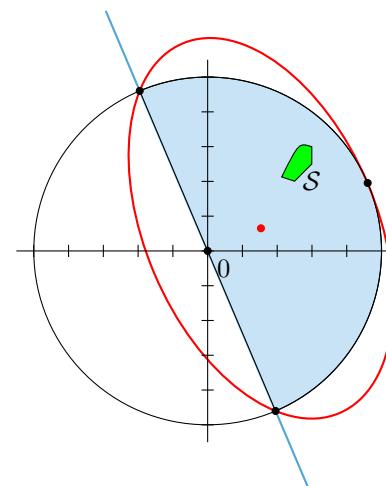
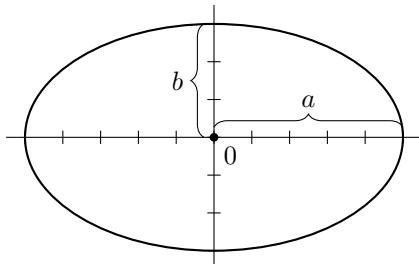
5

Elipsoidová metóda

V predchádzajúcich častiach sme si ukázali simplexovú metódu na riešenie úlohy lineárneho programovania. Povedali sme o nej, že je "efektívna", aj keď počet iterácií môže byť v najhoršom prípade exponenciálny od počtu premenných a obmedzení. V tejto kapitole si ukážeme inú metódu na riešenie lineárnych programov, ktorá bude polynomálna od veľkosti vstupu (viac o tom neskôr). Hlavný dôvod, pre ktorý ju tu uvádzame, však nie je jej polynomialita, ale iné zaujímavé vlastnosti, ktoré takisto uvidíme neskôr. Začnime zdanivo nesúvisiacou geometrickou úlohou:

Definícia 5.1. V rovine je daný konvexný útvar S , ktorý celý leží v jednotkovom kruhu a má plochu aspoň ε . Cielom problému 2D-MEMBERSHIP je nájsť nejaký bod z S .

Skúsme problém 2D-MEMBERSHIP riešiť zovšeobecneným binárnym vyhľadávaním: začneme s jednotkovým kruhom. Ak jeho stred leží v S , násli sme bod z S . Inak z konvexnosti S vieme, že existuje priamka prechádzajúca cez stred, ktorá rozdelí kruh na dve polovice tak, že S je v jednej z nich (na obrázku vpravo modrý polkruh). Chceli by sme rekurzívne pokračovať v hľadaní iba v modrom polkruhu; problém je ale v tom, že postupne po veľa deleniach budeme dostávať stále zložitejšie útvary, ktoré si bude treba pamätať. Ukáže sa, že dobrým riešením je zapamätať si najmenšiu elipsu opísanú modrému polkruhu (červená elipsa vpravo): je sice trocha väčšia, ako polkruh, ale (ako ukážeme) stále menšia, ako pôvodný kruh. Keďže S leží celý vovnútri elipsy, môžeme našu úvahu zopakovať: ak stred elipsy leží v S , máme riešenie, inak opäť existuje priamka prechádzajúca stredom elipsy tak, že S je v jednej polovici; môžeme nájsť najmenšiu elipsu opísanú príslušnej "poleipse" a taktto pokračovať ďalej. Pripomeňme, že elipsa E so stredom v bode $(0, 0)$ a jej plocha $\|E\|$ sú dané nasledovnými vzťahmi, pričom a, b sú dĺžky jednotlivých poloosí:



$$E = \left\{ (x, y) \in \mathbb{R}^2 \mid \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1 \right\}$$

$$\|E\| = \pi ab$$

Aby náš postup zafungoval, musíme vedieť nájsť najmenšiu elipsu opísanú generickej poleipse a ukázať, že jej plocha je o konštantný faktor menšia. Majme teda elipsu v ľubovoľnej polohe a priamku prechádzajúcu jej stredom. Začneme tým, že elipsu presunieme do bodu $(0, 0)$ a otočíme tak, aby jej osi boli rovnobežné so súradnicovými osami.

Literatúra

- [1] G. Ausiello. *Complexity and approximation: combinatorial optimization problems and their approximability properties*. Springer, 1999.
- [2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.
- [3] R. G. Bland. A combinatorial abstraction of linear programming. *J. Comb. Theory, Ser. B*, 23(1):33–57, 1977.
- [4] V. Chvátal. *Linear Programming*. Series of Books in the Mathematical Sciences. W.H. Freeman, 1983.
- [5] D. Emanuel and A. Fiat. Correlation clustering - minimizing disagreements on arbitrary weighted graphs. In G. D. Battista and U. Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 2003.
- [6] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Algorithms and combinatorics. Springer-Verlag, 1988.
- [7] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182:105–142, 1999.
- [8] V. Klee and G. J. Minty. How Good is the Simplex Algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press Inc., New York, 1972.
- [9] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. Snps problems, complexity, and algorithms. In F. Meyer auf der Heide, editor, *ESA*, volume 2161 of *Lecture Notes in Computer Science*, pages 182–193. Springer, 2001.
- [10] A. Lubotzky. *Discrete Groups, Expanding Graphs and Invariant Measures*. Progress in Mathematics. Birkhäuser, 1994.
- [11] J. Matoušek and B. Gärtner. *Understanding and Using Linear Programming*. Universitext (En ligne). Springer, 2007.
- [12] A. Sankar, D. A. Spielman, and S.-H. Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM J. Matrix Analysis Applications*, 28(2):446–476, 2006.
- [13] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004.
- [14] V. V. Vazirani. *Approximation Algorithms*. Springer, Mar. 2004.
- [15] D. Williamson and D. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.