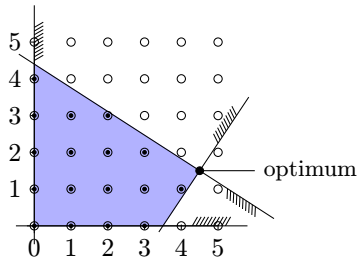
2 Zaokrúhľovanie lineárnych programov

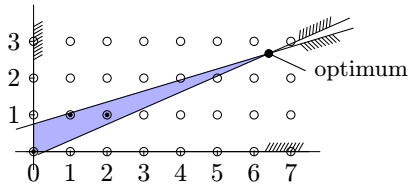
2.1 Celočíselné programy a relaxácia: dobrý, zlý a škaredý

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. V skutočnosti problémy, kde vystupujú veličiny s ľubovoľnou (alebo aspoň dostatočne veľkou) presnosťou sú skôr výnimkou.



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 ďaleko 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 2.1. Celočíselný lineárny program *(ILP) v normálnom tvare je linárny program*

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

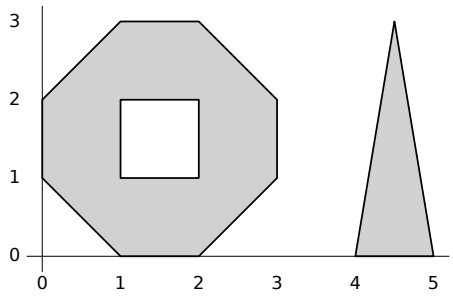
s dodatočnými obmedzeniami tvaru $x_i \in \mathbb{Z}$.

Skúseného čitateľa isto neprekvapí, že celočíselné obmedzenia podstatne zvyšujú vyjadrovaciu schopnosť lineárnych programov. Pri formulovaní zložitejších vzťahov pomocou ILP hrajú dôležitú

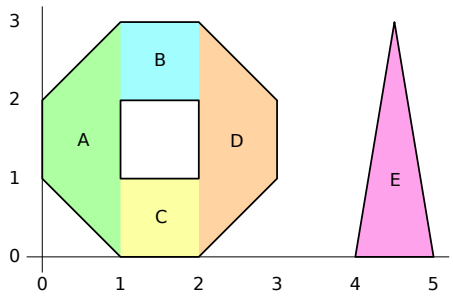
úlohu tzv. *indikátorové premenné*: sada premenných $\delta_1, \ldots, \delta_k \in \mathbb{Z}$ s obmedzeniami

$$\delta_1 + \delta_2 + \cdots + \delta_k = 1$$
$$\delta_i \geq 0 \text{ pre } i = 1 \ldots k$$

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



Množinu $\mathcal{D}$ môžme rozdeliť na 5 častí



z ktorých každú vieme reprezentovať lineárnymi obmedzeniami:

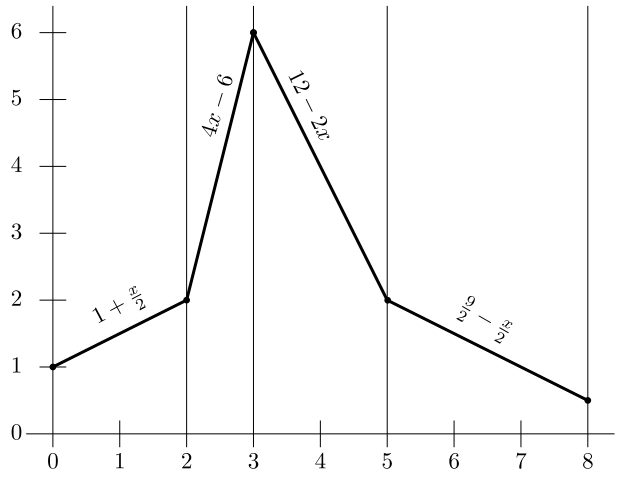
$x \geq 0$	$x \geq 1$	$x \geq 1$	$x \geq 2$	$x \geq 4$
$x \leq 1$	$x \leq 2$	$x \leq 2$	$x \leq 3$	$x \leq 5$
$y \geq 1 - x$	$y \geq 2$	$y \geq 0$	$y \geq x - 2$	$y \leq 6x - 24$
$y \leq 2 + x$	$y \leq 3$	$y \leq 1$	$y \leq 5 - x$	$y \leq -6x + 30$
				$y \geq 0$

Zavedieme indikátorové premenné $\delta_1, \ldots, \delta_5 \in \mathbb{Z}$ 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

$x \geq 0$	$x \geq \delta_2$	$x \geq \delta_3$	$x \geq 2\delta_4$	$x \geq 4\delta_5$
$x \leq 5 - 4\delta_1$	$x \leq 5 - 3\delta_2$	$x \leq 5 - 3\delta_3$	$x \leq 5 - 2\delta_4$	$x \leq 5$
$y \geq \delta_1 - x$	$y \geq 2\delta_2$	$y \geq 0$	$y \geq -5 + 3\delta_4 + x$	$y \leq 3 - 27\delta_5 + 6x$
$y \leq 3 - \delta_1 + x$	$y \leq 3$	$y \leq 3 - 4\delta_3$	$y \leq 8 - 3\delta_4 - x$	$y \leq 33 - 3\delta_5 - 6x$
				$y \geq 0$

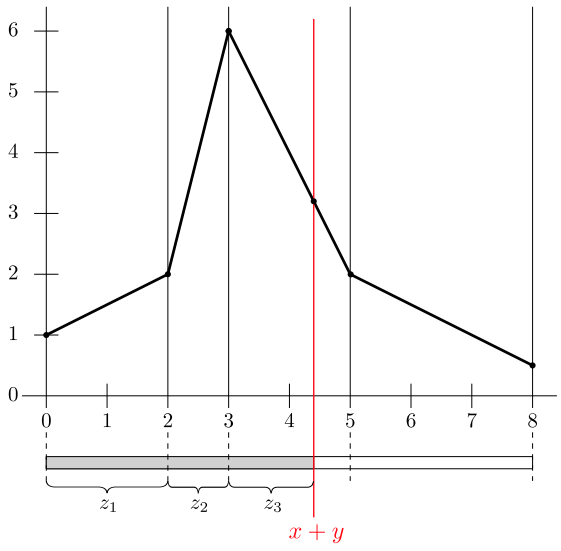
$$\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 = 1$$
$$\delta_i \geq 0, \delta_i \in \mathbb{Z} \text{ pre } i = 1 \ldots 5$$

Inou v praxi dôležitou vlastnosťou celočíselných lineárnych programov je schopnosť otpimalizovať nelineárne účelové 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_2, z_3, z_4, ktoré udávajú, aká časť príslušného intervalu je "zaplnená" hodnotou $x + y$:

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



Potrebujeme 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} nenulová. 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 premenným $z_1, \ldots, z_4$ ľahko vyjadríme účelovú funkciu

