2.1 Vaříme zdravě a levně

Potravní inspekce Evropské unie odhalila, že jídla podávaná v restauračním zařízení „U neurvalce“, jako například utopenci, slanečci a guláš se šesti, neodpovídají novým předpisům, a jmenovitě zmínila ve zprávě nedostatek vitamínů A a C a vlákniny. Provozovatel restaurace se snaží nedostatky řešit dodáním zeleninové přílohy, kterou hodlá zkombinovat z bílého zelí, mrkve a zásob nakládaných okurek nalezených ve sklepě. Následující tabulka shrnuje číselné podklady: předepsaná množství jednotlivých vitamínů a minerálů na porci, jejich zastoupení v příslušných potravinách a jednotkové ceny potravin[1].

surovina	mrkev syrová	zelí bílé syrové	okurky nakládané	požadováno na 1 porci
vitamín A [mg/kg]	35	0.5	0.5	0.5 mg
vitamín C [mg/kg]	60	300	10	15 mg
vláknina [g/kg]	30	20	10	4 g
cena [Kč/kg]	15	10	3*	—

*Zůstatková účetní cena zásob, patrně neprodejných.

Za jakou minimální dodatečnou cenu na každou porci lze požadavkům inspekce tímto způsobem vyhovět? To lze vyjádřit následující úlohou lineárního programování:

$$
\begin{array}{ll}
\text{minimalizovat} & 15x_M + 10x_Z + 3x_O \\
\text{za podmínek} & x_M \geq 0 \\
& x_Z \geq 0 \\
& x_O \geq 0 \\
& 35x_M + 0.5x_Z + 0.5x_O \geq 0.5 \\
& 60x_M + 300x_Z + 10x_O \geq 15 \\
& 30x_M + 20x_Z + 10x_O \geq 4.
\end{array}
$$

Proměnná x_M udává množství mrkve přidané k jedné porci, a podobně pro x_Z a x_O. Účelová funkce vyjadřuje cenu této kombinace. Množství mrkve, zelí i okurek jsou vždy nezáporná, což říkají podmínky $x_M \geq 0$, $x_Z \geq 0$, $x_O \geq 0$ (kdybychom je nezahrnuli, optimální řešení by možná mohlo mít množství některé dražší suroviny záporné, čímž by se jakoby ušetřilo). Konečně nerovnice na posledních třech řádcích specifikují dodržení celkového obsahu vitamínů A a C a vlákniny.

Úlohu můžeme vyřešit například simplexovou metodou. Optimální řešení dává cenu 1,40Kč s dávkou 9,5g mrkve, 38g zelí a 290g okurek na porci

[1] Pro ty, kdo se zajímají o zdravou výživu: obsahy vitamínů a další údaje jsou víceméně realistické.

(vše zaokrouhleno na dvě platné číslice). Další inspekcí by nejspíš neprošlo – ve skutečnosti by asi bylo třeba přidat další omezení, například že okurek nesmí být příliš mnoho.

Tento příklad jsem zařadil, abych příliš nevybočoval z řady. Jak se zdá, všechny úvody do lineárního programování začínají různými vyživovacími příklady, nejspíš proto, že první větší optimalizační problém, na kterém se simplexový algoritmus v roce 1947 testoval, bylo složení potravních dávek. Jaké potraviny zkombinovat a v jakém množství, aby se dodržela minimální množství všech důležitých živin a aby denní příděl přitom vyšel co nejlevněji? Tehdy zformulovaná úloha lineárního programování měla 77 proměnných a 9 omezení a její řešení simplexovým algoritmem na ručních kalkulátorech zabralo asi 1000 pracovních hodin.

Později, když už George Dantzig měl přístup k počítači, zkoušel optimalizovat dietu i pro sebe. Optimální řešení první úlohy lineárního programování, již k tomu cíli sestavil, doporučovalo konzumaci několika litrů octa denně. Když z dalšího zadání ocet odstranil, vyšlo jako nejvhodnější základ stravy zhruba 200 polévkových kostek na den.