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

Prezentované príklady snáď utvrdili čitateľovu intuíciu, že pomocou ILP sa dajú zapísať oveľa zložitejšie problémy, ako pomocou lineárnych programov, a teda aj riešenie ILP by malo byť oveľa ťažšie. Ukážeme si, že tomu tak naozaj je: už aj úloha zistiť, 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 2.2. *Majme n logických premenných $x_1, \ldots, x_n \in \{true, false\}$.* Literál *je premenná x_i alebo jej negácia $\overline{x_i}$.* Klauzula *je dizjunkcia literálov $C = l_1 \vee l_2 \vee \cdots \vee l_{k_C}$. Formula* v konjunktívnej normálne forme *(CNF) je konjunkcia klauzúl $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Problém* SAT *je rozhodovací problém, v ktorom je na vstupe daná formula F v CNF a úlohou je zistiť, či existuje ohodnotenie premenných $x_1, \ldots, x_n$ 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 popísať, ako pracuje `A`'. Nech vstupná formula F pozostáva z klauzúl $C_1, \ldots, C_m$. 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 klauzulu $C_i = (x_1 \vee \overline{x_2} \vee x_3)$ 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 $0 \leq x_i \leq 1$. Spĺňajúce ohodnotenie musí spĺňať 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

$$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3)$$

dostaneme obmedzenia

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

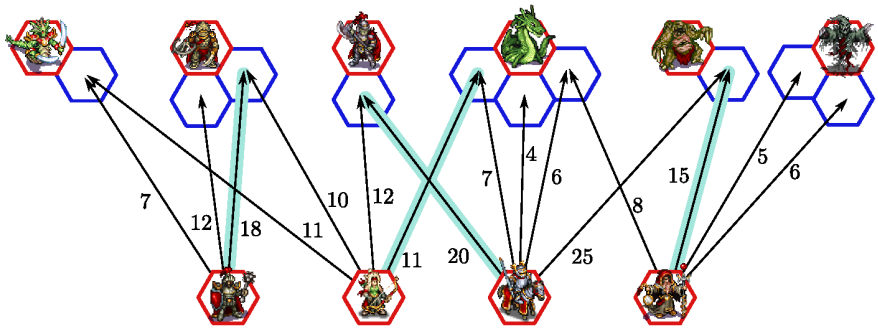
Zjavne každé splňujúce ohodnotenie F je prípustným riešením I a naopak. Preto F je splniteľná práve vtedy, ak existuje prípustné riešenie I.

Cvičenie. *Skúste iné známe NP-ťažké problémy (napr. problém obchodného cestujúceho, nájdenie 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 ťahu má k dispozícii niekoľko figúr, z ktorých každá môže potenciálne zaútočiť na niektorú nepriateľskú figúru (najprv sa musí presunúť na niektoré priľahlé 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.

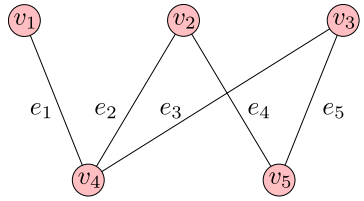
Problém, ktorý riešime, je známy ako problém najväčšieho bipartitného párovania:

Definícia 2.3. *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 hrana vybratá. 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 vybratá najviac jedna hrana. Celý ILP potom vyzerá takto (obmedzenia $x_e \leq 1$ netreba písať explicitne, nakoľko vyplývajú z ostatných):

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

Napríklad pre graf vľavo dostaneme obmedzenia



$$\begin{aligned} x_{e_1} &\leq 1 \\ x_{e_2} + x_{e_4} &\leq 1 \\ x_{e_3} + x_{e_5} &\leq 1 \\ x_{e_1} + x_{e_2} + x_{e_3} &\leq 1 \\ x_{e_4} + x_{e_5} &\leq 1 \\ x_{e_1}, x_{e_2}, x_{e_3}, x_{e_4}, x_{e_5} &\geq 0 \end{aligned} \tag{20}$$

Keďže 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 konkrétne takým, ktoré sme dostali uvedeným postupom z nejakého bipartitného grafu. Môžme teda dúfať, 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 $\omega_{e_2} = \cdots = \omega_{e_5} = 10$ a vyriešiť relaxovaný program. Ak spustíme simplexový algoritmus[1], na naše potešenie bude nájdené optimálne riešenie celočíselné, aj keď existujú aj neceločíselné optimálne riešenia (napr. $x_{e_1} = 0$, $x_{e_2} = \cdots = x_{e_5} = \frac{1}{2}$). Môžme 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 náš program v maticovom tvare

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

kde

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

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 vynechať. Pridáme rezervné premenné a dostaneme program v normálnom tvare:

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

kde

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

[1]Pre väčší dramatický efekt to odporúčame čitateľovi naozaj vyskúšať.

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

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

takže podľa Cramerovho pravidla je i-ta zložka riešenia

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

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

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

Pri použití Cramerovho pravidla potrebujeme tieto determinanty:

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

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

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

Matica (nie nutne štvorcová) A s celočíselnými prvkami 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 2.5. *Majme lineárny program v normálnom tvare* $\max\{\mathbf{c}^\mathsf{T}\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\}$, *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

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

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

$$\det(A_B\langle i\rangle) = (-1)^{i+1}b_1C_1 + \cdots + (-1)^{i+m}b_mC_m,$$

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

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 2.6. *Nech A je TUM rozmerov $n \times m$ a nech $k \leq m$. Potom $B := \left(\begin{array}{c|c} A & \begin{array}{c} 0 \\ \hline I_k \end{array} \end{array}\right)$, 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. Keďže C je regulárna, platí $\det(C) = \pm 1$. □

Veta 2.7. *Majme lineárny program v tvare $\max\{\mathbf{c}^\mathsf{T}\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\}$, kde A je TUM matica a $\mathbf{b}$ je celočíselný vektor. Potom všetky bázové riešenia sú celočíselné.*

Dôkaz: Podľa Vety 2.6 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). □

Veta 2.8. *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 $\det(C) = 0$. □

Dôsledok 2.9. *Každý lineárny program tvaru*

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

alebo

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

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

1. *matica incidencie neorientovaného bipartitného grafu, alebo*
2. *matica incidencie orientovaného grafu*[2]

má všetky bázové riešenia celočíselné.

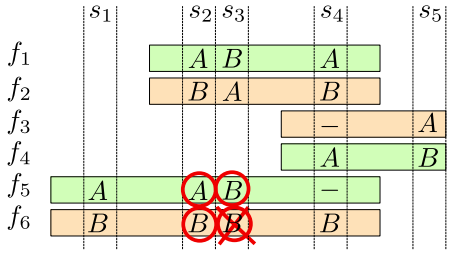
Dôkaz: V prípade 1) označme I_1 riadky prislúchajúce vrcholom jednej bipartície, v prípade 2) budú v I_1 všetky riadky. Použijeme vetu 2.8. □

Teraz vidíme, že problém MAX-WEIGHTED-BIPARTITE-MATCHING je iba jedným z mnohých problémov, kde simplexový algoritmus na relaxovanom lineárnom programe vždy nájde celočíselné optimum. Pochopiteľne, pri NP-ťažkých problémoch už také šťastie nebudeme 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.*

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. [7]). Jedince jedného druhu majú veľmi podobný genóm, ktorý sa líši iba na niekoľkých miestach. Cieľom je identifikovať tzv. *snipy* (*Single Nucleotide Polymorhism*) v genóme, čo sú miesta v genóme, kde sa môžu vyskytovať rôzne hodnoty, pričom naokolo je rovnaký obsah. Genóm je sada sekvencovaných 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.