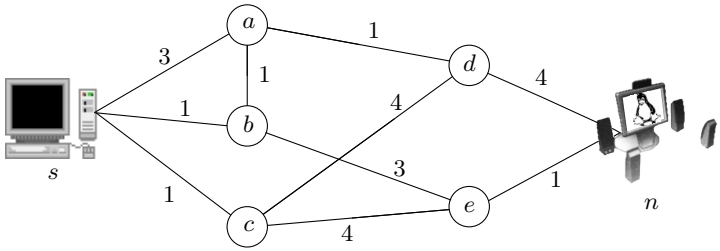
Tato historka, jejíž pravdivost není zcela vyloučena, nijak nesnižuje sílu lineárního programování, ale ilustruje, jak těžké je matematicky zachytit všechny důležité aspekty problému ze života. V oblasti výživy například dodnes není jasné, jaký přesně mají jednotlivé potraviny vliv na lidský organismus (i když samozřejmě spousta věcí jasných je, a naděje, že věda budoucnosti bude doporučovat například tlačenku jako hlavní ingredienci zdravé stravy, budou patrně zklamány). I kdyby to bylo jasné dokonale, málokdo chce a umí přesně zformulovat požadavky, co by od své výživy požadoval – mnohem snáz se to zřejmě dělá pro výživu někoho jiného. Navíc mezi potřebou různých živin jsou různé nelineární závislosti, takže úloha lineárního programování nemůže problém výživy dokonale vystihnout.

Lineární programování se běžně využívá v průmyslu, v zemědělství i ve službách, ale mnohé z takových aplikací jsou z abstraktního hlediska pouhými variantami problému výživy a neobsahují žádné zásadně nové matematické triky. Sestavit v praxi pro takové problémy dobré modely může stále být obtížné, ale obtížnost není rázu matematického. Proto se zde takovými úlohami nebudeme dále zabývat (spousta příkladů je uvedena v Chvátalově knize zmiňované v kapitole 8) a ukážeme si problémy, ve kterých má využití lineárního programování jiný charakter.

2.2 Tok v síti

Správce počítačové sítě přemluvil zaměstnavatele ke koupi nového stroje s kvalitnějšími reproduktory, a chce na něj po síti přenést svoji hudební

sbírku. Síť vypadá takto:



Jaké největší přenosové rychlosti může dosáhnout z počítače s (starý) na počítač n (nový)? Čísla u jednotlivých propojení udávají jejich přenosovou rychlost (třeba v MB/s). Předpokládáme také, že každým propojením se dají přenášet data oběma směry, ale ne zároveň. Například propojením ab se dají *buď* posílat data z a do b jakoukoli rychlostí mezi 0 a 1 MB/s, *nebo* posílat data z b do a jakoukoli rychlostí mezi 0 a 1 MB/s, ale obojí najednou nelze.

Uzly $a, b, \ldots, e$ nejsou uzpůsobeny ke skladování většího množství dat, takže všechna data, která do nich přicházejí, se musí ihned posílat dál. Z toho už je vidět, že nelze využít maximální kapacitu všech propojení, a pro každé propojení je tedy potřeba najít správnou hodnotu datového toku, aby celková přenosová rychlost z s do n byla co největší.

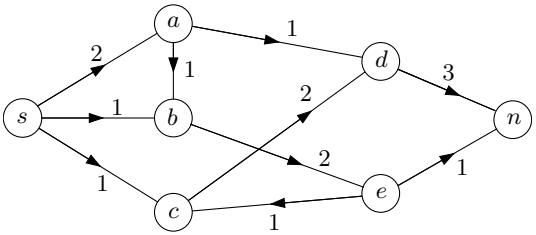
Pro každé propojení zavedeme jednu proměnnou. Například x_{be} udává, jakou rychlostí se data budou přenášet z b do e. Přitom x_{be} může být i záporné, což znamená, že data tečou obráceným směrem, z e do b. (Tudíž další proměnnou x_{eb}, která by udávala rychlost přenosu z e do b, už zavádět nepotřebujeme.) Budeme mít 10 proměnných x_{sa}, x_{sb}, x_{sc}, x_{ab}, x_{ad}, x_{be}, x_{cd}, x_{ce}, x_{dn} a x_{en}. Úloha lineárního programování bude:

$$
\begin{array}{ll}
\text{maximalizovat} & x_{sa} + x_{sb} + x_{sc} \\
\text{za podmínek} & -3 \le x_{sa} \le 3, \quad -1 \le x_{sb} \le 1, \quad -1 \le x_{sc} \le 1 \\
& -1 \le x_{ab} \le 1, \quad -1 \le x_{ad} \le 1, \quad -3 \le x_{be} \le 3 \\
& -4 \le x_{cd} \le 4, \quad -4 \le x_{ce} \le 4, \quad -4 \le x_{dn} \le 4 \\
& -1 \le x_{en} \le 1 \\
& \qquad x_{sa} = x_{ab} + x_{ad} \\
& x_{sb} + x_{ab} = x_{be} \\
& \qquad x_{sc} = x_{cd} + x_{ce} \\
& x_{ad} + x_{cd} = x_{dn} \\
& x_{be} + x_{ce} = x_{en}.
\end{array}
$$

Účelová funkce $x_{sa} + x_{sb} + x_{sc}$ udává, jakou celkovou rychlostí se data odesílají z počítače s. Poněvadž se nikde neukládají ani (doufejme) neztrácejí,

musí se stejnou rychlostí přijímat v n. Dalších 10 podmínek $-3 \leq x_{sa} \leq 3$ až $-1 \leq x_{en} \leq 1$ omezuje přenosové kapacity. Zbývající podmínky pak říkají, že cokoliv do každého z uzlů a až e přichází, musí také odcházet.

Optimální řešení této úlohy vypadá takto:



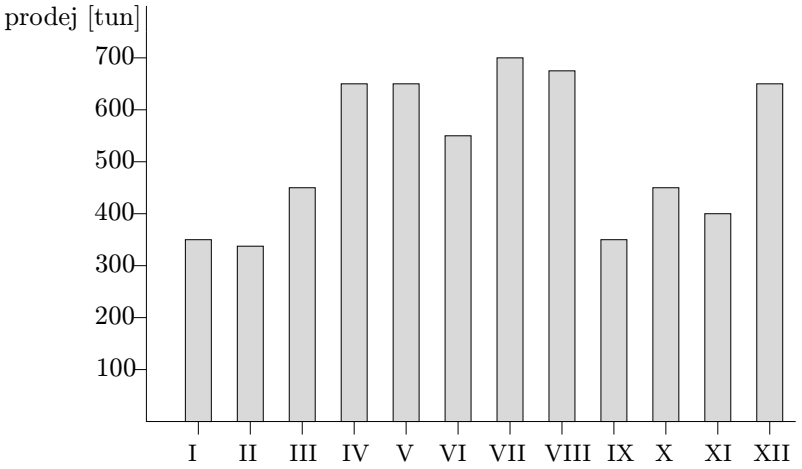
Číslo u každého propojení udává velikost příslušného datového toku, a šipka určuje jeho směr. Všimněte si, že mezi c a e je potřeba posílat data ve směru z e do c, takže $x_{ce} = -1$. Hodnota účelové funkce je 4, což je požadovaná maximální přenosová rychlost.

V tomto příkladu je snadno vidět, že přenosová rychlost nemůže být větší, protože celková kapacita propojení počítačů s a a se zbytkem sítě je 4. To je speciální případ pozoruhodné věty o maximálním toku a minimálním řezu, která se probírá v kombinatorice.

Uvedený příklad datového toku v síti je malý, jednoduchý a ne úplně realistický. V praxi se ovšem uvažují toky ve složitých sítích, dokonce s mnoha zdrojovými a cílovými uzly. Mohou to být například sítě elektrické (teče proud), silniční či železniční (tečou vlaky nebo auta), telefonní (tečou hlasové nebo datové signály), finanční (tečou peníze) a tak dále.

2.3 Zmrzlina po celý rok

Další aplikace lineárního programování souvisí opět s jídlem, což by vzhledem k důležitosti jídla v životě a složitosti optimalizace spánku nebo lásky nemělo být příliš překvapivé. Výrobce zmrzliny potřebuje sestavit plán výroby na další rok. Marketingové oddělení vydalo následující předpověď měsíční spotřeby zmrzliny v příštím roce založenou na datech z předchozích let, rozsáhlém průzkumu trhu a pozorování ptactva:



Jak má vypadat plán výroby, který takovouto poptávku uspokojí?

Jednoduchým řešením by byla produkce „právě včas“, tedy že zmrzlina spotřebovaná v měsíci i se v měsíci i také vyrobí. To ovšem znamená, že se objem výroby bude z měsíce na měsíc velice měnit, a takové změny znamenají nezanedbatelné náklady: bude potřeba najmout nebo propustit sezónní pracovníky, přenastavit stroje a tak dále. Takže by bylo lepší produkovat zmrzlinu rovnoměrněji během celého roku: v měsících s nízkou poptávkou by se nevyužitá kapacita továrny zaměstnala výrobou zmrzliny do zásoby na měsíce s vysokou poptávkou.