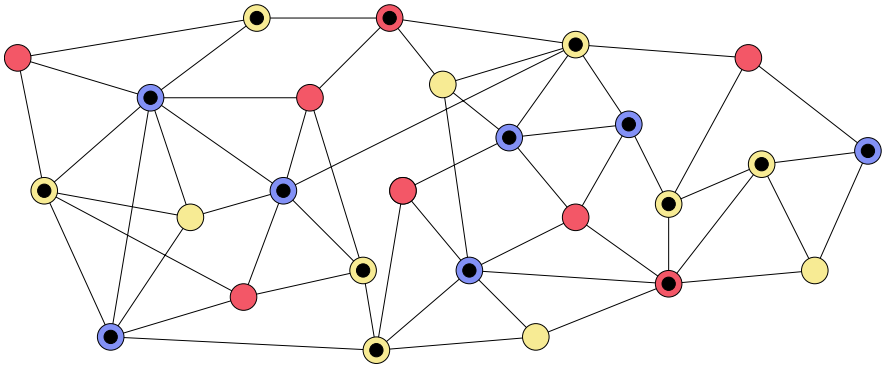


Konfigurácia fragmentov f_4,f_5 a snipov s_2,s_3 tvorí konflikt: podľa snipu s_2 musia fragmenty f_4 a f_5 mať rôzne farby, podľa snipu s_3 potom ale všetky hodnoty s_3 musia byť B, čiže s_3 nie je snip. Keby hodnota s_3 na fragmente f_6 bola A, zelené a červené fragmenty by zodpovedali dvom alternatívnym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

Vstupné dáta vždy obsahujú chyby. Konfigurácia dvoch fragmentov a dvoch snipov ako na predchádzajúcom obrázku tvorí konflikt: jeden zo snipov musel byť prečítaný zle a treba ho ignorovať. Ak máme pre každý snip číselne vyjadrenú jeho významnosť, môžme 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 2.10. *Daný je jednoduchý graf $G = (V, E)$ s vrcholmi ohodnotenými nezápornými váhami, t.j. funkciou $\omega : V \mapsto \mathbb{R}^+$. Cieľom problému* MIN-VERTEX-COVER *je vybrať množinu vrcholov tak, aby každá hrana bola incidentná aspoň s jedným vybratým vrcholom a celková váha vybratých vrcholov bola minimálna.*

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



Príklad vrcholového pokrytia. 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 zapísať 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:

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

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-ťažký, 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 konštanta, preto potrebujeme brať suprémum cez všetky vstupy danej veľkosti):

Definícia 2.11. *Majme optimalizačný problém $\mathcal{P}$ a polynomiálny algoritmus $\mathcal{A}$, ktorý ho rieši. Nech $m^\star(\mathbf{x})$ 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*

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

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

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

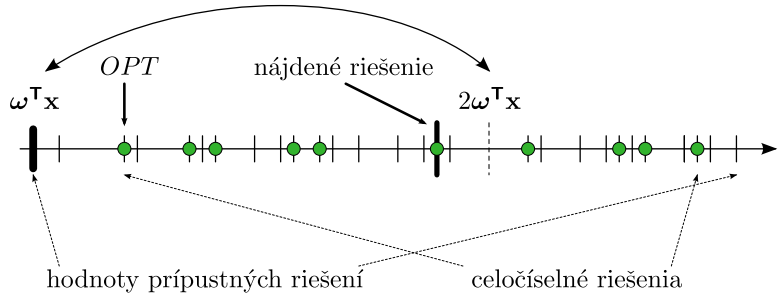
pričom suprémum sa berie cez všetky vstupy dĺžky n.

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

lineárny program

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

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



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

Najjednoduchší spôsob, ako z $\mathbf{x}$ dostať celočíselné riešenie je zaokrúhliť 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

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

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. $x_v \in \left\{0, \frac{1}{2}, 1\right\}$. 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 vovnútri telesa, a teda sa dá napísať ako $t\mathbf{y} + (1-t)\mathbf{z}$ pre dva body $\mathbf{y}$, $\mathbf{z}$ a $0 < 1 < t$.

Lema 2.12. *Majme ľubovoľný lineárny program v normálnom tvare. 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 $\tilde{\mathbf{c}}$ taký, že $\tilde{c}_j = 0$ pre $j \in B$ a $\tilde{c}_j = -1$ inak. Zjavne $\tilde{\mathbf{c}}^\mathsf{T}\mathbf{x} = \mathbf{0}$ a zároveň pre ľubovoľné $\mathbf{v} \geq \mathbf{0}$ platí $\tilde{\mathbf{c}}^\mathsf{T}\mathbf{v} \leq \mathbf{0}$. Navyše, ak $\mathbf{v}$ má aspoň jednu zložku $v_j > 0$ pre $j \notin B$, tak $\tilde{\mathbf{c}}^\mathsf{T}\mathbf{v} < \mathbf{0}$. Vieme, že $\mathbf{x} = t\mathbf{y} + (1-t)\mathbf{z}$ a teda $\mathbf{0} = \tilde{\mathbf{c}}^\mathsf{T}\mathbf{x} = t\tilde{\mathbf{c}}^\mathsf{T}\mathbf{y} + (1-t)\tilde{\mathbf{c}}^\mathsf{T}\mathbf{z}$ pre $\mathbf{y}, \mathbf{z} \geq \mathbf{0}$. To je možné iba vtedy, ak $\tilde{\mathbf{c}}^\mathsf{T}\mathbf{x} = \tilde{\mathbf{c}}^\mathsf{T}\mathbf{y} = \tilde{\mathbf{c}}^\mathsf{T}\mathbf{z} = \mathbf{0}$, 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 $A\mathbf{x} = A\mathbf{y} = A\mathbf{z} = \mathbf{b}$. Keďže $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ 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 $\mathbf{x} = \mathbf{y} = \mathbf{z}$. □

Vieme teda, že nám stačí ukázať, že každé prípustné riešenie $\mathbf{x}$ programu (23), ktoré priradí nejakému vrcholu hodnotu $x_v \notin \left\{0, \frac{1}{2}, 1\right\}$ sa dá napísať ako konvexná kombinácia dvoch prípustných riešení $\mathbf{y}$ a $\mathbf{z}$.
Zoberme si ľubovoľné prípustné riešenie $\mathbf{x}$, ktoré priradí nejakému vrcholu hodnotu $x_v \notin \left\{0, \frac{1}{2}, 1\right\}$. Vrcholy, v ktorých $\mathbf{x}$ má iné hodnoty ako $\left\{0, \frac{1}{2}, 1\right\}$ rozdeľme do dvoch skupín takto

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

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

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

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 $\mathbf{y}$,$\mathbf{z}$ spĺňajú obmedzenia na hranách. 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: buď $x_u, x_v \in \left\{0, \frac{1}{2}, 1\right\}$ alebo $u \in V_+$, $v \in V_-$ (alebo naopak). V každom prípade, pre ľubovoľnú voľbu ε, aj táto podmienka ostane splnená v $\mathbf{y}$ aj $\mathbf{z}$.
Na tomto mieste sme čitateľovi dlžní vysvetlenie. Našli sme 2-aproximačný algoritmus a tvárime sa spokojne. Prečo? V realite je predsa riešenie, ktoré by bolo dvojnásobkom hľadaného optima, zväčša nanič. Odpovedí, prečo sú algoritmy s dokázateľným aproximačným pomerom zaujímavé, je niekoľko.

Teoretický[3] **záujem.** Napriek silným výsledkom teórie zložitosti, ani po desaťročiach snahy poriadne nerozumieme, prečo niektoré problémy sú "ľahké" a iné "ťažké", či už "ťažkosť" znamená optimálne riešenie alebo aproximáciu. Sú problémy, pre ktoré sa dá nájsť v polynomiálnom čase ľubovoľne presná aproximácia. 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 takéhoto 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.

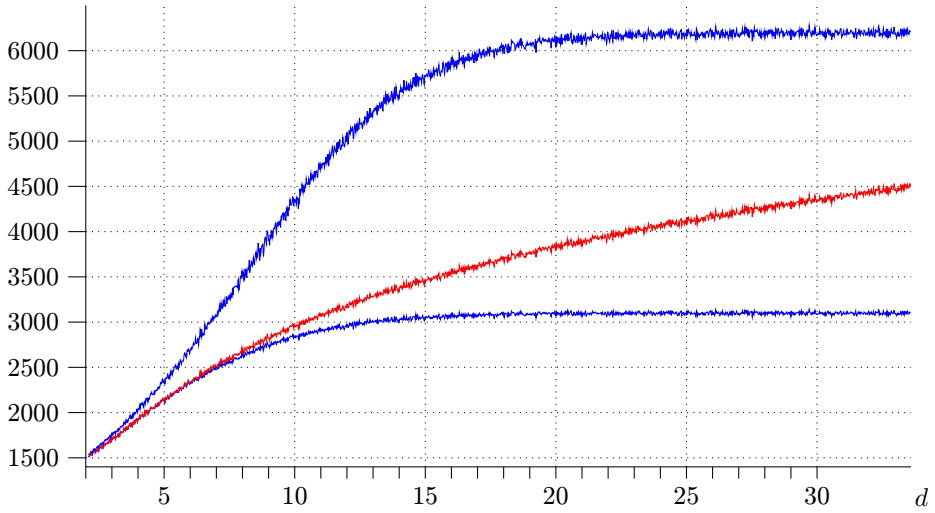
Použitie ako zarážka. Väčšinou sa na riešenie (NP-)ťažkých problémov používajú rôzne heuristiky, buď navrhnuté špecificky pre daný problém alebo metaheuristiky typu simulované žíhanie, tabu-search, či neurónové siete. Heuristiky väčšinou nájdu riešenie pomerne blízko optimu, ale v zriedkavých prípadoch 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

[3] autor nevníma slovo "*teoretický*" derogatívne

"bežné" inštancie je ale ošemetná záležitosť. Analýza priemerného prípadu sa spolieha na apriórnu informáciu o pravdepodobnostnom rozdelení vstupov; v realite sú ale dokonale náhodné vstupy veľkou výnimkou. V prípade, že algoritmus je plánovaný pre konkrétne 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 optima z opačnej strany (v našom prípade je hodnota relaxovaného optima dolným odhadom pre skutočné optimum); máme teda pre každú inštanciu interval, v ktorom vieme, že sa skutočné optimum nachádza.

Ako ilustráciu sa pozrime na náš 2-aproximačný algoritmus pre Min-Vertex-Cover. Ako prvý test použijeme napríklad zoznam všetkých súvislých kubických[4] grafov na 20 vrcholoch (je ich 510489). V nevážhovanom prípade (t.j. všetky vrcholy majú váhu 1) sú všetky relaxované riešenia tvaru $x_{v_1} = \cdots = x_{v_{20}} = \frac{1}{2}$; optimum relaxovaného programu je teda 10 a po zaokrúhlení máme triviálne riešenie s hodnotou 20. Priemerné celočíselné optimum je ≈ 11.5. Ak sme každému vrcholu nastavili váhu náhodné číslo z rozsahu $\{1, \ldots, 100\}$, 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 $\{1, \ldots, 10\}$, 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 pozrime 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, ktoré boli súvislé. Nasledujúci obrázok ukazuje závislosť priemerného relaxovaného optima, zaokrúhleného riešenia a skutočného optima pre rôzne hodnoty d s váhami tvaru $1 + 2^i$, kde i je náhodné číslo z rozsahu $\{1, \ldots, 10\}$.



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 zaokrúhľovaním 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 relaxované riešenie zložené takmer zo samých $\frac{1}{2}$, takže zaokrúhlením sa získa dvojnásobok; zároveň ale stúpa celočíselné optimum, takže

[4] t.j. každý vrchol má práve troch susedov

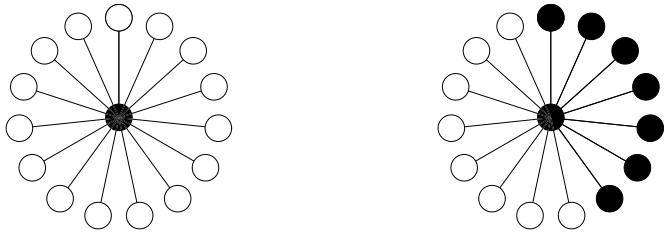
aproximačný faktor postupne klesá. Vidíme teda, že aj pre jednoduchý algoritmus, ktorý má garantovaný aproximačný pomer 2, existujú triedy vstupov, kde dosahuje oveľa lepší pomer.

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 môžme formulovať aj tak, že chceme zachovať čo najviac snipov tak, aby žiadne dva vybraté neboli spojené hranou. Dostávame tak problém Max-Independent-Set:

Definícia 2.13. *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 aproximácie sa ale správajú veľmi odlišne.



V neváhovanom hviezdicovom grafe s n vrcholmi je maximálna nezávislá množina $S^\star$ 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 pokrytia.

Rovnako ako problém Min-Vertex-Cover, aj Max-Independent-Set sa dá zapísať ako celočíselný program

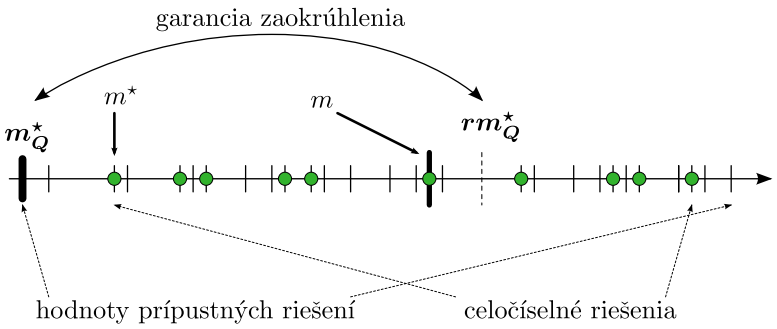
$$\begin{array}{rlcll} \text{maximalizovať} & \sum\limits_{v \in V} \omega_v x_v & & & \\ \text{pri obmedzeniach} & x_u + x_v & \leq & 1 & \forall e = (u,v) \in E \\ & x_v & \geq & 0 & \forall v \in V \\ & x_v & \in & \mathbb{Z} & \end{array} \tag{24}$$

Keď budeme uvažovať špeciálny prípad neováhovaných grafov, t.j. $\omega_{v_1} = \cdots = \omega_{v_n} = 1$ podobne ako v prípade Min-Vertex-Cover budú pre husté grafy optimálne riešenia relaxovaného programu tvaru $x_{v_1} = \cdots = x_{v_n} = \frac{1}{2}$. Tu sa však podobnosť s Min-Vertex-Cover končí: je totiž veľa 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 [6] 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 zapísať 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.

2.2 Deterministické zaokrúhľovanie

V predchádzajúcej kapitole sme pri probléme Min-Vertex-Cover použili nasledovnú všeobecnú schému na navrhovanie r-aproximačných algoritmov:

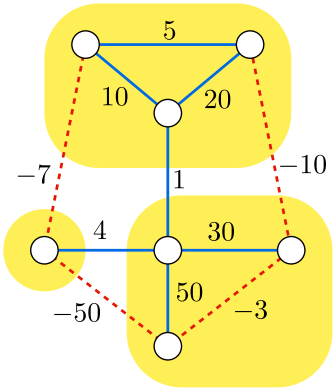


Ak hľadáme minimum celočíselného programu s hodnotou $m^\star$, nájdeme (potenciálne menšie) minimum relaxovaného programu s hodnotou $m_Q^\star$, zaokrúhlime ho a dostaneme riešenie s hodnotou m. Ak ukážeme, že pri zaokrúhľovaní hodnota riešenia stúpla najviac r-krát, máme zaručený r-aproximačný algoritmus.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

Min-Multi-Cut

Ako motivačný príklad zoberme nasledovný problém[5]: máme databázu textov, v ktorých sa vyskytujú odkazy na rôzne osoby a našim cieľom je rozlíšiť, 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 referencuje 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. Keďže 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 vovnútri komponentu, ich váha (v absolútnej hodnote) sa priráta k celkovej chybe.

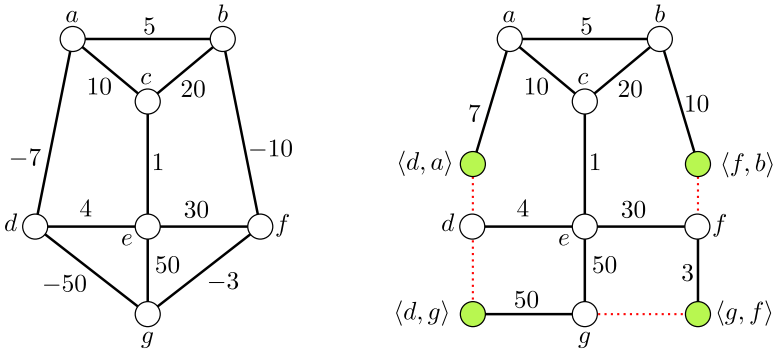
[5] porov. [1, 4]



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-ťažká, preto sa skromne uspokojíme s približným riešením. Pretransformujme si náš 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.

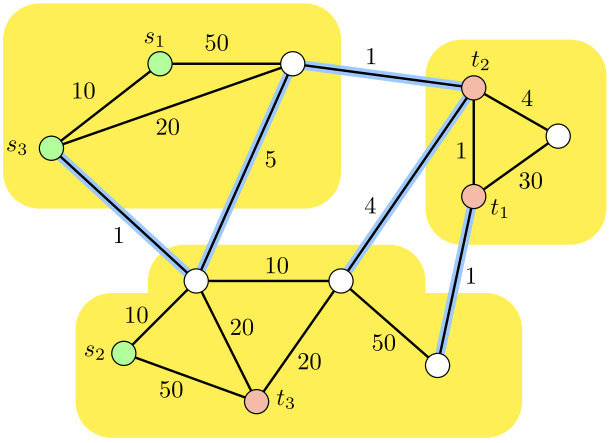
Cvičenie. *Dokážte, že táto transformácia je ekvivalentná, t.j. ak máme riešenie pôvodného problému s celkovou chybou c, existuje riešenie transformovaného problému s váhou c a naopak, ak máme riešenie transformovaného problému s váhou c, tak existuje riešenie pôvodného problému s celkovou chybou c.*



Pôvodný a transformovaný graf (keďže 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 2.14. *Majme daný jednoduchý graf $G = (V, E)$ s hranami ohodnotenými nezápornými váhami, t.j. funkciou $\omega : E \mapsto \mathbb{R}^+$ a v ňom k dvojíc vrcholov (s_i, t_i), $i = 1, \ldots, k$. Cieľom problému* Min-Multi-Cut *je odobrať 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-ťažký. Pokúsme 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 odstránili 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 písať explicitne, lebo vyplýva z minimalizácie: ak je nejaké $x_e > 1$, môžme ho zmenšiť na 1, obmedzenia ostanú zachované a hodnota riešenia klesne):

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

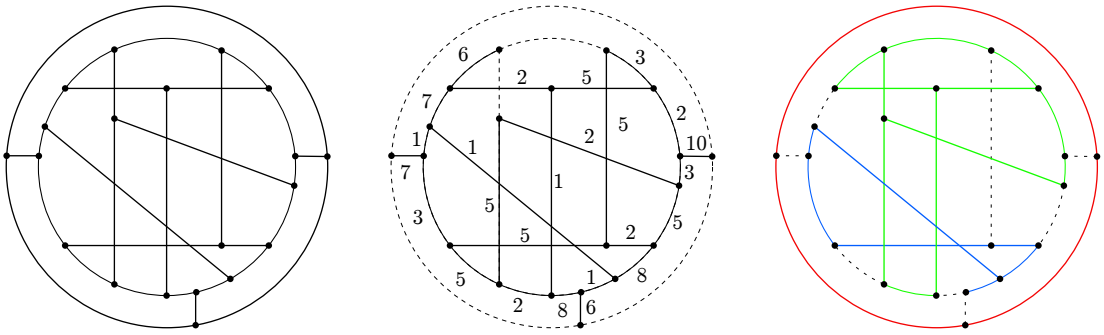
Program (25) teraz relaxujeme tak, že odstránime obmedzenia $x_e \in \mathbb{Z}$. Dostaneme lineárny program

$$\begin{aligned} &\text{minimalizovať} && \sum_{e \in E} x_e \omega_e && && \\ &\text{pri obmedzeniach} && \sum_{e : e \in \pi} x_e && \geq\ 1 && \forall i \in \{1, \ldots, k\},\ \ \forall \pi \in \mathcal{P}_{s_i,t_i} \\ & && x_e && \geq\ 0 && \forall e \in E \end{aligned} \tag{26}$$

ktorý môžme interpretovať nasledovne: na hrany grafu chceme namontovať žiariče so silou x_e tak, aby každá častica idúca z s_i do t_i dostala počas svojej cesty aspoň jednu jednotku žiarenia. Každá hrana má cenu, ktorú zaplatíme za montáž žiariča jednotkovej sily (ω_e), pričom cena žiariča na danej hrane je lineárna od jeho intenzity. Náš 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, priraďme každému vrcholu v potenciál $p_v^{(i)} \in \mathbb{R}$ tak, že $p_{t_i}^{(i)} - p_{s_i}^{(i)} \geq 1$. Každej ceste $\pi : s_i = v_1, v_2, \ldots, v_z = t_i$ prislúcha postupnosť potenciálov vrcholov $p_{v_1}^{(i)}, p_{v_2}^{(i)}, \ldots, p_{v_z}^{(i)}$. Ak vynútime, aby sila žiariča na každej hrane 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 na celej ceste sa nazbiera aspoň jednotka. Namiesto programu (26) nám teda stačí riešiť polynomiálne veľký program

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

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. Ľahko vidno, že takto nastavené potenciály spĺňajú 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 $x_e^\star$ a cenou $m_Q^\star = \sum_{e\in E} x_e^\star\omega_e$. Keby všetky $x_e^\star \in \{0,1\}$, 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 $k \in \{1,\ldots,20\}$ dvojíc 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 zaokrúhlením všetkého nenulového nahor preto v 89.7% prípadov dostaneme prinajhoršom 4-aproximáciu. Prv, než by nás nedôslednosť zviedla k nejakým unáhleným 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 vzdialenosti aspoň 4.



Vľavo je graf s 20 vrcholmi a 30 hranami (všetky hrany majú váhu 1). Uvažujeme inštanciu Min-Multi-Cut, v ktorej treba oddeliť každú dvojicu vrcholov vo vzdialenosti aspoň 4 (priemer grafu je 5, dvojíc vo vzdialenosti aspoň 4 je 22). V strede je optimálne riešenie relaxovaného LP (čísla na hranách sú násobky $\frac{1}{15}$) s hodnotou 7. Keďže 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 "zaokrúhľovacím" algoritmom. Označme $m^\star$ optimálnu cenu riešenia problému Min-Multi-Cut: platí platí $m_Q^\star \leq m^\star$. Ukážeme si teraz, ako "zaokrúhliť" hodnoty $x_e^\star$ (t.j. vybrať hrany do rezu) tak, aby sme dostali riešenie problému Min-Multi-Cut s cenou najviac $4\ln(2k)m_Q^\star$.
Náš "zaokrúhľovací" algoritmus bude postupne z grafu "vyhrýzať" množiny $V_1, V_2, \ldots, V_k$. 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 oddeľujúce 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 (aspoň 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 oddeľujú 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 $x_e^\star$ a predstavme si optimálne riešenie (26) ako sieť potrubí, kde hrana e má prierez ω_e a dĺžku $x_e^\star$, t.j. má objem $\omega_e x_e^\star$. Pre hranu $e = (u, v)$ označme $d(u, v) = x_e^\star$ a prirodzeným spôsobom môžme merať vzdialenosť ľubovoľných dvoch vrcholov $d(v_1, v_2)$. Keď algoritmus vyberie nejakú množinu V_i, musí prerezať všetky potrubia, ktoré ju oddeľujú 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 $V_1, \ldots, V_{r-1}$ z grafu G. Prirodzeným spôsobom si zadefinujeme pojem 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

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

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

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

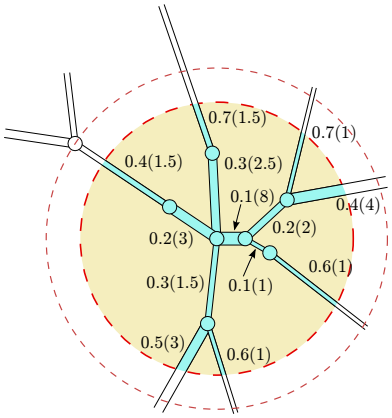
Vnútorné hrany gule prispievajú do jej objemu celým svojím objemom, hrany z hranovej hranice iba príslušnou časťou. Najviac nás, pochopiteľne, zaujíma cena gule, t.j. veľkosť jej hranového rezu:

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

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

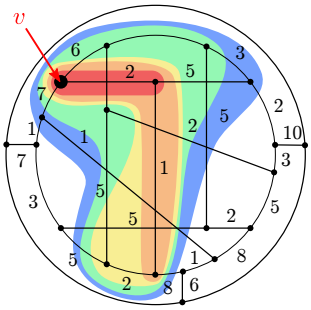
$$F_\rho(v) = \frac{C_\rho(v)}{V_\rho(v)}$$

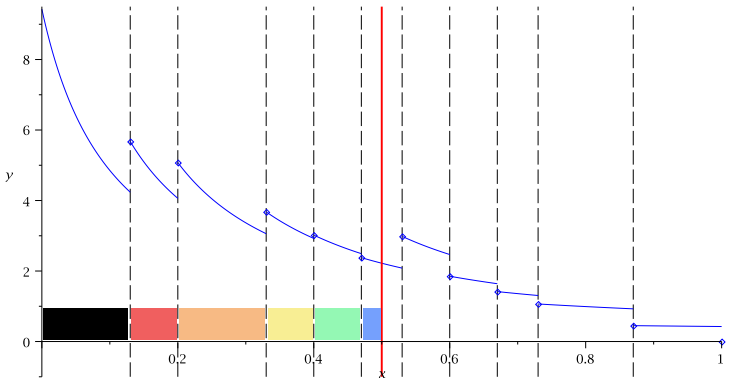
Funkcia $F : \rho \mapsto F_\rho(v)$ pre daný vrchol v má body nespojitosti pre tie hodnoty ρ, pre ktoré existuje vrchol w vo vzdialenosti ρ od v. Na každom intervale medzi bodmi nespojitosti je F diferencovateľná a klesajúca. Preto môžme ľahko nájsť minimum $F_\rho(s_i)$ na intervale $(0, 1/2)$: vyberieme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



Predstava relaxovaného riešenia ako siete potrubí. Pri hranách je napísaná ich dĺžka ($x_e^\star$) a v zátvorke 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.

$$\forall v \exists \rho < 1/2 : \; F_\rho(v) \le 2\ln(2k) \tag{28}$$





Funkcia F pre jeden vrchol v z predchádzajúceho príkladu. Optimum je $\Psi = 7$ a $k = 22$. Do vzdialenosti $\frac{2}{15} \approx 0.13$ je vnútro gule prázdne a veľkosť rezu je 3, preto pre $0 \le \rho < \frac{2}{15}$ je $F_\rho(v) = \frac{3}{\frac{7}{22}+3\rho}$. Pre $\frac{2}{15} \le \rho < \frac{3}{15} = 0.2$ 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 $\mathcal{B}_{\rho_1}(s_{i_1}), \ldots, \mathcal{B}_{\rho_h}(s_{i_h})$. Cena rezu je

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

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

Dôkaz dokončíme dôkazom tvrdenia (28): Fixujme si vrchol v a uvažujme všetky funkcie ako funckie vzdialenosti 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

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

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.

Predcchádzjúce úvahy môžme zhrnúť do tvrdenia:

Veta 2.15. *Existuje algoritmus, ktorý pre problém* Min-Multi-Cut *vráti v polynomiálnom čase riešenie s veľkosťou nanajvýš* $4\ln(2k)$*-násobku optima.*

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 $d \geq 3$. Zvoľme 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 vzdialenosti aspoň α. Keď pre každú hranu e zvolíme $x_e = \frac{1}{\alpha}$, dostaneme prípustné riešenie programu (26), a preto $m^\star_Q \leq \frac{n}{\alpha}$. Ukážeme teraz, ako docieliť, aby ľubovoľné celočíselné riešenie malo veľkosť aspoň n. Nech M je optimálne celočíselné riešenie, t.j. množina hrán, ktorá rozpojí všetky vrcholy vo vzdialenosti aspoň α. Odstránením M z G dostaneme graf $G' = (V, E - M)$ s niekoľkými komponentami súvislosti, pričom platí:

Lema 2.16. *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 hocijaký 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 vzdialenosti najviac 2 od v. Pokračujme indukciou a dostaneme, že môže byť najviac $d + d^2 + \cdots + d^{\alpha-1}$ vrcholov vo vzdialenosti nanajvýš α od v. Lenže $1 + d + d^2 + \cdots + d^{\alpha-1} = \frac{d^\alpha - 1}{d-1} < d^\alpha$, 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. □

Nech teraz $S_1, \ldots, S_h$ 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 2.17. *Graf $G = (V, E)$ nazveme* expandérom*, ak pre každú množinu vrcholov $S \subset V$ platí*

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

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

Predpokladajme teraz, že náš graf G je expandér a zvoľme 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

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

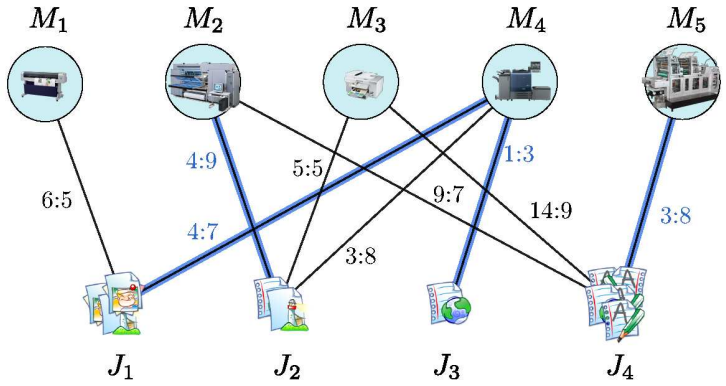
Keď si to zhrnieme, máme inštanciu na grafe stupňa d s n vrcholmi, kde optimálny multirez má $\Omega(n)$ hrán, ale existuje relaxované riešenie s cenou $O\left(\log d \frac{n}{logn}\right)$. Zároveň máme $k = \Theta(n^2)$ dvojíc s_i, t_i, lebo pre každý vrchol je aspoň $n/2$ vrcholov vo vzdialenosti aspoň α. Ak chceme docieliť, aby parametre n a k boli nezávislé, stačí pre nejaké ℓ nahradiť každú hranu cestou dĺžky ℓ a ponechať dvojice s_i, t_i: počet vrcholov narastie, ale aproximačný faktor ostane rovnaký. Vidíme teda, že náš zaokrúhľovací algoritmus nemôže dať v najhoršom prípade lepší 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 ťažké 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 (konkrétne, že $x_i \in \{0, \frac{1}{2}, 1\}$), na základe ktorej sme mohli odhadnúť zaokrúhľovaciu 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 cieľom je naplánovať úlohy na tlačiarne tak, aby sa stihol termín a zároveň náklady boli čo najmenšie. To nás vedie k nasledujúcej formulácii:

Definícia 2.18. *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. Cieľom 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 zmestiť 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 algoritmicky 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 $\sum_{j \in J} p_j = 2T$, rozhodnúť, či sa dajú úlohy podeliť tak, aby na každom procesore bol čas najviac T je ekvivalentné známemu NP-úplnému problému MIN-PARTITION[6]. Neočakávame preto, že by sme ho vedeli efektívne riešiť. Ukážeme riešenie trochu jednoduchšej úlohy. Povedzme, že naša typografická firma pri prevzatí zakázky sľúbila 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 myslený 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 2.18 nám priamočiaro dáva ILP:

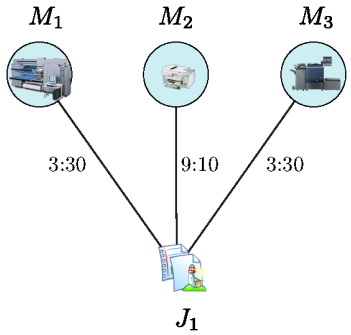
$$\begin{aligned} &\text{minimalizovať} && \sum_{(i,j) \in E} x_{ij} c_{ij} && \\ &\text{pri obmedzeniach} && \sum_{i \in M} x_{ij} = 1 && \forall j \in J \\ & && \sum_{j \in J} t_{ij} x_{ij} \leq T && \forall i \in M \\ & && x_{ij} \in \{0, 1\} && \forall i \in M,\ \forall j \in J \end{aligned} \tag{29}$$

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

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

[6]Problém MIN-PARTITION je definovaný nasledovne: pre množinu čísel $A = \{a_1, \ldots, a_n\}$ 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 stratégiu: zafixovať celočíselné hodnoty a zvyšok riešiť nejakým iným lineárnym 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 využiť 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



(30)

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.

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

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 kľúčová nasledovná charakterizácia optimálnych riešení:

Lema 2.19. *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 2.19 hovorí, že ak sa po vyriešení programu (31) ocitneme v situácii, kde nie je žiadna celočíselná premenná, tak buď máme procesor, ktorý dokáže spracovávať najviac jednu úlohu ($d(i) \leq 1$), alebo procesor, ktorý dokáže spracovávať práve dve úlohy a z každej z nich spracováva kus (navyše, v súčte sú tie kusy ≥ 1). Pri dôkaze lemy 2.19 využijeme nasledujúce pomocné tvrdenie:

Lema 2.20. *Nech* $\mathbf{x}$ *je bázové riešenie programu (31), v ktorom všetky* $x_{ij} > 0$. *Potom existuje množina* $M'' \subseteq M'$ *veľkosti* $|M''| = |E| - |J|$ *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 1.2). Najprv požadujeme program v normálnom tvare $\max_{\mathbf{x}}\{\mathbf{c}^\mathsf{T}\mathbf{x} \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq 0\}$, 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

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

kde

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

Podľa Definície 1.2, 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 $\sum_{j \in J} t_{ij} x_{ij} + s_i = T_i$. □

Dôkaz Lemy 2.19: 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 2.20. Platí

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

Nerovnosť (♣) 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 (♣). 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 párnu 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 $\sum_{j \in J} x_{ij} \geq 1$. □

S charakterizáciou bázových riešení pomocou Lemy 2.19 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 žiaden 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) stať, 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 žiaden 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'. Poďme teraz túto tému rozviesť 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 $x_{ij} = 0$,
vyhoď 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. Ľahko 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 2.21. *Uvedený algoritmus vyrieši 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 nezmení 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 spojená[7] 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 $\sum_{j \in J} x_{ij} \geq 1$.

□

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

2.3 Randomizované zaokrúhľovanie

Budeme ďalej pokračovať v rovnakej schéme ako doteraz: máme riešiť optimalizačný problém. Zapíšeme si ho ako celočíselný program. Vyriešime jeho relaxovanú verziu. Rozmýšľame, 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 intuíciu 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 2.2). Uvažujme teraz jeho optimalizačnú verziu:

Definícia 2.22. *Majme n logických premenných $x_1, \ldots, x_n \in \{true, false\}$.* Literál *je premenná x_i alebo jej negácia $\overline{x_i}$.* Klauzula *je dizjunkcia literálov $C = l_1 \vee l_2 \vee \cdots \vee l_{k_C}$. Formula* v konjunktívnej normálne forme *(CNF) je konjunkcia klauzúl $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. 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 $x_1, \ldots, x_n$ tak, aby súčet váh splnených klauzúl bol maximálny.*

Ak je F splniteľná 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.

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

Ľahko 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 garantovane 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 $\mathrm{E}[W]$. 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

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

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. Keďže každá premenná x_i je nastavená nezávisle[8] 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, že každá klauzula má aspoň k literálov. Potom algoritmus `A1` v očakávanom prípade získa váhu aspoň

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

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 $i = 1, 2, \ldots$) 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

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

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 $\overline{x_i}$, je táto pravdepodobnosť $1 - 2^{1-s(C_i)}$ a inak ostala rovnaká $1 - 2^{-s(C_i)}$. Rovnako sa dá zrátať hodnota $\mathrm{E}[W \mid x_1 = 0]$. Pretože x_1 bolo nastavené na 1 s pravdepodobnosťou 1/2, je

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

a teda jedna z $\mathrm{E}[W \mid x_1 = 0]$, $\mathrm{E}[W \mid x_1 = 1]$ je aspoň tak veľká ako $\mathrm{E}[W]$. Dostali sme algoritmus, ktorý používa $n - 1$ náhodných bitov a má v očakávanom prípade cenu aspoň $\mathrm{E}[W]$.
Derandomizovaný algoritmus `A1det` bude pracovať v n iteráciách, pričom v t-tej iterácii si bude pamätať konštanty $c_1, \ldots, c_{t-1}$. Pre všetky C_i zráta $\Pr[X_i = 1 \mid x_1 = c_1, \ldots, x_{t-1} = c_{t-1}, x_t = 0]$ a $\Pr[X_i = 1 \mid x_1 = c_1, \ldots, x_{t-1} = c_{t-1}, x_t = 1]$. Na základe toho toho zráta očakávané hodnoty $\mathrm{E}[W \mid x_1 = c_1, \ldots, x_{t-1} = c_{t-1}, x_t = 0]$ a $\mathrm{E}[W \mid x_1 = c_1, \ldots, x_{t-1} = c_{t-1}, x_t = 1]$ a podľa toho, ktorá je väčšia, nastaví c_t. Po n iteráciách budú hodnoty $c_1, \ldots, c_n$ 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 2.23. *Majme randomizovaný algoritmus $\mathtt{A_R}$ riešiaci v polynomiálnom čase optimalizačný problém $\mathcal{P}$, pričom na vstupe $\mathbf{x}$ urobí $R(\mathbf{x})$ náhodných rozhodnutí $r_1, r_2, \ldots, r_{R(\mathbf{x})}$; rozhodnutia sú nezávislé binárne náhodné premenné s $\Pr[r_i = 1] = p_i$. Predpokladajme, že existuje polynomiálny algoritmus, ktorý pre ľubovoľné i a konštanty $c_1, \ldots, c_i \in \{0, 1\}$ vyráta $\mathrm{E}[\mathtt{A_R}(\mathbf{x}) \mid b_1 = c_1, \ldots, b_i = c_i]$. Potom existuje deterministický algoritmus, ktorý v polynomiálnom čase nájde riešenie aspoň tak dobré, ako $\mathrm{E}[\mathtt{A_R}(\mathbf{x})]$.*

Vo všeobecnom prípade sa v jednej iterácii derandomizovaného algoritmu využije

$$\begin{aligned}\mathrm{E}[W \mid r_1 = c_1, \ldots, r_{i-1} = c_{i-1}] = {} & p_i \mathrm{E}[W \mid r_1 = c_1, \ldots, r_{i-1} = c_{i-1}, r_i = 1] + \\ & (1 - p_i)\mathrm{E}[W \mid r_1 = c_1, \ldots, r_{i-1} = c_{i-1}, r_i = 0],\end{aligned}$$

odkiaľ 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.

[8] zdôrazňujeme, že náhodné premenné X_i nie sú nezávislé

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 čitateľa znalého 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 ohodnotení. Keď 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 zapísať problém Max-Sat ako celočíselný lineárny program. Celkom prirodzene majme pre každú premennú x_i zo vstupnej formuly premennú $p_i \in \{0,1\}$. Cieľom 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. Cieľom 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 žiaden literál z C_i nie je splnený, z_i musí byť 0. Označme C^+ indexy premenných, ktoré sa v klauzule C vyskytujú v pozitívnej forma a C^- indexy tých premenných, ktoré sú v C negované. Dostávame nasledovný program:

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

a jeho relaxovanú verziu

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

Konečne sa dostávame k jadru tejto kapitoly – randomizovanému zaokrúhľovaniu. Algoritmus `A2` vyrieši program (34), čím dostane optimálne hodnoty $p_i^\star$, $z_i^\star$ a každú premennú x_i nastaví na 1 nezávisle s pravdepodobnosťou $p_i^\star$. 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

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

Klauzula C_i je nesplnená, ak nie je splnený žiaden z jej literálov. Pozitívny literál x_j je nesplnený s pravdepodobnosťou $1 - p_j^\star$ a negatívny $\overline{x_j}$ s pravdepodobnosťou $p_j^\star$, takže

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

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

$$=\; 1 - \left(1 - \frac{\sum_{j \in C_i^+} p_j^\star + \sum_{j \in C_i^-} (1 - p_j^\star)}{s(C_i)} \right)^{s(C_i)}$$

$$\geq\; 1 - \left(1 - \frac{z_i^\star}{s(C_i)} \right)^{s(C_i)} \tag{37}$$

kde nerovnosť (36) je aplikáciou aritmeticko-geometrickej nerovnosti

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

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. $\Pr[X_i = 1] \geq \beta z_i^\star$ pre nejaké β, ktoré môže závisieť od C_i, ale nezávisí od $z_i^\star$. 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

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

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 vrátime k nerovnosti (37), dostávame

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

Označme

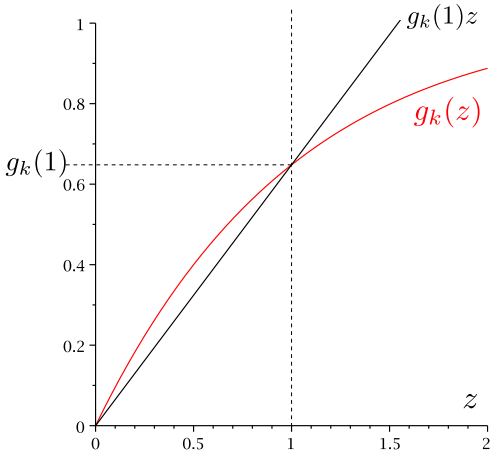
$$\beta(k) := g_k(1) = 1 - \left(1 - \frac{1}{k}\right)^k$$

funkciu premennej k. Pre očakávanú cenu algoritmu `A2` máme

$$\mathrm{E}[W] \geq \sum_{i=1}^{m} \omega_i \beta\left(s(C_i)\right) z_i^\star. \qquad (38)$$

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

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



Môžeme použiť vetu 2.23 a vyrobiť deterministický algoritmus `A2det`; príslušné podmienené pravdepodobnosti sa budú rátať analogicky, podľa riadku (35).

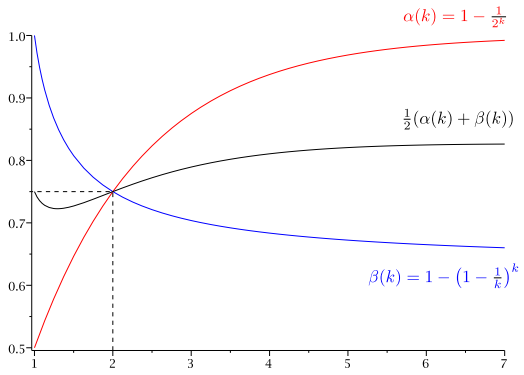
Výsledný algoritmus s garanciou 3/4

Máme teraz dva algoritmy: `A1det`, ktorý funguje dobre na inštanciách s dlhými klauzulami a `A2det`, ktorý funguje dobre na inštanciách s krátkymi klauzulami. Čo môžeme robiť na inštanciách, ktoré majú dlhé aj krátke klauzuly? Urobíme prirodzenú vec: spustíme obidva algoritmy a z výsledkov vyberieme ten lepší. Z (32) a (38) a z vety 2.23 dostávame pre ceny algoritmov `A1det` a `A2det` na formule F:

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

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

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



kde posledná nerovnosť vyplýva z toho, že $z_i^\star \leq 1$. 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

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