Jiným jednoduchým řešením by tedy mohl být naprosto „rovnoměrný“ plán výroby s týmž objemem v každém měsíci. Po chvíli uvažování se zjistí, že takový rozvrh nemusí být možný, pokud chceme na konci roku skončit bez přebytků. Ale i když je takové řešení uskutečnitelné, nemusí být ideální, protože skladováním zmrzliny také vznikají netriviální náklady. Nejvhodnější plán výroby patrně bude někde mezi těmito dvěma extrémy (výrobou sledující poptávku a výrobou neměnnou). Chceme najít kompromis, při kterém jsou celkové náklady na změny výrobní kapacity a na skladování přebytků minimální.

Abychom úlohu mohli zapsat formálně, zavedeme pro poptávku v měsíci i (v tunách) nezápornou proměnnou d_i. Dále zavedeme nezáporné proměnné x_i označující produkci v měsíci i a další nezáporné proměnné s_i pro celkovou skladovou zásobu na konci měsíce i. Na uspokojení poptávky v měsíci i se dá použít zmrzlina vyrobená v měsíci i a přebytky z měsíce $i-1$:

$$x_i + s_{i-1} \geq d_i \quad \text{pro } i = 1, 2, \ldots, 12.$$

Množství $x_i + s_{i-1} - d_i$ je přesně přebytek na konci měsíce i, a tedy

$$x_i + s_{i-1} - s_i = d_i \quad \text{pro } i = 1, 2, \ldots, 12.$$

Předpokládáme, že na začátku ledna nebyla žádná zásoba, a tudíž definujeme $s_0 = 0$. (Pokud bychom vzali v úvahu i minulou výrobu, s_0 by byl přebytek z předchozího roku.) Také dosadíme $s_{12} = 0$, pokud ovšem nechceme pokračovat v plánování i na další rok.

Chceme najít takové nezáporné řešení těchto rovnic a nerovnic, které odpovídá nejnižším celkovým nákladům. Předpokládejme, že změna produkce o 1 tunu mezi měsíci $i-1$ a i stojí 1500 Kč a skladování 1 tuny zmrzliny stojí 600 Kč na měsíc. Potom celkové náklady vyjadřuje funkce

$$1500 \sum_{i=1}^{12} |x_i - x_{i-1}| + 600 \sum_{i=1}^{12} s_i,$$

kde dosadíme $x_0 = 0$ (opět by se dala snadno vzít v úvahu i výroba z minulého roku).

Tato účelová funkce sice bohužel není lineární, ale naštěstí existuje jednoduchý (ale důležitý) trik, který nám ji umožní na lineární převést, a to za cenu zavedení dalších proměnných.

Změna v produkci je buď zvýšení nebo snížení. Zavedeme nezápornou proměnnou y_i pro zvýšení výroby v měsíci i oproti měsíci $i-1$ a nezápornou proměnnou z_i pro snížení. Potom

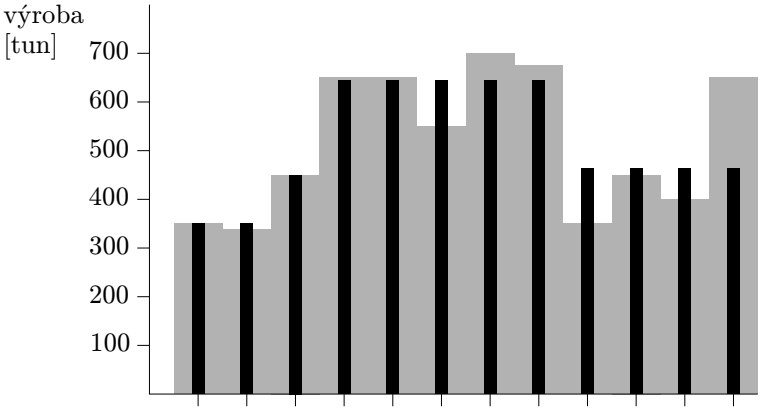
$$x_i - x_{i-1} = y_i - z_i \text{ a } |x_i - x_{i-1}| = y_i + z_i.$$

Plán výroby s minimálními celkovými náklady se tudíž dá najít jako optimální řešení následující úlohy lineárního programování:

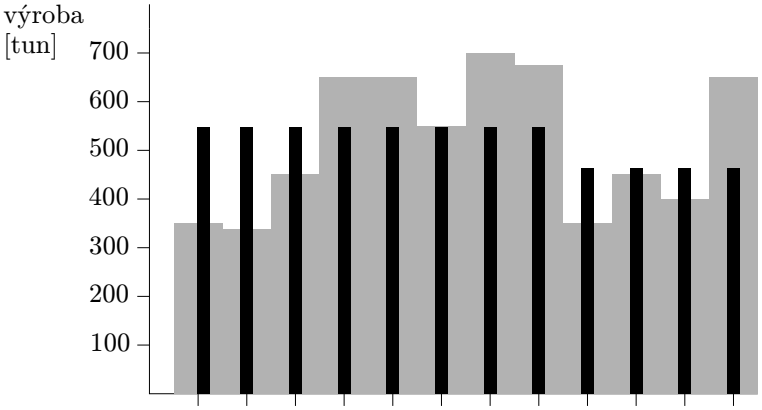
$$\begin{array}{ll}
\text{minimalizovat} & 1500 \sum_{i=1}^{12} y_i + 1500 \sum_{i=1}^{12} z_i + 600 \sum_{i=1}^{12} s_i \\
\text{za podmínek} & x_i + s_{i-1} - s_i = d_i \text{ pro } i = 1, 2, \ldots, 12 \\
& x_i - x_{i-1} = y_i - z_i \text{ pro } i = 1, 2, \ldots, 12 \\
& x_0 = 0 \\
& s_0 = 0 \\
& s_{12} = 0 \\
& x_i, s_i, y_i, z_i \geq 0 \text{ pro } i = 1, 2, \ldots, 12.
\end{array}$$

Abychom ověřili, že optimální řešení $(\mathbf{s}^*, \mathbf{y}^*, \mathbf{z}^*)$ této úlohy lineárního programování skutečně odpovídá plánu výroby, povšimneme si, že pro všechna i musí y_i^* nebo z_i^* být nulové, jinak by šlo obě zmenšit a dostat lepší řešení. To znamená, že $y_i^* + z_i^*$ se skutečně rovná změně produkce mezi měsíci $i-1$ a i.

Když tento lineární program vyřešíme pro výše uvedená zmrzlinářská data, dostaneme následující plán výroby (znázorněný černými sloupky, zatímco šedivý graf odpovídá poptávce).



Další obrázek ukazuje plán výroby, který bychom dostali při nulových nákladech na skladování (tj. po dosazení „0“ místo „600“ ve výše uvedené úloze lineárního programování).



Myšlenka uvedeného řešení je použitelná obecně pro mnoho problémů optimálního řízení. Pěkným příkladem je „Přistání měsíčního modulu“, kdysi populární hra pro programovatelné kalkulačky (nejspíš příliš prostá na to, aby přežila v dnešní konkurenci). Lunární modul s omezenou zásobou paliva klesá kolmo dolů k povrchu Měsíce pod vlivem gravitace a sestup

může řídit brzdnými raketovými motory. Cílem je přistát na povrchu (přibližně) nulovou rychlostí, dřív než dojde palivo. Čtenář může zkusit zformulovat úlohu určit minimálního množství paliva potřebného na měkké přistání pomocí lineárního programování. Přitom je potřeba předpokládat, že tah motorů se mění jen po určitých časových intervalech, řekněme po sekundách (ostatně i ve hře to tak bylo). Jsou-li časové intervaly dostatečně krátké, řešení úlohy se touto diskretizací času téměř nezmění.

Poznamenejme, že v případě přistání lunárního modulu se úloha dá vyřešit přesně pomocí diferenciálního počtu (nebo pomocí matematické teorie optimálního řízení). Ale v situacích jen mírně složitějších je už přesné analytické řešení neschůdné.

Na závěr zopakujme užitečný trik, který jsme v tomto oddílu předvedli.

> Optimalizační úlohy, které mají v účelové funkci nebo omezujících podmínkách výrazy s absolutními hodnotami, se často dají převést na úlohy lineárního programování vhodným zavedením dalších proměnných.

2.4 Oddělování bodů

Počítačem řízenou králičí past Gromit KP 2.1 chceme naprogramovat tak, aby chytala králíky, ale aby lasičky, které náhodou zabloudí dovnitř, pouštěla zase ven. Past umí chycené zvíře zvážit a také určit plochu jeho stínu.

Na následujícím grafu jsou zobrazena data naměřená pro vzorek králíků a lasiček: