1 Úloha

Úloha celočíselného programování je prakticky stejná jako úloha lineárního programování. Navíc se ale požaduje, aby řešení některých proměnných ve vektoru x bylo celočíselné nebo dokonce binární. Celočíselné veličiny většinou reprezentují počet (kusů, lidí apod.) Binární veličiny lze interpretovat jako rozhodování $1 = ano$, $0 = ne$. Tím se rozsah úloh velmi zvětšuje.

2 Formulace úlohy

Nalezněte hodnoty vektoru $x = [x_1, x_2, \cdots, x_n]$, které

(i) maximalizují lineární kriterium $c'x = c_1x_1 + c_2x_2 + \cdots + c_nx_n$,

(ii) splňují vedlejší podmínky $Ax \leq b$, tedy $a_{i,1}x_1 + a_{i,2}x_2 + \cdots + a_{i,n}x_n \leq b_i$, pro $i = 1, 2, \cdots, m$ (m podmínek pro n proměnných) a

(iii) alespoň jedna veličina v x je celočíselná.

Zápis standardní úlohy celočíselného programování je následující:

$$c'x \to \text{opt}$$
$$Ax \leq b$$
$$x_{I_1} \geq 0,\ x_{I_2} \in \mathbb{N}$$

kde „opt" označuje maximum nebo minimum, x_{I_1} množina veličin pro které vyžadujeme nezápornost a x_{I_2} je množina veličin, které mají být celočíselné; $x_{I_1} \cup x_{I_2} = x$.

Příklad

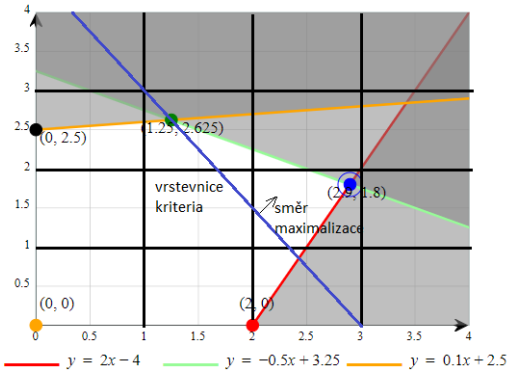
Najděte optimální řešení úlohy

$$3x_1 + 2x_2 \to \max$$
$$2x_1 - x_2 \leq 4$$
$$2x_1 + 4x_2 \leq 13$$
$$-x_1 + 10x_2 \leq 25$$
$$x_1 \geq 0,\ x_2 \in N$$

Řešení najdeme v Excelu

	A	B	C	D	E	F
1	Úvodní příklad					
2		R	N			
3	x	2.5	2			
4						
5	c	3	2			
6				Ax	b	
7	A	2	-1	3	4	
8		2	4	13	13	
9		-1	10	17.5	25	
10						
11	J	11.5				
12						
13	Řešení					
14	x2 z N	2.5	2		J = 11.5	
15	x2 z R	2.9	1.8		J = 12.3	
16						

Dole na obrázku jsou uvedena obě řešení, jak pro $x_2 \in N$, tak i pro $x_2 \geq 0$. Řešení můžeme srovnat s grafickým vyjádřením úlohy



Tady vidíme, že je to tak :-)

Poznámka

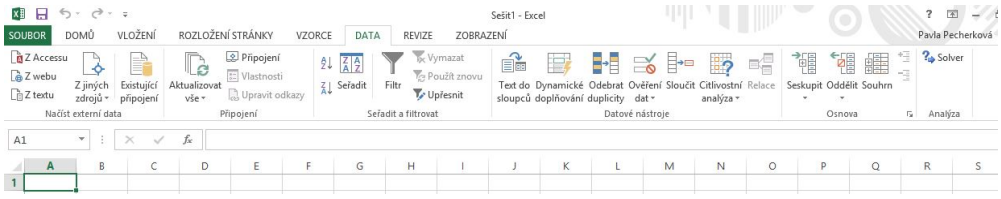
O řešení úloh lineárního programování jsme mluvili v kurzu LP1. Ti, kdo zapomněli, jak se úloha LP v Excelu realizuje, najdou podrobný návod v kapitole **??**.

3 Řešení úlohy

3.1 Excel - řešitel

Pro řešení úloh lineárního programování je možné použít aplikaci Řešitel, tedy aplikaci, která slouží k vyhledávání extrémů funkce. Řešitel je součástí MS Excel. Výhoda této aplikace je v tom, že je přehledná a Excel je běžně využívaný a rozšířený tabulkový procesor. Nevýhoda je, že je omezený prostorem a pro složité úlohy se musí použít jiný program.

Všechny úlohy byly naprogramovány ve verzi MS Excel 2013 a i na tuto verzi optimalizovány. Z toho důvodu neručíme za nefunkčnost ve starších verzích. Tato verze je zdarma ke stažení na http://download.cvut.cz/. Řešitele naleznete po spuštění v záložce *DATA*, viz obrázek 3.1.

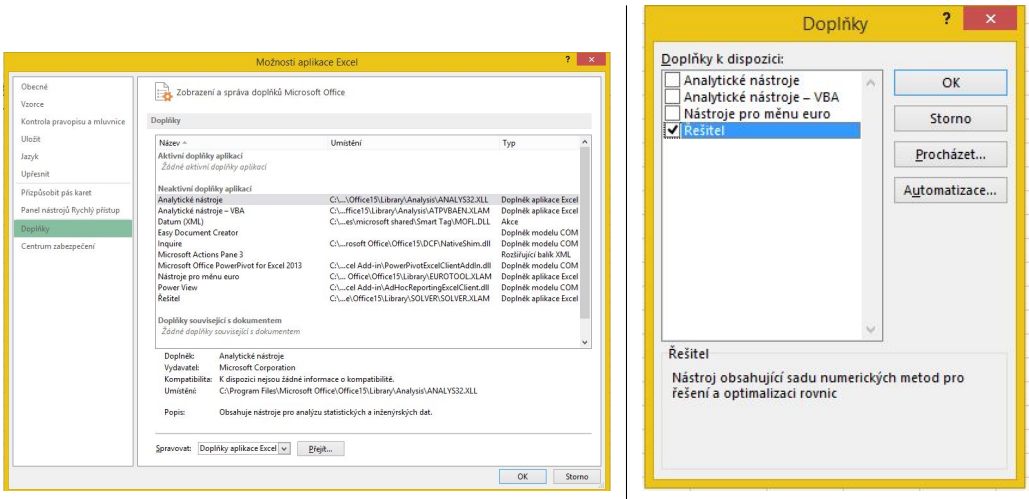


Obrázek 3.1: Excel - Data - Řešitel (Solver)

Pokud tam není, je nutné ho aktivovat. Lze aktivovat následujícím způsobem:

1. otevřete *"Soubor - Možnosti - Doplňky"* a otevře se vám nové okno, viz obrázek 3.2 (a),
2. v otevřeném okně dole klikneme na *"Spravovat: Doplňky aplikace Excel"* a otevře se další okno, viz obrázek 3.2 (b). Zaškrtneme *"Řešitel"* a potvrdíme pomocí tlačítka *"OK"*.

Pokud vše proběhlo v pořádku, v záložce *DATA* se objeví Řešitel (v anglické verzi Solver).



(a) (b)

Obrázek 3.2: Aktivace Řešitele

3.2 Model v sešitě Excel

1. Nejdříve vymezíme blok buněk pro neznámé x (případně další neznámé).
2. Dále zapíšeme všechny zadané veličiny - většinou to jsou ceny c a koeficienty lineárních omezení A a požadované pravé strany b.
3. Spočteme všechno, co je potřeba spočítat (do Řešitele se dávají jen odkazy na buňky nebo bloky buněk). Většinou to je:
   (a) kritérium $c'x = \sum_i c_i x_i$
   (b) vypočtené pravé strany omezení $Ax = \sum_j a_{ij} x_j \ \forall i$
4. Zavoláme Řešitele a zadáme všechny odkazy
   (a) zvolíme Min nebo Max pro kritérium,
   (b) zadáme blok nebo bloky pro neznámé (optimalizované) veličiny,
   (c) přidáme omezení ve tvaru $\leq$, $\geq$ nebo binární (volí se na stejném místě)
   (d) většinou necháme zatrženou volbu "neomezené veličiny jsou nezáporné" (nemusí se to již deklarovat v podmínkách)
   (e) a v okénku dole vybereme volbu "Simplex LP" - lineární programování.

3.3 Triky při práci v Excelu

Blok buněk - tažení myší nebo `Shift+šipka`.

Přemístění bloku (s `Ctrl` kopie) - uchopení za okraj (buňky nebo bloku) a tažení.

Vkopírování hodnoty do celého bloku - vytvoříme blok, zadáme hodnotu a `Ctrl+Enter`.

Maticové vzorce - provádějí operace (nejčastěji násobení) prvek po prvku. Zadávají se pomocí `Ctrl+Shift+Enter`. Výsledek může být v jedné buňce, nebo v bloku buněk. Blok je možno zadat

předem (pak se objeví výsledky v celém bloku). Vzorec je možno kopírovat se všemi pravidly o posunu adres.

Fixace adres (absolutní adresy) - zafixované adresy se při kopírování neposouvají. Adresu fixuje dolar před písmenem, číslem nebo obojím. Dolar před písmenem fixuje adresu při vodorovném pohybu, před číslem při svislém pohybu a před obojím i ve vodorovném i svislém směru.

Zadání dolarů - automaticky zadáme dolary klávesou F4 ihned po vložení adresy (jinak se musí adresa dát do bloku). Opakované stisknutí F4 provede: oba dolary, jen před číslem, jen před písmenem, žádný dolar (atd.).

Skalární součin - pokud potřebujeme maticový výpočet $= \sum_i a_i b_i$ lze ho jednoduše vytvořit tak, že vytvoříme sumu součinu dvou sloupců a poté zmáčkneme `Ctrl+Shift+Enter`. Například `=suma(A1:A10*B1:B10) + Ctrl+Shift+Enter.`

Součin matice a vektoru - matice je `B1:D10` a vektor `A1:A10`. Součin je `=suma(B1:D10*$A$1:$A$10) + Ctrl+Shift+Enter` a rozkopírovat svisle. (Adresa s dolary je absolutní - viz fixace adres.)

3.4 Příklad v Excelu

Řešíme obecný příklad lineárního programování

$$5x_1 + x_2 \to \max$$

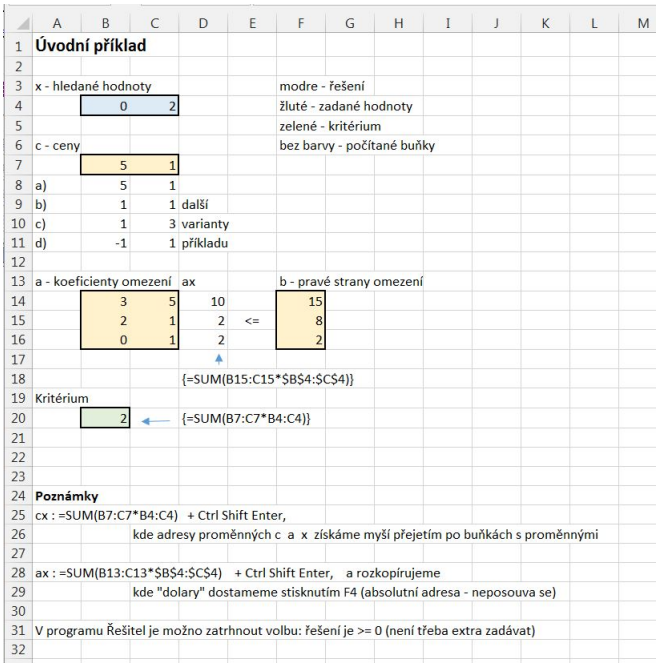
$$\begin{aligned} 3x_1 + 5x_2 &\leq 15 \\ 2x_1 + x_2 &\leq 8 \\ x_2 &\leq 2 \end{aligned}$$

$$x_1, x_2 \geq 0$$

Pro přehlednost, je hned na začátku uvedeno řešení, které je označeno modrým pozadím (obrázek 3.3). Dále jsou uvedeny váhy v kritériu (pro variantu (a) v obecném příkladu), tedy váhy $c = [5, 1]$. Následují podmínky jako matice

$$A = \begin{bmatrix} 3 & 5 \\ 2 & 1 \\ 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 15 \\ 8 \\ 2 \end{bmatrix}$$

které jsou také ve čtverečku s žlutým pozadím, stejně jako váhy v kritériu. Kritérium, které je označeno zeleným pozadím v rámečku, je základní parametr, kde se hledá maximum nebo minimum. Ani tento parametr neměníme. Při práci se obvykle mění pouze políčka s žlutým pozadím, tedy zadání.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Úvodní příklad**												
2													
3	x - hledané hodnoty					modre - řešení							
4		0	2			žluté - zadané hodnoty							
5						zelené - kritérium							
6	c - ceny					bez barvy - počítané buňky							
7		5	1										
8	a)	5	1										
9	b)	1	1	další									
10	c)	1	3	varianty									
11	d)	-1	1	příkladu									
12													
13	a - koeficienty omezení			ax		b - pravé strany omezení							
14		3	5	10		15							
15		2	1	2	<=	8							
16		0	1	2		2							
17													
18				{=SUM(B15:C15*$B$4:$C$4)}									
19	Kritérium												
20		2	←	{=SUM(B7:C7*B4:C4)}									
21													
22													
23													
24	**Poznámky**												
25	cx : =SUM(B7:C7*B4:C4) + Ctrl Shift Enter,												
26	kde adresy proměnných c a x získáme myší přejetím po buňkách s proměnnými												
27													
28	ax : =SUM(B13:C13*$B$4:$C$4) + Ctrl Shift Enter, a rozkopírujeme												
29	kde "dolary" dostameme stisknutím F4 (absolutní adresa - neposouva se)												
30													
31	V programu Řešitel je možno zatrhnout volbu: řešení je >= 0 (není třeba extra zadávat)												
32													

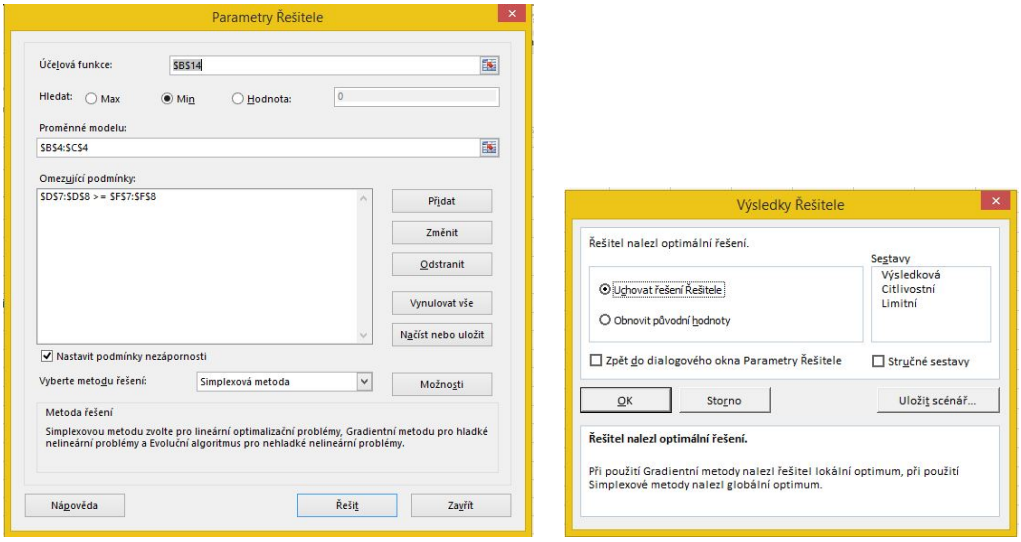
Obrázek 3.3: Příklad v Excelu

V případě, že již máme nastavené základní hodnoty, přecházíme k Řešiteli (Solver). Spustíme DATA-Řešitel (Solver) a objeví se nám tabulka znázorněna na obrázku 3.4 (a). Pro správnou funkci je potřeba nastavit následující:

1. **Účelová funkce** - odkaz na buňku s kritériem (zelené pozadí).
2. **Hledat** - hledá se minimum, maximum nebo určitá hodnota. Pro zadávajícího je důležité si uvědomit co hledá (u zisku bude hledat maximum a u nákladů minimum).
3. **Proměnné modelu** - výsledek (modré pozadí). Zde bude optimální hodnota pro zadané veličiny (například kolik kusů výrobku A a B vyrobit).
4. **Omezující podmínky** - zde musí být všechny podmínky, které je potřeba splnit, tzn. vytvoří se sloupcový vektor, do kterého se vypočítá součin výsledku (například kolik kusů vyrobit) s hodnotou řádku tak, aby se dal výsledek porovnat s omezením. Tedy „$a \cdot x$" porovnáme s pravou strannou omezení. Pokud bude ještě jiný požadavek, například minimální vyrobené množství, musí se zapsat také do Omezujících podmínek, kde se klikne na ikonku Přidat a nadefinuje se nová podmínka.
5. **Nastavit podmínky nezápornosti** - u většiny problémů se očekává, že výsledek nemůže být záporný (nemůžeme vyrobit -5 výrobků). Tato podmínka se může zadat přímo do omezujících podmínek nebo se zaškrtne políčko s podmínkou nezápornosti.
6. **Vyberte metou řešení** - na výběr je Gradientní metoda, Simplexová metoda a Evoluční algoritmus. Zvolení vhodné metody je na zadavateli a určí se podle typu úlohy.
   (a) Simplexová metoda - lineární optimalizační úlohy,

(b) Gradientní metoda - hladké nelineární optimalizační úlohy,

(c) Evoluční algoritmus - nehladké nelineární optimalizační úlohy.



(a) (b)

Obrázek 3.4: Řešitel

Poté se již jen zadá funkce řešit a objeví se následující tabulka 3.4 (b). Tam necháme zaškrtnutou možnost **Uchovat řešení Řešitele** a potvrdit tlačítkem **OK**.

3.5 Linear programming grapher

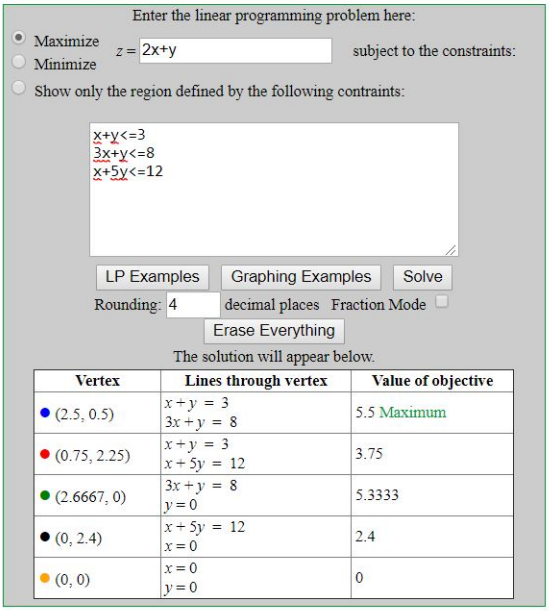
Jedná se o webovou aplikaci na adrese

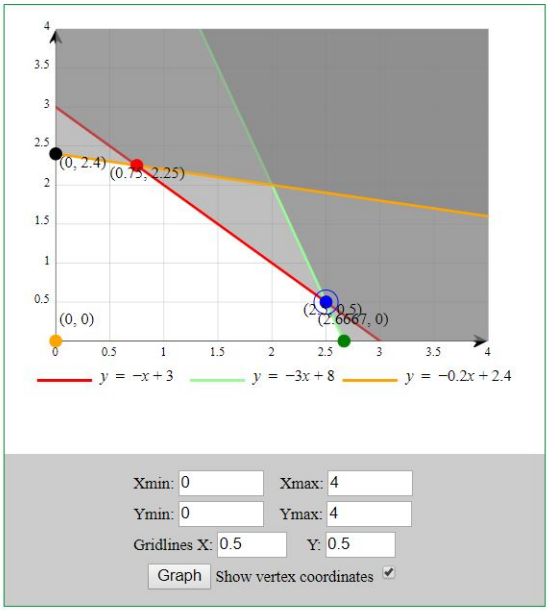
https://www.zweigmedia.com/utilities/lpg/index.html?lang=en

do které lze zadat model dvourozměrné úlohy lineárního programování a úlohu spustit tlačítkem Solve. Ukáže se přípustný simplex řešení s popisem jednotlivých hranic omezení a samozřejmě optimální řešení. Model se zadá nejlépe tak, že spustíte LP Examples a výsledek upravíte podle svého.

Vlevo nahoře, pod žlutou záložkou Main Page, je tlačítko Simplex method utility. Ta umožňuje řešit obecnou úlohu LP. Doporučuje se pracovat v nové verzi (červený text).

Pohled na Grapher je na následujícím obrázku





Sbírka příkladů (Bude se postupně rozšiřovat. Každý nový příspěvek je vítán.)

4 Triky v omezeních

Zavedení celočíselných/binárních proměnných má dvojí využití:

- Zavádí popis veličin, které nelze dělit - např. počet najatých dělníků nebo počet objednaných krabic se zbožím.
- Umožňuje využití programovacích triků, které se zejména týkají možnosti rozhodování. Např. pro binární $y \in \{0, 1\}$ je „situace nastane“ pro $y = 1$ a „situace nenastane“ pro $y = 0$. Také se často využívá pro výběr: pro binární vektor $y = \{y_1, y_2, \cdots, y_n\}$ vybereme ty akce pro které je $y_i = 1$.

Dále si ukážeme některé základní triky a jejich realizaci.

4.1 Binární veličiny

Binární veličiny mohou nabývat dvou hodnot, 0 nebo 1. Souvisí s rozhodováním: něco bude $\rightarrow$ 1, nebo nebude $\rightarrow$ 0.

Poznámka

Ne vždycky 1 znamená ano a 0 ne. Uvidíme později u aktivace omezení. Je třeba na to dávat pozor.

Pokud máme n takových veličin $x_1, x_2, \cdots, x_n \in \{0, 1\}$, lze realizovat jednu z následujících podmínek:

- **alespoň** k bude splněno (k nebo více): $\sum_{i=1}^{n} x_i \geq k$,
- **právě** k bude splněno: $\sum_{i=1}^{n} x_i = k$,
- **nejvýše** k bude splněno (k nebo méně): $\sum_{i=1}^{n} x_i \leq k$.

Příklad

Chceme investovat 30tis Kč a je k dispozici 5 příležitostí. Hodnoty investic na pořízení akce a očekávané výnosy (v tis. Kč) jsou v tabulce

akce	1	2	3	4	5
náklad (n)	5	3	8	10	7
výnos (v)	14	14	16	19	15

Které akce vybereme, abychom maximálně vydělali, jestliže smíme zvolit nejvýše 3 akce?

Řešení

Zvolíme indikátorový vektor $y \in \{0, 1\}$ - binární jako stavový vektor.

Kriterium

$$J = \sum y_i \left(v_i - n_i\right) \to \max$$

Omezení

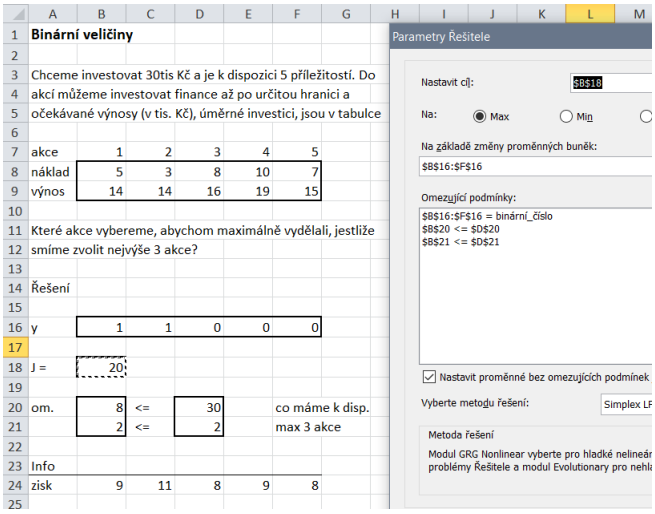
– k dispozici 30tis.

$$\sum y_i n_i \leq 30$$

– max. 3 akce

$$\sum y_i \leq 3$$

Excel: P01_alespon.xlsx



4.2 Indikace nenuly

Máme spojitou veličinu x a ve výpočtech chceme rozlišit dva případy: a) $x = 0$ a b) $x > 0$.

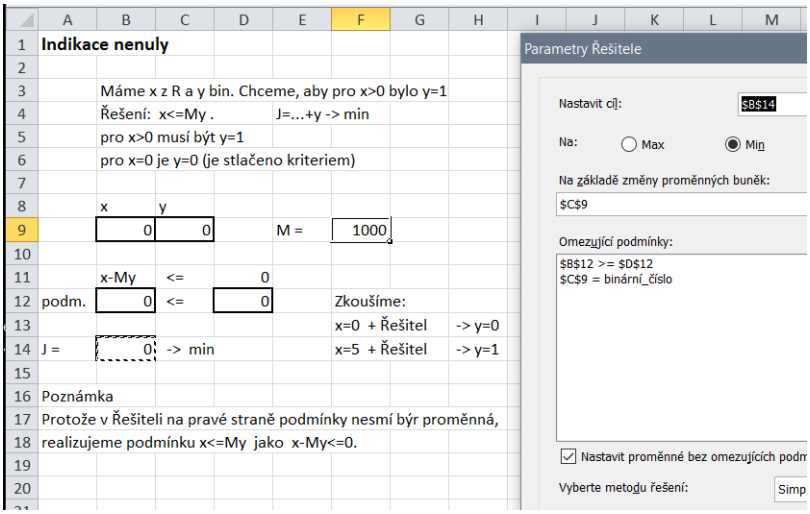
Definujeme $y \in \{0, 1\}$ - binární, indikátor nenulovosti x a zavedeme podmínku

$$My \geq x$$

kde M je hodně velké číslo (větší než maximum x).

Potom, je-li $x > 0$ musí být $y = 1$. Bude-li $x = 0$, podmínka připouští cokoli. Nicméně, předpokládá se, že minimalizace kriteria, ve kterém y figuruje, jeho hodnotu bude tlačit k nule.

Excel: Pr02_indikace1.xlsx



Příklad

Chceme investovat 30tis Kč a je k dispozici 5 příležitostí. Do akcí můžeme investovat finance až po určitou hranici. Tyto hranice (v tis. Kč) a koeficienty očekávaných výnosů, které jsou úměrné investici, jsou v tabulce

akce	1	2	3	4	5
max. investice (m)	15	30	18	20	27
relativní výnos (v)	1.15	1.12	1.18	1.16	1.11
náklady na inv. (n)	5	8	7	8	6

Které akce vybereme, abychom docílili maximální zisk, jestliže smíme zvolit nejvýše 3 akce?

Řešení

Ptáme se do kterých akcí a kolik máme investovat. Pro otázku kolik zavedeme vektor $x \leq 0$ a pro otázku do kterých zavedeme indikátor $y \in \{0, 1\}$-binární

$$x = [x_1, x_2, \cdots, x_5] \in R_0^+$$

$$y = [y_1, y_2, \cdots, y_5] \in \{0, 1\}$$

Kriterium

$$J = \sum_i v_i x_i - \sum_i n_i y_i \to \max$$

Omezení

– omezení investic

$$x_i \leq m_i, \forall i$$

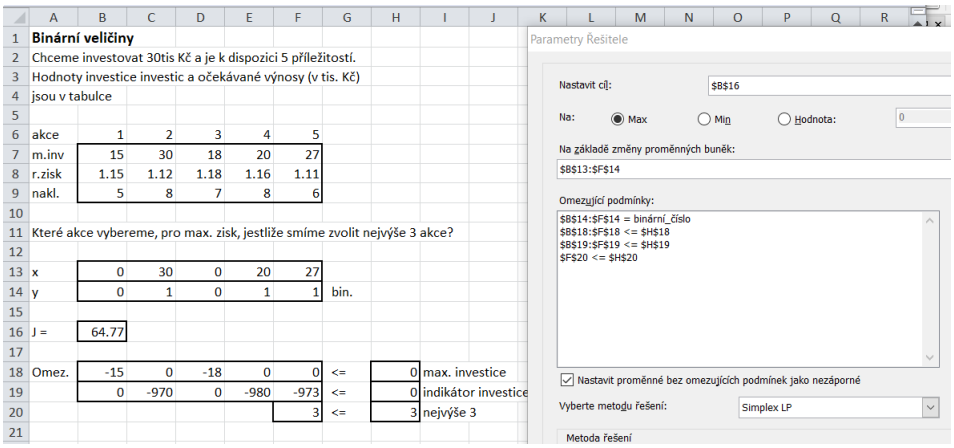
– indikátor pro $x > 0$

$$x_i \leq M y_i, \forall i$$

– max. 3 akce

$$\sum_i y_i \leq 3$$

Excel: Pr02_indikace2.xlsx



	A	B	C	D	E	F	G	H	I
1	**Binární veličiny**								
2	Chceme investovat 30tis Kč a je k dispozici 5 příležitostí.								
3	Hodnoty investice investic a očekávané výnosy (v tis. Kč)								
4	jsou v tabulce								
5									
6	akce	1	2	3	4	5			
7	m.inv	15	30	18	20	27			
8	r.zisk	1.15	1.12	1.18	1.16	1.11			
9	nakl.	5	8	7	8	6			
10									
11	Které akce vybereme, pro max. zisk, jestliže smíme zvolit nejvýše 3 akce?								
12									
13	x	0	30	0	20	27			
14	y	0	1	0	1	1	bin.		
15									
16	J =	64.77							
17									
18	Omez.	-15	0	-18	0	0	<=	0	max. investice
19		0	-970	0	-980	-973	<=	0	indikátor investice
20						3	<=	3	nejvýše 3
21									

4.3 Aktivace omezení

Uvažujme binární proměnnou $y \in \{0, 1\}$ a podmínku

$$a'x \leq b$$

. Zavedeme konstantu M jako „hodně velké číslo“ (musí být větší než $a'x - b$, $\forall x$). Většinou stačí $M = 1000$.

Zkonstruujeme další podmínku

$$a'x \leq My + b.$$

Tato podmínka říká

- je-li $y = 1$, pak původní podmínka je vždy splněna (původní podmínka je vyřazena)

$$a'x \leq M \underbrace{1}_{y=1} + b \;\Rightarrow\; a'x - b \leq M$$

- je-li $y = 0$, pak původní podmínka zůstává v platnosti

$$a'x \leq M \cdot \underbrace{0}_{y=0} + b \quad \Rightarrow \quad a'x \leq b.$$

Hodnota binární proměnné y tedy aktivuje nebo deaktivuje původní podmínku $a'x \leq b$.

POZOR: Hodnota $y = 0$ aktivuje a $y = 1$ deaktivuje. Tedy podmínka platí pro $y = 0$. Pokud bychom chtěli zachovat aktivaci pro $y = 1$, byla by podmínka $a'x \leq M(1 - y) + b$.

Příklad

Chceme investovat 30tis Kč a je k dispozici 5 příležitostí. Hodnoty investic na pořízení akce a očekávané výnosy (v tis. Kč) jsou v tabulce

akce	1	2	3	4	5
náklad (n)	11	5	8	7	9
výnos (v)	12	14	16	19	15

Jestliže vybereme první akci, dostaneme navíc ještě 5tis. Kč. Které akce vybereme, abychom maximálně vydělali (měli maximální zisk), jestliže smíme zvolit nejvýše 4 akce?

Řešení

Jako stavovou proměnnou zvolíme binární vektor $y \in \{0, 1\}$

$$y = [y_1, y_2, y_3, y_4, y_5]$$

který bude indikovat vybrané akce.

Kriterium

$$J = \sum_i y_i \left(v_i - n_i\right)$$

Omezení

– finance k dispozici pro $y_1 = 0$ (pro $y_1 = 1$ je deaktivováno)

$$\left(\sum_i y_i n_i\right) - M y_1 \leq 30$$

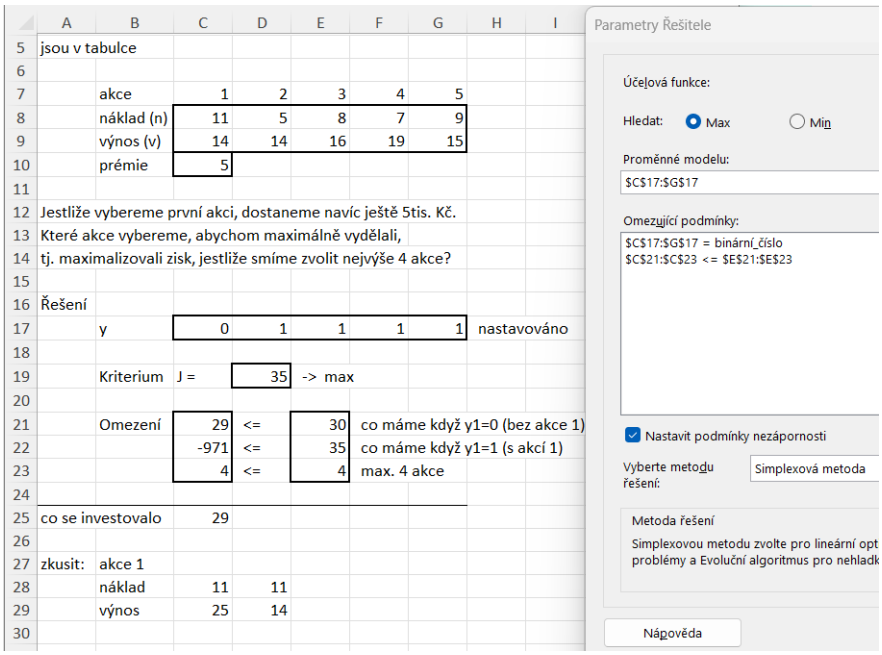
– finance k dispozici pro $y_1 = 1$ (pro $y_1 = 0$ je deaktivováno)

$$\left(\sum_i y_i n_i\right) - M\left(1 - y_1\right) \leq 35$$

– max. 4 akce

$$\sum_i y_i \leq 4$$

Excel: Pr03_aktivace.xlsx



4.4 Implikované omezení

Máme dvě omezení $A : a_1'x < b_1$ a $B : a_2'x \leq b_2$. Chceme aby platilo: když platí první omezení, pak platí i druhé (když první neplatí, tak nic nepožadujeme). Toto pravidlo lze popsat implikací[1] ($\Rightarrow$) nebo alternativou[2] ($\vee$). Ekvivalenci $(A \Rightarrow B) \equiv (A^\neg \vee B)$ ukazuje tabulka

A	B	$A \Rightarrow B$	$A^\neg$	B	$A^\neg \vee B$
0	0	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	1	1	0	1	1

kde $^\neg$ označuje negaci.

Odtud vidíme, že $(A \Rightarrow B)$ je stejné jako $(A^\neg \vee B)$, přičemž alternativu umíme - viz dříve (odstavec 4.1).

Poznámka: *Podmínku „omezení platí" realizujeme jako „omezení je aktivní" - viz odstavec 4.3.*

Negace $\left(a_1'x < b_1\right)^\neg = a_1'x \geq b$, a tedy

$$\left(a_1'x < b_1\right) \Rightarrow \left(a_2'x \leq b_2\right) \iff \left(a_1'x \geq b\right) \vee \left(a_2'x \leq b_2\right)$$

Realizujeme jako alternativu podle předchozího návodu (alespoň jedno omezení musí být aktivováno, tj. alespoň jedno y musí být 0, tedy 0,0 nebo 0,1 nebo 1,0, a tedy $\sum y \leq 1$)

$$\begin{aligned} a_1'x + B_1 y_1 &\geq b_1 \\ a_2'x - B_2 y_2 &\leq b_2 \\ y_1 + y_2 &\leq 1 \end{aligned}$$

[1] spojení „jestliže-pak"
[2] spojka „nebo"

Poznámka

Pozor na negaci! *V našem příkladě skutečně je* $\left(a_1' x < b_1\right)^{\neg} = a_1' x \geq b$. *Pokud bychom měli podmínku* $a_1 x \leq b_1$ *asi bychom mohli postupovat stejně. Chybu bychom udělali jen pro* $a_1 x = b_1$, *což je splněno jen pro jedinou hodnotu* x, *a tedy na množině míry nula. Nakonec bychom tuto hodnotu mohli konfrontovat s konečným řešením.*

Jiná situace je ale, pokud by se podmínka týkala celočíselné proměnné y. *Například* $y \leq 5$. *Negací této podmínka bude* $y \geq 6$.

Pro binární proměnnou y *často požadujeme např.* $y = 1$ *(tedy něco bude vybráno). Tuto podmínku lze zapsat jako* $y \geq 1$ *a její negace bude* $y \leq 0$.

Realizaci podmínky implikace $(y_1 = 1) \Rightarrow (y_2 = 1)$ *pro binární* y *lze realizovat velmi jednoduše jako* $y_2 \geq y_1$. *Skutečně*

y_1	y_2	$y_1 \Rightarrow y_2$	$y_2 \geq y_1$
0	*0*	*1*	*1*
0	*1*	*1*	*1*
1	*0*	*0*	*0*
1	*1*	*1*	*1*

Příklad

Uvažujeme o realizaci některých z pěti různých akcí. Specifikace je dána v tabulce.

Akce	1	2	3	4	5
Náklady (n)	89	61	77	64	57
Výnosy (v)	102	87	89	81	69

Platí podmínka: jestliže vybereme buď 1. nebo 2. akci, musíme vybrat také 4. nebo 5. akci. Jak máme akce vybrat, abychom dosáhli maximální zisk, jestliže smíme vybrat maximálně 3 akce?

Řešení

Zavedeme dva binární vektory

$$x = [x_1, x_2, x_3, x_4, x_5] \text{ bin}$$

který indikuje vybrané akce a

$$y = [y_1, y_2] \text{ bin}$$

který bude použit v realizaci implikace.

Kriterium

$$J = \sum x_i \left(v_i - n_i\right) \rightarrow \max$$

Omezení

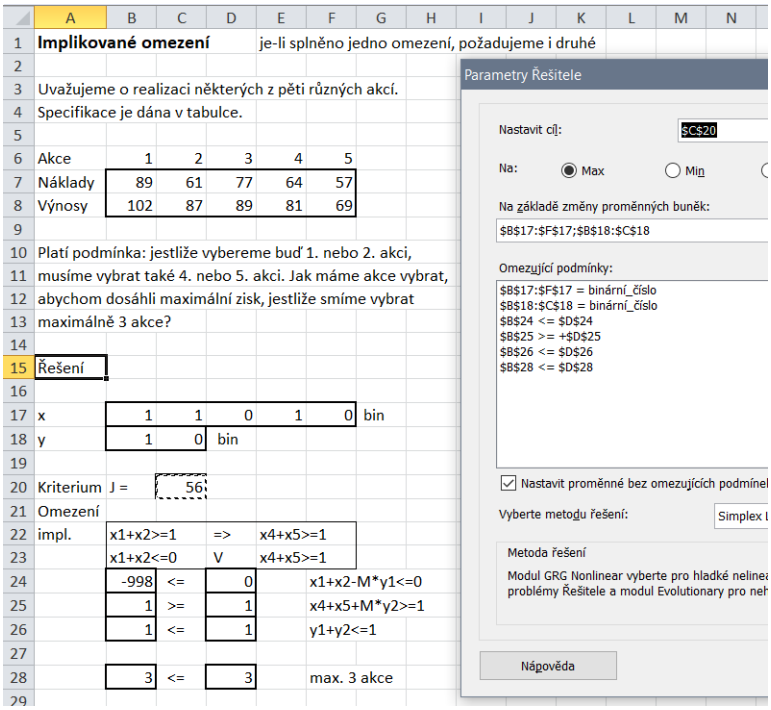
- implikace $x_1 + x_2 \geq 1 \Rightarrow x_4 + x_5 \geq 1$
  realizace $x_1 + x_2 \leq 0 \vee x_4 + x_5 \geq 1$
  program:

$$x_1 + x_2 - My_1 <= 0$$
$$x_4 + x_5 + My_2 \geq 1$$
$$y_1 + y_2 \leq 1$$

- max. 3 akce

$$\sum x_i \leq 3$$

Excel: Pr04_implikace.xlsx.



	A	B	C	D	E	F	G
1	**Implikované omezení**				je-li splněno jedno omezení, požadujeme i druhé		
2							
3	Uvažujeme o realizaci některých z pěti různých akcí.						
4	Specifikace je dána v tabulce.						
5							
6	Akce	1	2	3	4	5	
7	Náklady	89	61	77	64	57	
8	Výnosy	102	87	89	81	69	
9							
10	Platí podmínka: jestliže vybereme buď 1. nebo 2. akci,						
11	musíme vybrat také 4. nebo 5. akci. Jak máme akce vybrat,						
12	abychom dosáhli maximální zisk, jestliže smíme vybrat						
13	maximálně 3 akce?						
14							
15	Řešení						
16							
17	x	1	1	0	1	0	bin
18	y	1	0	bin			
19							
20	Kriterium	J =	56				
21	Omezení						
22	impl.	x1+x2>=1		=>	x4+x5>=1		
23		x1+x2<=0		V	x4+x5>=1		
24		-998	<=	0		x1+x2-M*y1<=0	
25		1	>=	1		x4+x5+M*y2>=1	
26		1	<=	1		y1+y2<=1	
27							
28		3	<=	3		max. 3 akce	
29							

4.5 Omezení na oblasti

Nechť omezení tvoří několik různých oblastí (disjunktních nebo i překrývajících se). Jednotlivé oblasti jsou popsány pomocí lineárních omezení.

Ukažme si toto omezení na následujícím příkladě, kdy platí:

$$\begin{aligned} &\text{první oblast} && a_1'x \leq b_1,\ \ a_2'x \leq b_2,\ \ a_3'x \leq b_3 \\ &\text{druhá oblast} && a_4'x \leq b_4,\ \ a_5'x \leq b_5 \\ &\ldots \end{aligned}$$

Chceme zaručit, že optimální bod řešení bude ležet alespoň v jedné z oblastí.

Modelujeme takto

1. oblast

$$\begin{aligned} a_1'x - M_1 y_1 &\leq b_1, \\ a_2'x - M_2 y_1 &\leq b_2, \\ a_3'x - M_3 y_1 &\leq b_3 \end{aligned}$$

2. oblast

$$a_4^{'}x - M_4 y_2 \leq b_4,$$
$$a_5^{'}x - M_5 y_2 \leq b_5$$

3. oblast ....

Zároveň musí platit, že

$$\sum_{j=1}^{k} y_j \leq k - 1$$

kde k je celkový počet oblastí a $y_j \in \{0, 1\}$. Konstanta B může být jediná, pokud je větší než všechny levé strany podmínek.

Příklad

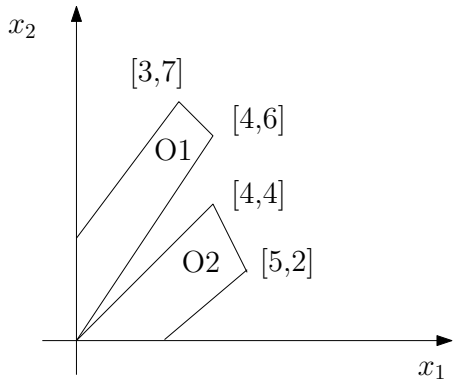
Máme dvě oblasti ohraničené přímkami

$$x_2 = \frac{4}{3}x_1 + 3, \;\; x_2 = -x_1 + 10, \;\; x_2 = \frac{3}{2}x_1$$

a

$$x_2 = x_1, \;\; x_2 = -2x_1 + 12, \;\; x_2 = \frac{4}{5}x_1 - 2$$

podle obrázku



V jedné z oblastí chceme koupit nemovitost a uložit do ní své peníze. Ceny nemovitostí rostou se vzdáleností od počátku, a to tak, že vodorovně je cena $15x_1$ a svisle $8x_2$. Kde máme nakoupit, abychom uložili co nejvíce peněz?

Řešení

První oblast je vymezena nerovnostmi

$$x_2 \leq \frac{4}{3}x_1 + 3, \;\; x_2 \leq -x_1 + 10, \;\; x_2 \geq \frac{3}{2}x_1$$

druhá oblast

$$x_2 \leq x_1, \;\; x_2 \leq -2x_1 + 12, \;\; x_2 \geq \frac{4}{5}x_1 - 2$$

Stavové vektory zavedeme jako $x = [x_1, x_2]$, $x_i \in R_0^+$ a $y = [y_1, y_2]$, $y_i \in \{0, 1\}$. x označuje souřadnice bodu, představující jednotlivé nemovitosti a y je indikátor oblasti.

Kriterium

$$J = 15x_1 + 8x_2 \rightarrow \max$$

Omezení

$$x_2 - My_1 \leq \frac{4}{3}x_1 + 3$$
$$x_2 - My_1 \leq -x_1 + 10$$
$$x_2 + My_1 \geq \frac{3}{2}x_1$$

pro první oblast a

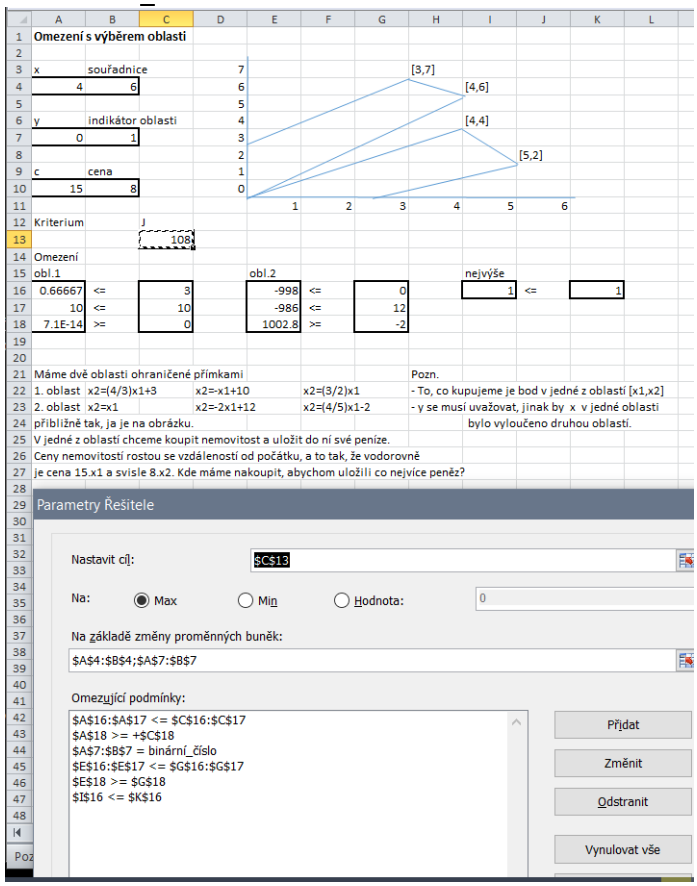
$$x_2 - My_2 \leq x_1$$
$$x_2 - My_2 \leq -2x_1 + 12$$
$$x_2 + My_2 \geq \frac{4}{5}x_1 - 2$$

pro druhou oblast. Pro realizaci je třeba podmínka upravit tak, aby na pravé straně bylo jen číslo.

Aktivní ($y = 0$) může být vždy nejvýše jedna oblast, proto

$$y_1 + y_2 \leq 1$$

Excel: Pr05_oblasti.xlsx



5 Triky v kriteriu

5.1 Fixní náklady

Pokud budeme modelovat případnou realizaci určité výroby, pak náklady na výrobu se sestávají z počáteční investice a dále z investice do každého výrobku. Pokud se výroba nerealizuje, jsou všechny náklady nula. Máme tedy kriterium se skokem v počátku.

Za předpokladu, že K je počáteční skok, bude mít kriterium tvar

$$J = \begin{cases} 0 & \text{pro } x = 0 \\ K + c'x & \text{pro } x \geq 0 \end{cases}.$$

Zápis takové úlohy je následující

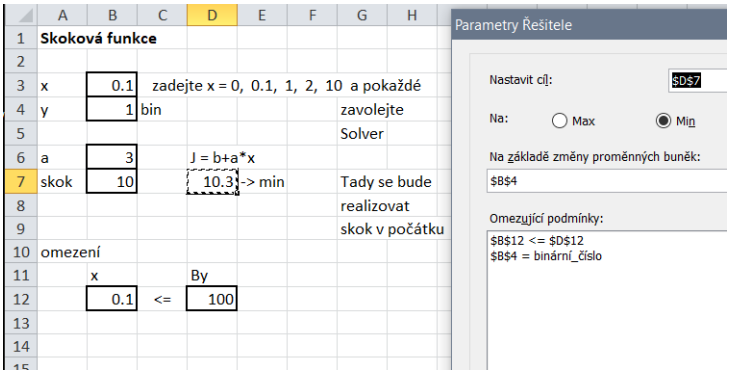
$$Ky + c'x \to \min^{3}$$

a podmínky

$$x \leq By,\ x \geq 0,\ y \in \{0, 1\}$$

S touto podmínkou jsme se již setkali u „indikace nenuly“, viz odstavec 4.2. Pokud je $x > 0$ musí být $y = 1$. Penalizace v kriteriu zaručí, že pro $x = 0$ bude také $y = 0$.

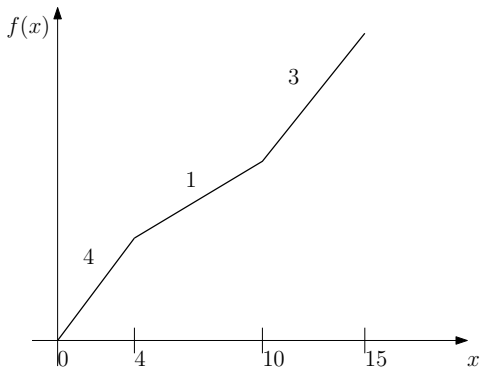
Excel: Pr11_skokFunkce.xlsx



5.2 Po částech lineární reprezentace

Dalším typem kriteriální funkce je funkce po částech lineární. Konstrukci takového kriteria ukážeme na příkladě. Mějme např. kriterium podle obrázku

[3] chceme minimální náklady



První úsečka je na intervalu $(0, 4)$ a má sklon 4, druhá na $(4, 10)$ se sklonem 1 a třetí na $(10, 15)$ se sklonem 3. Proměnnou x vyjádříme jako součet tří členů $x = \delta_1 + \delta_2 + \delta_3$ tak, že platí

$$\begin{aligned} 0 \leq \delta_1 \leq 4 \qquad & \text{délka 1. úseku} \\ 0 \leq \delta_2 \leq 6 \qquad & \text{délka 2. úseku} \\ 0 \leq \delta_3 \leq 5 \qquad & \text{délka 3. úseku} \end{aligned}$$

Funkci $f(x)$ vyjádříme jako součin směrnic přímek na jednotlivých intervalech a definovaných funkcí δ_i

$$f(x) = 4\delta_1 + \delta_2 + 3\delta_3.$$

Musíme ale zajistit, aby se δ_i "zapínaly postupně" a aby nižší δ_i držely svou konečnou hodnotu i za svým definičním intervalem. Tedy, aby při průchodu hodnot x od 0 do 15 bylo

	δ_1	δ_2	δ_3
$x \leq 4$	$x - 0$	0	0
$x \in (4, 10)$	4	$x - 4$	0
$x > 10$	4	6	$x - 10$

To zaručí následující podmínky s dvěma novými binárními proměnnými w_1 a w_2

$$\begin{aligned} 4w_1 \leq \quad & \delta_1 \quad \leq 4 \\ 6w_2 \leq \quad & \delta_2 \quad \leq 6w_1 \\ 0 \leq \quad & \delta_3 \quad \leq 5w_2 \end{aligned} \tag{5.1}$$

$$w_1, w_2 \in \{0, 1\}$$

Důkaz, že předpis platí je ukázán v tabulce 1.

Poslední varianta $w_1 = 0$ a $w_2 = 1$ je nepřípustná a podmínka $6w_2 \leq \delta_2 \leq 6w_1 \rightarrow 6 \leq 0$ ji vylučuje.

Postup konstrukce:

Kriterium je tvořeno třemi přímkami danými intervalem I_i a směrnicí s_i (viz zadání), $i = 1, 2, 3$.

Pro dané x chceme určit hodnotu kriteria $J = f(x)$.

Zavedeme stavové proměnné:

$w_1 = w_2 = 0$	$w_1 = 1$ a $w_2 = 0$	$w_1 = w_2 = 1$
$0 \leq \delta_1 \leq 4$	$\delta_1 = 4$	$\delta_1 = 4$
$\delta_2 = 0$	$0 \leq \delta_2 \leq 6$	$\delta_2 = 6$
$\delta_3 = 0$	$\delta_3 = 0$	$0 \leq \delta_3 \leq 5$

Tabulka 1: Po částech lineární funkce

- binární $w_i \in \{0, 1\}$, $i = 1, 2$ a
- reálnou $d_i \geq 0$, $i = 1, 2, 3$.

Ty budeme optimalizovat. (Proměnná d označuje δ.)

Pomocí w (5.1) sestavíme meze pro d a zadáme podmínky:

$$\text{dolní mez} \leq d \leq \text{horní mez}$$

$$w \text{ je binární}$$

$$x = \sum d_i$$

Potom $f(x) = \sum s_i \delta_i(x)$, kde s_i jsou směrnice přímek z kriteria (viz zadání).

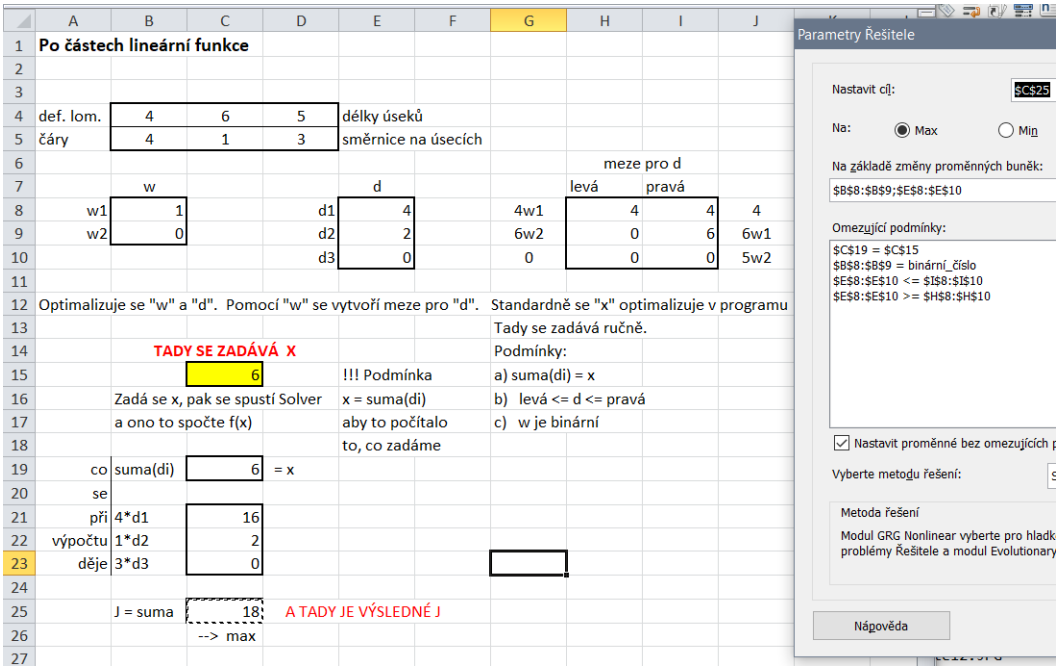
Poznámka

Stejnou konstrukci lze provést i pro více intervalů, pomocí podmínek

$$L_j w_j \leq \delta_j \leq L_j w_{j-1},$$

kde L_j je délka j-tého segmentu.

Excel: Pr12_lomCara1.xlsx (ručně zadané x a vypočtené J)



Příklad

Navrhujeme výrobu jednoho druhu výrobku. Náklady předpokládáme úměrné počtu výrobků $n = 0.8x$, kde x je počet vyrobených výrobků a zisk očekáváme po částech lineární funkcí $f(x)$

$$f(x) = \begin{cases} x & \text{pro } x \in (0, 10) \\ \frac{5}{4}x - \frac{5}{2} & \text{pro } x \in (10, 50) \\ \frac{1}{2}x + 35 & \text{pro } x \in (50, 100) \end{cases} \tag{5.2}$$

Kolik máme vyrábět, abychom dosáhli maximálního zisku?

Řešení

Jako stavové proměnné zavedeme

$$w = [w_1, w_2] \in \{0, 1\} \quad \text{a} \quad \delta = [\delta_1, \delta_2, \delta_3] \geq 0.$$

Pro lomenou čáru označíme směrnice $s = \left[1,\ \frac{5}{4},\ \frac{1}{2}\right]$ a úseky $u = [10,\ 40,\ 50]$.

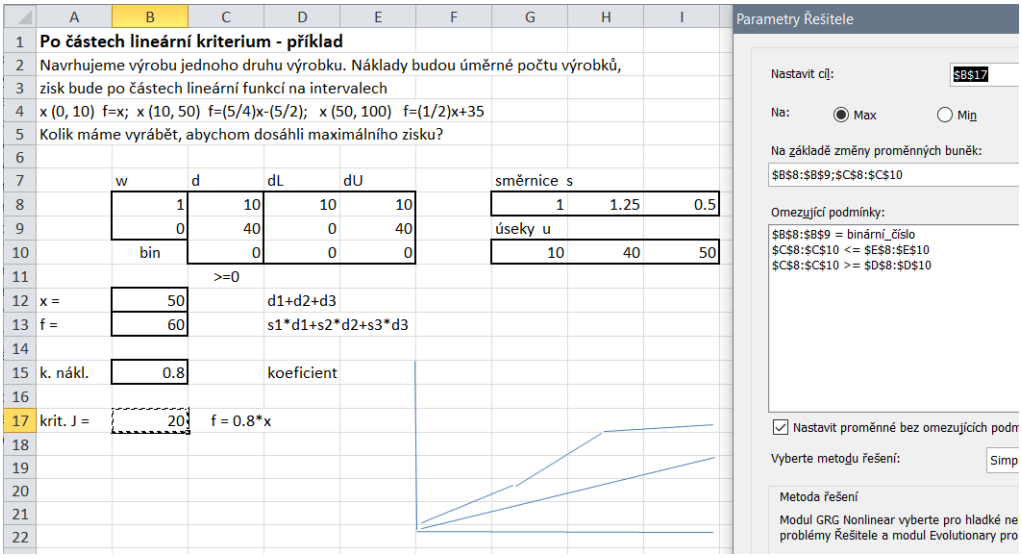
Dále sestavíme dolní a horní meze pro δ

$$\begin{bmatrix} u_1 w_1 \\ u_2 w_2 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 w_1 \\ 40 w_2 \\ 0 \end{bmatrix} \leq \delta \leq \begin{bmatrix} u_1 \\ u_2 w_1 \\ u_3 w_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 40 w_1 \\ 50 w_2 \end{bmatrix}$$

Kriterium bude dáno rozdílem „výnos“ - „náklady“

$$J = \underbrace{s_1\delta_1 + s_2\delta_2 + s_3\delta_3}_{\text{výnos = lomená čára}} \underbrace{- 0.8 \left(\underbrace{\delta + \delta_2 + \delta_3}_{x} \right)}_{\text{náklady}} \to \max$$

Excel: Pr12_lomCara2.xlsx



	A	B	C	D	E	F	G	H	I
1	**Po částech lineární kriterium - příklad**								
2	Navrhujeme výrobu jednoho druhu výrobku. Náklady budou úměrné počtu výrobků,								
3	zisk bude po částech lineární funkcí na intervalech								
4	x (0, 10) f=x; x (10, 50) f=(5/4)x-(5/2); x (50, 100) f=(1/2)x+35								
5	Kolik máme vyrábět, abychom dosáhli maximálního zisku?								
6									
7		w	d	dL	dU		směrnice s		
8		1	10	10	10		1	1.25	0.5
9		0	40	0	40		úseky u		
10		bin	0	0	0		10	40	50
11			>=0						
12	x =	50		d1+d2+d3					
13	f =	60		s1*d1+s2*d2+s3*d3					
14									
15	k. nákl.	0.8		koeficient					
16									
17	krit. J =	20	f = 0.8*x						
18									
19									
20									
21									
22									

5.3 Aproximace nelineárních funkcí

Nelineární funkci můžeme vhodně rozdělit na intervaly a na nich použít lineární aproximace. Počet takových intervalů je závislý na nelineární funkci a musí být zvolen vhodně tak, aby platilo, že v intervalu je funkce lineární. Pokud platí toto pravidlo, lze použít techniku z předchozího odstavce.

Příklad

Budeme řešit předchozí příklad s tím, že výnos (5.2) bude dán spojitou křivkou

$$f(x) = -0.008\,(x-100)^2 + 80$$

Tuto křivku budeme realizovat na intervalech délky 10 po částech lineárně. Náklady budou opět ležet na přímce $0.8x$.

Řešení

Přesné řešení dostaneme, když zisk zderivujeme a položíme rovný nule

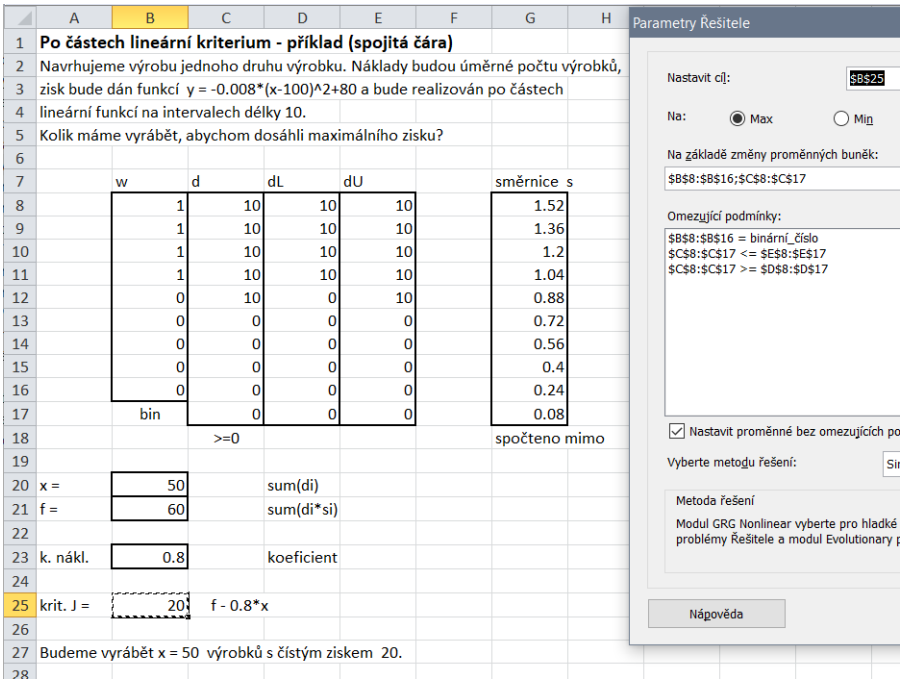
$$\frac{d}{dx}\left[-0.008\,(x-100)^2 + 80\right] - 0.8x = 0$$

$$-0.16\,(x-100) = 0.8$$

$$x - 100 = -50$$

$$x = 50$$

Řešení s aproximovaným ziskem je v Excelu Pr12_lomCara3.xlsx. Výsledek je stejný.



	A	B	C	D	E	F	G	H
1	**Po částech lineární kriterium - příklad (spojitá čára)**							
2	Navrhujeme výrobu jednoho druhu výrobku. Náklady budou úměrné počtu výrobků,							
3	zisk bude dán funkcí y = -0.008*(x-100)^2+80 a bude realizován po částech							
4	lineární funkcí na intervalech délky 10.							
5	Kolik máme vyrábět, abychom dosáhli maximálního zisku?							
6								
7		w	d	dL	dU		směrnice s	
8		1	10	10	10		1.52	
9		1	10	10	10		1.36	
10		1	10	10	10		1.2	
11		1	10	10	10		1.04	
12		0	10	0	10		0.88	
13		0	0	0	0		0.72	
14		0	0	0	0		0.56	
15		0	0	0	0		0.4	
16		0	0	0	0		0.24	
17		bin	0	0	0		0.08	
18			>=0				spočteno mimo	
19								
20	x =	50		sum(di)				
21	f =	60		sum(di*si)				
22								
23	k. nákl.	0.8		koeficient				
24								
25	krit. J =	20	f - 0.8*x					
26								
27	Budeme vyrábět x = 50 výrobků s čistým ziskem 20.							
28								

5.4 Součin v kritériu

Binární veličiny

Uvažujeme kriterium, ve kterém se vyskytuje výraz $x_1 x_2 \cdots x_k$ jako součin k binárních veličin. Linearizaci provedeme takto: Zavedeme binární veličinu $w \in \{0,1\}$, která se rovná součinu $w = \prod_i x_i$, a omezení

$$\begin{aligned} kw &\leq \sum x_i \\ w &\geq \sum x_i - (k-1) \end{aligned}$$

A skutečně. Je-li $\sum x_i$ rovno $0, 1, \cdots, k-1$ (tedy kdy je alespoň jeden člen $x_i = 0$) je $w = \prod_i x_i = 0$. Pro $\sum x_i = k$ (kdy všechny x_i jsou rovny jedné) je $w = \prod_i x_i = 1$. To je přesně to, co má dělat součin $x_1 x_2 \cdots x_k$.

Příklad

a) Pro $x = [0,0,1,0]$ je $k = 4$, a$\sum x_i = 1$. Podmínky jsou

$$4w \leq 1$$

$$w \geq 1 - (4-1) = -2$$

Tedy, $w \geq -2$ a $w \leq 1/4$. Z možných hodnot $\{0,1\}$ vyhovuje pouze $w = 0$.

b) Pro (třeba) $x = [1,0,1,1]$ je $k = 4$ a $\sum x_i = 3$. Podmínky

$$4w \leq 3$$

$$w \geq 3 - (4-1) = 0$$

Tedy, $w \geq 0$ a $w \leq 1/4$. Z možných hodnot $\{0,1\}$ vyhovuje opět pouze $w = 0$.

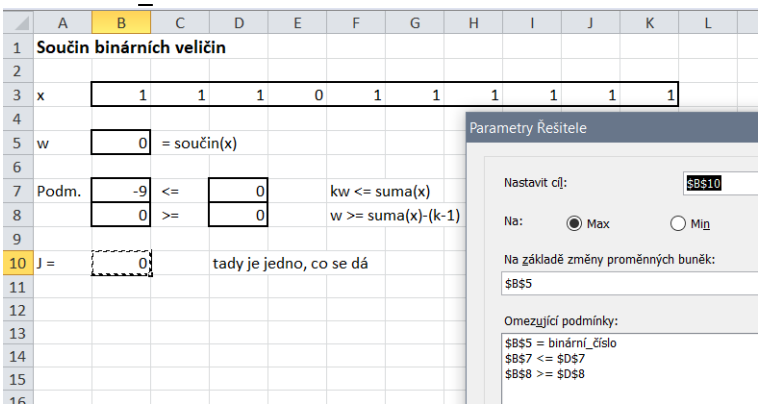
c) Pro $x = [1,1,1,1]$ je $k = 4$ a $\sum x_i = 4$. Podmínky

$$4w \leq 4$$

$$w \geq 4 - (4-1) = 1$$

Tedy, $w \geq 1$ a $w \leq 1$. Z možných hodnot $\{0,1\}$ vyhovuje pouze $w = 1$.

Excel: Pr13_soucin1Bin.xlsx



Binární veličiny a jedna spojitá

Tento případ ukážeme na příkladě tří binárních a jedné spojité nebo diskrétní veličiny.

Máme x_1, x_3, x_3 - binární veličiny, $y \in (0,\, u)$ spojitou (diskrétní) veličinu s maximální hodnotou u).

Zavedeme $w = x_1 \cdot x_2 \cdot x_3 \cdot y$, kde w je nyní spojitá veličina a dále podmínky

$$w \leq u x_j,\ j = 1, 2, 3$$
$$w \geq 0$$
$$w \leq y$$
$$w \geq u \left(\sum x_i - 3 \right) + y$$

Poznámka

1. *Obecně může být v součinu k binárních proměnných. Potom poslední podmínka bude*

$$w \geq u \left(\sum x_i - k \right) + y$$

2. *Pokud by některá binární proměnná byla umocněná, lze mocninu vynechat, protože platí $x^k = x$ pro $x \in \{0, 1\}$.*

Analýza úlohy

Dále budeme uvažovat součin binární veličiny $x \in \{0, 1\}$ a reálné veličiny $y \in (0, u)$. Kriterium bude $J = xy$. Opět zavedeme $w = xy$ a podmínky

$$w \leq ux$$
$$w \geq 0$$
$$w \leq y$$
$$w \geq u\,(x - 1) + y$$

Pro $x = 0$ bude

$$w \leq 0,\ \ w \geq 0,\ \ w \leq y,\ \ w \geq -u + y$$

kde první dvě podmínky definují $w = 0$ a druhé dvě jsou automatiky splněny.

Pro $x = 1$ bude

$$w \leq u,\ \ w \geq 0,\ \ w \leq y,\ \ w \geq y$$

kde třetí a čtvrtá podmínka dává $w = 1$ a první dvě jsou splněny.

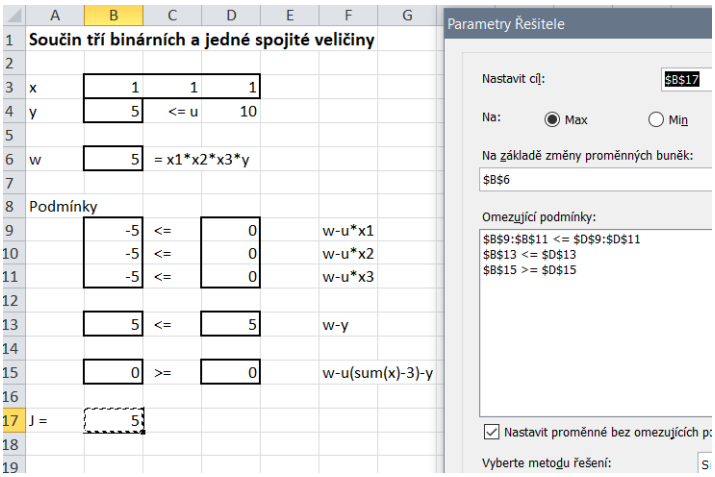
Nebo

Pro y binární bude

$$w \geq 0,\ w \leq x,\ w \leq y,\ w \geq x + y - 1$$

po prozkoumání zjistíme, že skutečně je $w = xy$.

Příklad programu je v Excelu Pr13_soucin2xy.xlsx



5.5 Modelování minimaxu v kriteriu

Máme k pracovních týmů pracujících na různých úkolech. Chceme navrhnout podmínky práce tak, aby práce všech týmů byla skončena co nejdříve, přičemž každý tým musí ukončit všechny své úkoly. Hledáme tedy minimum pro nejdelší dobu plnění úkolů - což je úloha minimaxu.

Modelujeme takto:

$p_1(x), p_2(x), \cdots, p_k(x)$ jsou doby trvání práce jednotlivých týmů (v závislosti na optimalizované veličině x).

Definujeme novou proměnnou q (trvání nejdelší práce) a zavedeme podmínky

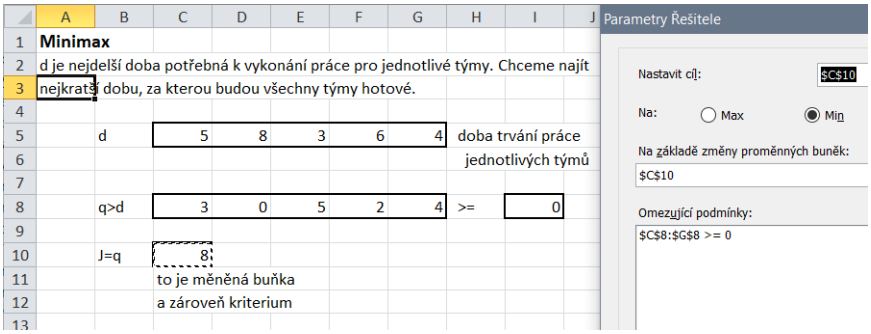
$$\begin{aligned} p_1 &\leq q \\ p_2 &\leq q \\ &\cdots \\ p_k &\leq q \end{aligned}$$

a jako kriterium vezmeme

$$J = q \to \min$$

Proměnnou q zadáme jako proměnnou modelu (tj. jako hledané řešení).

Excel: Pr14_minimax1.xlsx (ruční nastavení hodnot a výpočet součinu)



Příklad

Máme tři prcovní týmy po pěti lidech. $x_1 = [x_1, x_2 \cdots x_5]_1$, $x_2\,[x_1, x_2 \cdots x_5]_2$, $x_3\,[x_1, x_2 \cdots x_5]_3$, ze kterých vybíráme max 3-členné týmy na úklid tří objektů. Každý člověk má svůj výkon v (práce/hod), hodinový plat p (plat/hod) a časové omezení o (omezení na x). Týmy pracují paralelně. Jak máme týmy sestavit, aby celková práce byla ukončena co nejdříve?

Řešení

Zavedeme stavové vektory:

- $x_i = [x_1, x_2, x_3, x_4, x_5]_i \in R_0^+$, $i = 1, 2, 3$ - počet odpracovaných hodin osobou j z týmu i a
- $y_i = [y_1, y_2, y_3, y_4, y_5]_i \in \{0, 1\}$, $i = 1, 2, 3$ - indikátor účasti osoby j z týmu i na práci.

Zavedeme je jako matice s prvky x_{ij} a y_{ij}.

Pokud člověk j z týmu i nebude vybrán na práci, bude jeho $x_{ij} = 0$ a to bude indikováno pomocí $y_{ij} = 0$. Jinak bude jeho $x_{ij} > 0$ a $y_{ij} = 1$. Tato vazba se zajistí podmínkou

$$x_{ij} \leq M y_{ij}$$

kde třeba $M = 1000$.

Omezení jednotlivých lidí na čas bude dáno

$$x_{ij} \leq o_{ij}$$

kde o je matice omezení ze zadání.

Výběr lidí do týmů $i = 1, 2, 3$ se provede podmínkou

$$\sum_j y_{ij} \leq 3,\ i = 1, 2, 3$$

Vykonaná práce v týmech bude

$$\sum_j x_{ij} pr_{ij} \geq pož_i$$

kde pr_i práce člověka ij a $pož_i$ je práce požadovaná od týmu i. Vykonaná práce musí být alespoň rovna požadované.

Čas, který tým i potřebuje na splnění úkolu je d_i

$$d_i = \sum_j x_{ij},\ i = 1, 2, 3.$$

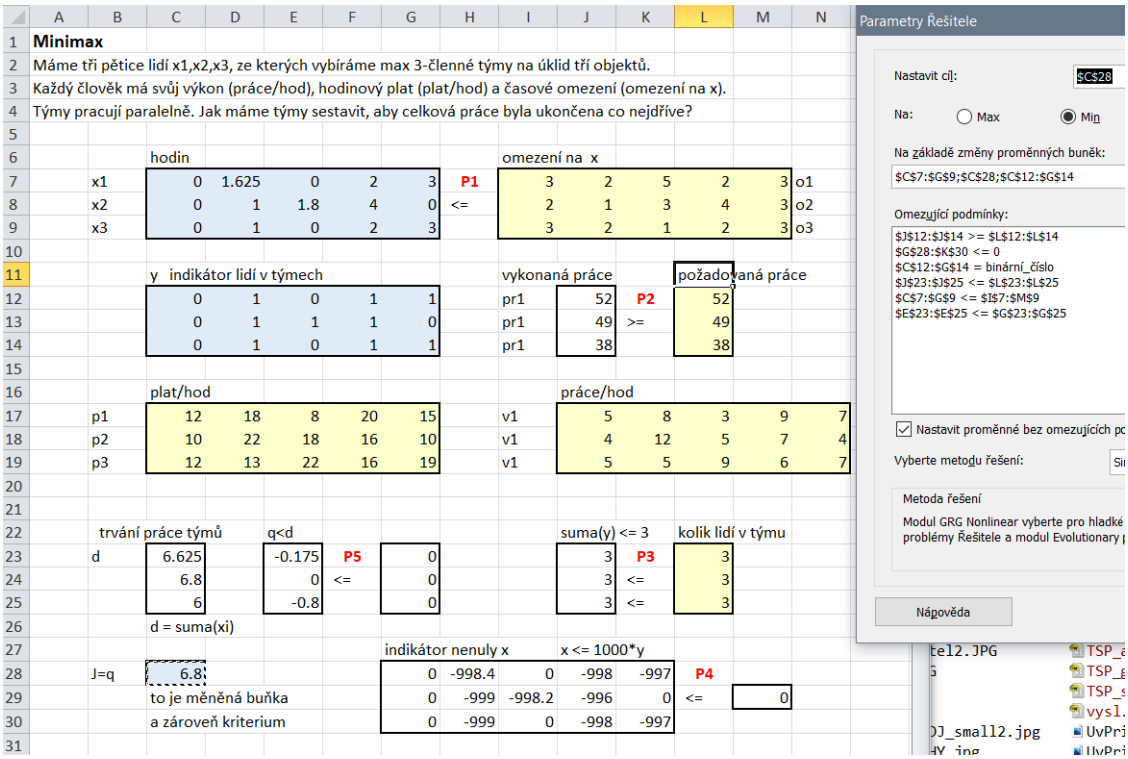
Celková doba J se bude rovnat minimální hodnotě q pro kterou platí

$$q \geq d_i,\ i = 1, 2, 3$$

a tedy kriterium bude

$$J = q \to \min$$

Excel: Pr14_minimax2.xlsx.



Modely celočíselného programování

6 Heuristiky

Celočíselné programování má jako základní metodu řešení metodu **větví a mezí**. Ta je založena na hledání LP řešení a postupném přidávání podmínek celočíselnosti ve formě přidaných omezení. Jedná se o úplný průzkum kde se vyznačují přípustná řešení a nakonec se vybere to nejlepší. To je ale dlouhé, proto se hledají rychlejší řešení, tzv. **heuristiky**.

Tady hraje významnou roli tzv. LP relaxation.

LP relaxation

Je podpůrná procedura pro řadu dalších metod. Její podstatou je to, že IP úlohu, kde $x \in \{0, 1\}$, nahradíme LP úlohou, kde $x \in (0, 1)$.

Poznámka: *Asi to jde použít i obecně pro IP s $x \in N$ tak, že se uvažuje jen LP úloha s $x \in (0, N)$.*

6.1 Heuristiky šité na míru

Ukážeme na příkladě: Tři pracovníci mají být umístěni na tři pracoviště. Na každé místo má přijít jeden pracovník a všechna místa musí být obsazena. Náklady na jednotlivá umístění jsou v tabulce

Prac.\Místo	1	2	3
A	1	3	4
B	3	5	7
C	3	2	4

Požadujeme nejnižší náklady.

1) Heuristický postup spočívá v tom, že náhodně vybereme pracovníka a přiřadíme ho na náhodně vybrané místo. To opakujeme, přičemž respektujeme již provedená přiřazení.

2) Lepší postup (tzv. greedy) je, když vybereme nejnižší cenu a provedeme odpovídající přiřazení. Pak vezmeme další nejnižší cenu a pokud je to možné, přiřadíme. Tak dostaneme A$\rightarrow$1 (1), C $\rightarrow$2 (2), B $\rightarrow$ 3 (7). Celkem 10. Tím, že jsme na začátku vybírali co nejnižší, zbylo nám nakonec velké.

Někdy bývá zvykem s řešením trochu zatřepat (juggle). To znamená některá řešení zaměnit a sledovat, co to dělá s kriteriem.

Např.

A $\leftrightarrow$ C: C $\rightarrow$ 1 (3), A $\rightarrow$ 3 (4), B $\rightarrow$ 2 (5). Celkem 12 - horší

A $\leftrightarrow$ B: B $\rightarrow$ 1 (3), C $\rightarrow$ 3 (4), A $\rightarrow$ 2 (3). Celkem 10 - stejné.

6.2 Greedy heuristics

Jedná se o postup řešení, který funguje podle lokálního kriteria, které bere v úvahu jen pohled dopředu a provedené kroky již dále neanalyzuje.

Například: Úlohu o obchodním cestujícím řeší tak, že začne v libovolném městě a postupně přidává další nejbližší města, pokud jsou volná pro přidání.

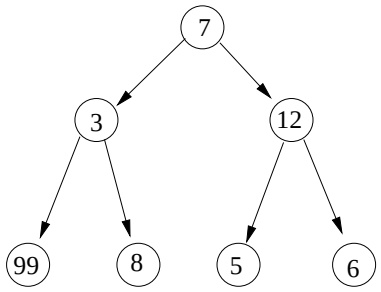
Greedy algoritmy mají následující části (*ilustrujeme na obchodním cestujícím*)

- množinu, ze které se vybírají kandidáti, které mají být přidáni k současnému řešení (*množina měst*),
- výběrovou funkci podle které se v každém kroku vybírá kandidát (*nejbližší město*),
- pravidlo, které kontroluje, zda je zvolený kandidát volný pro přidání (*město není dosud napojeno*),
- kriteriální funkci, která přiřazuje "cenu" částečnému nebo konečnému řešení (*délka cesty*),
- zastavovací pravidlo, které indikuje konec řešení úlohy (*všechna města jsou napojena*).

Poznámky

Algoritmus dělá jeden greedy krok za druhým a nikdy se nesnaží měnit to, co již udělal.

Nebezpečí při lokálním rozhodování je ilustrováno na následujícím obrázku. Zde hledáme cestu s největším součtem ohodnocení.



Nejdříve se rozhodneme pro dvanáctku a tím se mineme s 99.

Příklady

1. Kruskalův algoritmus (**minimální kostra**)

   Opakuj následující kroky, dokud je to možné:

   - Z hran grafu G, které dosud nebyly vybrány, vyber nejkratší hranu, která nevytváří žádnou kružnici s hranami již vybranými.

   Množina vybraných hran je kostrou grafu G, která je navíc minimální

   nebo

   Opakuj následující kroky, dokud je to možné:

   - Z hran G, které dosud nebyly vybrány, vyber nejdelší, která je nerozpojí.

   Množina nevybraných hran je minimální kostrou grafu G.

2. Huffmanův kód

   Máme písmenka a ty chceme zakódovat tak, aby kód byl co možná nejkratší. Postupujeme takto

   Písmenkům přiřadíme váhy (pravděpodobnosti) a seřadíme je do fronty od nejmenší váhy.

   Vezmeme dvě písmenka ze začátku fronty (od nejmenší váhy) a spojíme je v jeden prvek s váhou, která je součtem vah těch prvků. Spojený prvek nějak pojmenujeme a zařadíme do fronty podle jeho váhy. Původní dva prvky z fronty odstraníme.

   Tak se vytvoří graf jehož listy končí písmenky. Každému písmenku se přiřadí kód z nul a jedniček který dostaneme tak, ze začneme nahoře v koření stromu a postupujeme směrem k písmenku. Při tom jdeme-li doleva zapíšeme 0 a doprava 1.

   Příklad

písmeno	a	b	c	d
váha×100	35	10	21	34

   Seřazeno

písmeno	b	c	d	a
váha×100	10	21	34	35

   krok 1: $x1 = (b, c)$, váha 31,

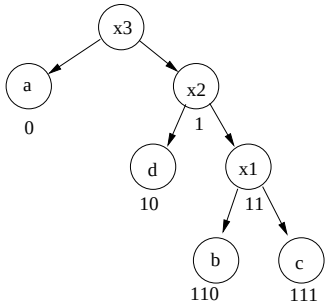
písmeno	$x1$	d	a
váha×100	31	34	35

   krok 2: $x2 = (x1, d)$, váha 65,

písmeno	a	$x2$
váha×100	35	65

   krok 3: $x3 = (a, x2)$, váha 100

   Graf



Třeba slovo "daca" bude 1001110. Přitom není třeba vyznačovat konce písmen, protože při dekódování se nemůžeme splést: 1 neexistuje, 10 je d, 0 je a, 1 - nic, 11 - nic, 111 je c, 0 je a.

6.3 Zaokrouhlovací heuristiky

Uvažujeme omezení $a_{i1}x_1 + \cdots + a_{i,j}x_j + \cdots a_{jn}x_n \leq b_j$, kde $a_{i,j} > 0$. Potom zaokrouhlením x_j nahoru můžeme porušit omezení. Zaokrouhlení dolů nevadí. Obdobně je to i pro $a_{i,j} < 0$. Toho můžeme využít v zaokrouhlovacích technikách.

Zavedeme značení

"horní zámek" je kladný koeficient v daném řádku matice A.

"dolní zámek" je záporný koeficient v daném řádku matice A.

Λ_j^U je počet horních zámků v j-tém sloupci matice A.

Λ_j^L je počet dolních zámků v j-tém sloupci matice A.

Význam: Jestliže je $\Lambda_j^U = 0$, pak j-tý sloupec matice A neobsahuje kladné prvky. Pro proměnnou x_j je možno psát

$$\begin{bmatrix} a_{1,j} \\ a_{2,j} \\ \cdots \\ a_{n,j} \end{bmatrix} x_j \leq \begin{bmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{bmatrix}.$$

Protože jsou ale všechny koeficienty a záporné, můžeme proměnnou x_j zaokrouhlit nahoru, aniž bychom porušili omezení (proměnná x_j není zhora zamčená). Podobně je to pro Λ_j^L.

Jednoduché zaokrouhlení

Všechny proměnné s $\Lambda^U = 0$ zaokrouhlíme nahoru a ty s $\Lambda^L = 0$ zaokrouhlíme dolů.

Zaokrouhlení

Startuje z LP relaxation a bere v úvahu Λ^U a Λ^L.

Aplikujeme zaokrouhlení

$$\hat{x}_j = \begin{cases} floor\,(x_j) & \text{pro } \Lambda_j^L \leq \Lambda_j^U \\ ceil\,(x_j) & \text{jinak} \end{cases}$$

Jestliže aktuální řešení neporušuje omezení, pokračujeme stejně i dále.

Jestliže jsou omezení porušena, vybereme jedno, které je porušeno a snažíme se toto porušení zmenšit tím, že vezmeme jednu celočíselnou proměnnou se zlomkovou hodnotou a snažíme se ji zaokrouhlit ve prospěch porušeného omezení. Vybíráme proměnnou s nejmenším počtem zámků (zjevně, abychom toho co nejméně pokazili kolem).

Příklad

Řešíme IP program pro

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & -5 & 4 \\ -2 & 4 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 8 \\ 15 \\ 12 \end{bmatrix}, \quad c = [2, \ -1, \ 3]$$

x celé, nezáporné.

LP řešení je

$$x = \left[\frac{70}{9}, \frac{1}{9}, 0\right]$$

Úvahy o zaokrouhlování nás vedou k řešení $x = [7, 0, 0]$ což také je celočíselné řešení.

7 Metaheuristiky

Heuristiky hledají lokální extrém. Nás ale většinou zajímají globální extrémy. Najít globální extrém znamená opustit nalezený extrém a hledat dále - každý dále nalezený "lepší extrém" je potom kandidátem na extrém globální. Metodami, jak opustit lokální extrém a hledat "globálně" se zabývají metaheuristiky.

Poznámka

Předpona "meta" znamená něco "za" - např. metafyzika je něco, co je ještě za realitou, kterou popisuje fyzika. Podobně metaheuristika je něco, co je ještě za heuristickým hledáním extrémů. Je to hledání globálního extrému.

7.1 Iterativní lokální hledání

Při této metodě postupujeme následujícím způsobem:

1. Generujeme náhodné řešení.
2. Určíme okolí řešení.
3. Náhodně bereme řešení z okolí a buď ihned nebo po několika krocích hledáme, zda existuje lepší řešení, než je naše současné.
4. Pokud jsme nalezli nové lepší řešení, nahradíme jím existující a jdeme na bod 2., pokud ne, pokračujeme dále:
5. Zapamatujeme existující nejlepší řešení a jeho polohu a pokud není vyčerpán předepsaný počet iteraci, jdeme na bod 1., pokud je vyčerpán, pokračujeme.
6. Jako globální extrém vezmeme aktuální nejlepší řešení.

Algoritmus lze stručně shrnout takto: Opakovaně hledáme lokální extrém a doufáme, že mezi nalezenými bude i extrém globální.

Poznámka

Samozřejmě velmi záleží na velikosti skoků náhodného hledání, a to jak při lokálním prohledávání, tak i při nových iteracích algoritmu. Problém ale není tak neprůhledný, jak by se mohlo zdát. Máme k dispozici data, v nichž extrém hledáme. Z nich můžeme určit hranic datové oblasti a hledat v rámci této hranice.

Metodu demonstruje program

```
// Heuristiky šité na míru: skripta str. 93
// Osobám 1:n jsou přidělovány úkoly 1:n. Cena za vykonání
//  úkolu j osobou i je v tabulce m(i,j). Jak přiřadit úkoly,
//  aby cena byla minimální?
// Tj. nalézt posloupnost n prvků matice m tak, aby se
//  neopakovaly řádky a sloupce a součet byl minimální.
// ----------------------------------------
exec SCIHOME/ScIntro.sce
mode(0), //getd('.');

n=10;                          // počet lidí a úkolů
m=ceil(10*rand(n,n,'u'));      // matice lidi krát úkoly

Cr=%inf;
for i=1:1000                    // iterace
  ji=perm1(1:n);               // lidi
  jo=perm1(1:n);               // úkoly
  cr=0;
  for k=1:n
    cr=cr+m(ji(k),jo(k));      // aktuální cena
  end
  if cr<Cr     // akt. cena menší než dosavadní ?
    Cr=cr;     // zapamatuj
    Ji=ji;
    Jo=jo;
  end
  ct(i)=Cr;    // historie cen
end
Cr,Ji,Jo

set(scf(),'position',[500 100 800 400])
subplot(121),
plot(ct,'.:')
title 'Vývoj optimální ceny'
subplot(122),
plot(Ji,Jo,'.','markersize',8)
title 'Opt. prvky matice m'

// Poznámky:
// - udělat jako genetické algoritmy,
```

```
// - jiná interpretace: obchodní cestující
```

a program

```
// Hledání globálního maxima funkce 2 proměnných
// - funkce: vážený součet tří nenormovaných Gaussovek
// ------------------------------------------------------------
exec SCIHOME/ScIntro.sce, mode(0)

function [f,m,w]=fc(x)
  // generátor součtu tří skoro-gaussovek
  x=x(:);
  m=[-20 0; 0 0; 10 20]';     // středy
  w=[2.5 2.8 3];              // váhy
  sg2=100;                    // šířka
  f=0;
  for i=1:3
    f=f+w(i)*exp(-(x-m(:,i))'*(x-m(:,i))/sg2);
  end
endfunction

z=50;                         // rozsah zobrazovaných dat
xx=-(z-1):(z-0);
yy=-(z-1):(z-0);
i=0;
for x=xx                      // generování dat
  i=i+1;
  j=0;
  for y=yy
    j=j+1;
    Cf(i,j)=fc([x y]);        // matice generovaných dat
  end                         //  jen pro zobrazení
end

set(scf(1),'position',[250 250 500 400])
//contour2d(xx,yy,Cf,8);
mesh(xx,yy,Cf);
title 'Prohledávaná oblast'

// PROHLEDÁVÁNÍ

t0=[-10,-10]';                // start prvního hledání
[C0,m]=fc(t0);                    // kvalita prvního řešení
st=[]; tt=[]; CC=[];

for k=1:10                    // cyklus pro velké náhodné iterace
  for j=1:100                 // cyklus pro hledání lok. extrému
    for i=1:100               // hledání lepšího řešení v okolí
      s=t0+.2*rand(2,1,'n');     // nový bod
      C=fc(s);                   // kvalita nového bodu
      if C>C0, break, end        // je-li lepší, konec
```

```
    end

    t0=s;                       // nový začátek v rámci hledání
    C0=fc(t0);                     // lokálního extrému

    st=[st s];                  // zapamatuje úspěšné body na cestě
  end
  tt=[tt t0];                   // zapamatuje konce hledání lok. ex.
  CC=[CC C0];                      // pro jednotlivé iterace

  t0=30*rand(2,1,'n');          // generuje nový začátek (nová iterace)
  C0=fc(t0);                    // kvalita

end

disp 'Výsledné řešení je'
[xxx,mC]=max(CC);               // vybere nejlepší hodnotu kriteria
tMax=tt(:,mC)                   // a podle toho i řešení

set(scf(2),'position',[800 50 600 500])
plot(st(1,:),st(2,:),'.')
plot(tMax(1),tMax(2),'ro','markersize',12)
title 'Hledání a konečné řešení (kroužek)'
plot(m(1,:),m(2,:),'g.')
```

7.2 Prohledávání s tabu seznamem

Pokud se naše hledání optima odehrává v diskrétním prostoru řešení, můžeme použít techniku prohledávání s tabu seznamem. Hledání probíhá v sériích (určitý počet kroků), které se opakují, dokud není konec. V každé sérii se náhodně vybere nové řešení v okolí starého. V tabu seznamu se nalezená řešení uchovávají a algoritmus se jim v této sérii vyhýbá. Po ukončení každé série se vybere nejlepší dosažené řešení a z něho startuje další série. Startuje se z náhodně vybraného řešení.

Poznámka

Pamatovat si, která řešení jsme již prozkoumali a v dalším hledání se jim vyhnout, je jistě dobrá myšlenka. Je ale třeba zajistit, aby kontrola tabu seznamu nebyla mnohem delší, než opakované ověření již ověřeného řešení. Pokud je například kontrola tabu seznamu 5 krát delší, než ověření jednoho řešení, pak by asi bylo lépe prozkoumat 5 krát více řešení v daném okolí, než používat tabu seznam.

Metodu budeme ilustrovat na příkladě:

Řešíme úlohu obchodního cestujícího, který má projít všemi uzly hranově ohodnoceného grafu, vrátit se do stejného uzlu a ujít při tom co nejmenší vzdálenost (součet ohodnocení hran na cestě).

Postupujeme takto:

1. Generujeme náhodně první řešení
2. Z tohoto řešení odvodíme k sousedů tak, že mezi sebou vyměníme dva uzly. Toto generování je s tabu-listem. To znamená, že vždy jeden ze dvou generovaných uzlů vynecháme ze základní množiny uzlů, ze které se vybírá.
3. Pro generovaná řešení spočteme hodnoty kriteria a vybereme řešení s minimální hodnotou.
4. Dále pokračujeme v iteracích
5. Jako výchozí řešení vezmeme výsledné z minulého kroku a zároveň ho dáme do seznamu řešení v této fázi.
6. Pokračujeme body 2., 3. a 5., a to buď po určitý počet kroků nebo podle nějakého zastavovacího pravidla (např., že po určitý počet kroků je výsledné řešení stejné).

Program

```
// Obchodní cestující jako tabu search
// - města jsou 1, 2, ..., n
// - základní tabu search je k=5 kroků:
//    k-krát se zamění dvě města
// - maximální počet iterací je N=10000
// - končí se, když je tEx=100 kroků nic nezlepšuje
// -----------------------------------------------
exec SCIHOME/ScIntro.sce
mode(0)

// DEFINUCE FUNKCÍ
function J=kr(x,I)
  // kriterium pro obch. cestujícího
  // - délka cesty s uzly x=[x1,x2..,xn,x1]
  // x    uzly cesty
  // I    matice délek hran

  if max(size(I))<length(x)
    printf('\nError in kr: too small criterion\n\n');
  end

  n=length(x);
  x=[x(:)' x(1)];        // návrat do výchozího města
  J=0;
  for i=1:n
    J=J+I(x(i),x(i+1));  // součet délek hran
  end
endfunction

function ii=gen2i(n,k)
  // tabu generátor dvou indexů
  // generuj 2 indexy z 1,2,..n
  // a to opakuj k-krát s tabu listem
  // (tj. žádná dvojice se nesmí opakovat)
  z=1:n;
```

```
  for i=1:k
    j=ceil(n*rand(1,1,'u'));
    ii(1,i)=z(j);
    z(j)=[];
    jj=j;
    n=n-1;
    j1=ceil(n*rand(1,1,'u'));
    if j1==0, printf('\nError in gen2i(): too big k\n\n'), end
    ii(2,i)=z(j1);
  end
endfunction

function y=perm1(x)
  // y = jedns náhodná permutace prvků x
  y=[];
  n=length(x);
  for i=1:n
    nx=length(x);
    j=ceil(nx*rand(1,1,'u'));
    y=[y x(j)];
    x(j)=[];
  end
endfunction

// HLAVNÍ PROGRAM
n=10;               // počet uzlů grafu (měst)
k=5;                // počet záměn uzlů v jedne iteraci
N=10000;            // maximum iterací
tEx=100;            // počet kroků beze změny pro ukončení

if 0 // 1 - generuje se nová síť (nové vzdálenosti)
  // Generátot kriteriální funkce
  I=ceil(20*rand(n,n,'u'));
  save I.dat I
else
  // Volání kr. fce z minula
  load I.dat I
end

Mi=%inf;            // inicializace minima kriteria
x=(1:n)';           // očíslování měst: 1,2,...,n
xm=x;
for it=1:N          // cyklus iterací
  ii=gen2i(n,k);    // generuje k různých dvojic (bez opakování)
  for i=1:k         // cyklus pro záměny měst
    x(ii(:,i))=x([ii(2,i),ii(1,i)]);    // záměna měst
    J(i,1)=kr(x,I); // kriterium pro novou cestu
    X(:,i)=x;       // pamatuj
  end
```

```
  X=[X xm];          // přidej minimální cestu z minula
  [Mi,im]=min(J);    // najdi nové minimum a jeho pořadí
  xm=X(:,im);        // najdi aktuální minimální cestu
  Xt(:,it)=x;        // zapamatuj aktuální minimální cestu
  Mt(1,it)=Mi;       // zapamatuj aktuální minimum
  if it>tEx          // test konce iterací
    if mean(Mt(it-tEx:it))==Mt(it)
      break
    end
  end                // konce cyklu pro záměny
end                  // konec cyklu pro iterace

// Výsledky
x_opt=x',Krit=Mi,Iter=it
```

7.3 Simulated annealing

Jedná se o metodu hledání globálního extrému u funkce s více lokálními extrémy. Metoda využívá metodu Monte Carlo a prohledává celý prostor řešení s postupně se zmenšující pravděpodobností přijetí nového řešení.

Algoritmus metody je následující

Zvolte:

x_0 počáteční řešení

$T = T_0$ počáteční teplota (velikost náhodných skoků)

$k = 0$

pro $k = 1, 2, \cdots, K$

1. generuj x_k ze základní množiny
2. urči rozdíl $\Delta_k = f(x_{k-1}) - f(x_k)$
3. vypočti pravděpodobnost přijetí nového x_k

$$p = \begin{cases} 1 & \text{pro } \Delta_k \leq 0 \\ \exp\{-\Delta_k / T\} & \text{pro } \Delta_k < 0 \end{cases}$$

4. s pravděpodobností p přijmi nové řešení (jinak $x_k = x_{k-1}$)
5. přepočti teplotu $T = h(T_0, k)$ (tak, aby $T \to 0$ pro $k \to \infty$)

Hodnoty x_k by měly konvergovat ke globálnímu extrému.

Poznámky

1. Funkci $h(\cdot)$ můžeme volit např. jako "zapomínání": $T = T_0 \varphi^k$, kde $\varphi < 1$ je blízko k jedné - např. $\varphi = 0.9$.
2. Body 1. až 4. se mohou několikrát opakovat se stejnou teplotou T.

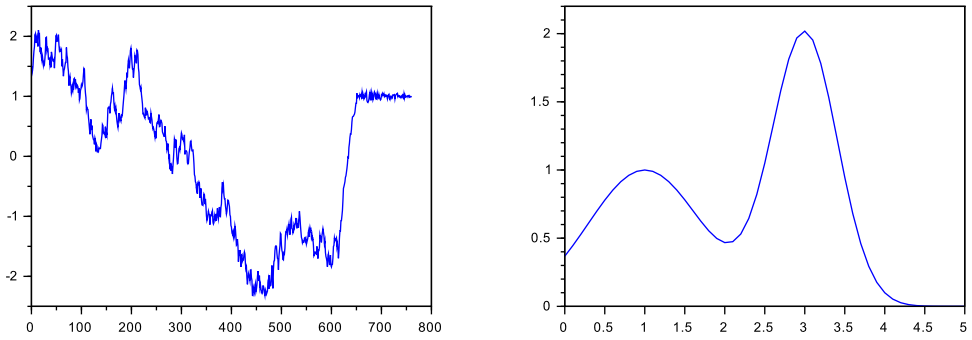
3. Parametr T je nazýván teplota. To souvisí s počáteční motivací k metodě, která byla inspirována žíháním kovů. Čím větší je T, tím větší odskoky od současného řešení se připouští (tím větší je okolí, ze kterého se generují nové hodnoty řešení). Hodnota T se během řešení zmenšuje, čímž se řešení stabilizuje.

Ukázkový program je následující:

```
// Simulated annealing
// - hledání globálního extrému funkce
// - funkce jedné proměnné
// ---------------------------------
exec('SCIHOME/ScIntro.sce',-1)
mode(2)

// kriterium
function y=fc(x)
  y=exp(-(x-1)**2)+2*exp(-3*(x-3)**2)
endfunction

// přepočet T
function T=nT(T,fi)
  T=fi*T;
endfunction

// pravděpodobnost pro přijetí nového řešení
function p=Pr(om,T)
  if om<0, p=1;
  else
    p=exp(-om/T);
  end
endfunction

if 1           // 1 - kreslení maximalizované funkce
  h=0:.1:5;      // rozsah x pro funkci
  y=[];          // kreslení
  for i=h
    y=[y fc(i)];
  end
  set(scf(1),'position',[800 0 400 300])
  plot(h,y)
  title('funkce')
end

// Zadání příkladu ==============================
T=100;          // teplota
fi=.95;         // zapomínání teploty

x=1.3;         // startovací bod
xt=x;          // pamatuj
tt=1;          // čas přijetí
fx=fc(x);      // hodnota startovacího bodu
```

```
pt=[];

nIter=log(1e-15)/log(fi); // automaticky počet iterací
                          // nebo fi=exp(log(1e-5)/nI)
for i=1:fix(nIter) // CYKLUS PRO SNIŽOVÁNÍ TEPLOTY

  x=5*rand(1,1,'u');    // nové řešení
  fxn=fc(x);            // nové kriterium
  om=fx-fxn;            // staré - nové kriterium
  p=Pr(om,T);           // pr. pro přijetí
  if p>.2      // přijetí
    xt=[xt x];
    tt=[tt i];
    fx=fxn;
  end
  T=nT(T,fi);  // přepočet teploty
  pt(i)=p;
  xx(i)=x;
end
set(scf(2),'position',[400 0 400 300])
plot(tt,xt,'.:')
title('vývoj přijatých řešení')

set(scf(4),'position',[400 400 400 300])
plot(xx)
title('včechna generovaná x')

set(scf(3),'position',[800 400 400 300])
plot(pt,'.:')
title('pr. přijetí')

if 0                          // vykreslení funkce
  y=[];
  xx=-50:.1:50;
  for x=xx
    y=[y fc(x)];
  end
  scf;
  plot(xx,y)
end
```

Výsledek je



kde vlevo je vidět postupné hledání globálního maxima a vpravo je funkce, jejíž globální maximum hledáme.

7.4 Mravenčí kolonie

Mravenčí optimalizace je metaheuristika inspirovaná chováním mravenčích kolonií. Má také blízko ke genetickým algoritmům. Je založena na heuristickém pohledávání a vyznačování úspěšných strategií. Nejdříve popíšeme princip mravenčího hledání, potom sestavíme algoritmus pro úlohu obchodního cestujícího.

Mravenci mají své hnízdo a odtud se vydávají hledat potravu. Jednotliví mravenci se náhodně potulují poblíž hnízda a pokud najdou potravu, vrací se stejnou cestou zpět k hnízdu a cestou vylučují feromon. Ostatní mravenci při svém putování dávají přednost cestě s feromonem. Tím ještě feromonové značení zesilují. Tak opouštějí prázdné cesty bez potravy a tvoří kolonie, putující pro potravu.

Vidíme zde dva základní prvky:

1. preference označené cesty,
2. značení závislé na úspěchu cesty.

Feromonová stopa postupně vyprchává a tím se udržuje aktuálnost značení. Označené zůstávají jen ty cesty, na které je feromon stále ukládán.

Abychom mohli být konkrétní, budeme uvažovat úlohu obchodního cestujícího: Máme n měst navzájem propojených různě dlouhými obousměrnými cestami s délkami d_i, $i = 1, 2, \cdots, n$. Obchodní cestující má projít všemi městy, každým právě jednou, a vrátit se do města ze kterého vyšel. Chce najít takovou cestu, která bude nejkratší. (Lze řešit pomocí IP, ale zadání úlohy je nepříjemné - musí se vyloučit všechny pod-cesty s délkami menšími než je n.)

Heuristické řešení je následující.

Úlohu řešíme v iteračních krocích $1, 2, \cdots$. V každém kroku vypustíme m mravenců z náhodně vybraných měst. Tito mravenci postupují mezi městy tak, aby se nikdy nevrátili do měst, která už navštívili (udržují si tabu list, kde mají zaznamenány už navštívená města a jako další město vybírají jen to, co není na tabu listu). Při výběru dalšího města se řídí podle matice pravděpodobností $P = [p_{i,j}]$, $i, j = 1, 2, \cdots, n$, kde $p_{i,j}$ je pravděpodobnost přechodu z města i do města j.[4] Když projdou n městy (tj. vytvoří nějakou cestu pro obchodního cestujícího)

[4] O tvorbě této matice bude zmínka dále.

spočtou délku této cesty a podle ní vytvoří feromonovou stopu (která se přidá na jejich cestě k již existující stopě). Feromon $F_{i,j}$ podle cesty k-tého mravence se přepočte podle vzorce

$$F_{i,j} = F_{i,j} + \frac{Q}{L_k}$$

pro všechny i, k, které leží na této cestě; L_k je délka cesty, Q je konstanta. A to se provede pro každého mravence $k = 1, 2, \cdots, m$.

Po dokončení každého kroku iterace se znovu konstruuje matice pravděpodobností P podle aktuálního feromonu - v nejjednodušší formě

$$P_{i,j} \propto F_{i,j}$$

nebo

$$P_{i,j} \propto F_{i,j}/d_{i,j}$$

tedy buď je přímo úměrná feromonu, nebo se vezme v úvahu ještě délka $d_{i,j}$ cesty i, j (preferují se kratší úseky cesty).

Z matice P se generují pravděpodobnosti přechodu mravence z aktuálního města i do dalšího města j tak, že se z matice P vyberou prvky $\{P_{i,j}\}$ kde $j \in J$ a J označuje množinu všech přípustných měst, tj. měst, do kterých mravenec smí pokračovat (co nejsou na tabu listu). Tyto prvky se pak normují tak, aby jejich součet byl roven jedné: $p_i = \left\{ \frac{P(i,j)}{\sum_k P(i,j)} \right\}$ pro i pevné a $j \in J$.

Tyto iterační kroky se opakují - buď určitý počet nebo dokud nedojde k ustálení směru pohybu mravenců. Každopádně, v každém kroku se určí nejkratší nalezená cesta a pokud je menší než aktuální minimum, tak se zapamatuje. Výsledkem pak není nejkratší cesta z posledního kroku, ale nejkratší cesta dosažená během iterací (mravenci často zamrznou na neoptimální cestě i když na optimální již byly - pozn. překladatele).

Start algoritmu je následující:

F je matice malých kladných náhodných čísel (tak, aby je ovlivnil feromonový přepočet, ale ne moc, aby se mravenci stačili rozběhnout kolem mraveniště a nebyli příliš rychle vázáni na nový feromon.

Náhodně se generuje m míst. Každému místu k se přiřadí tabu list ve kterém je jako první vyznačeno místo k. Následující místo se vybere podle pravděpodobností p_i, a to buď místo s maximální pravděpodobností (greedy heuristika) nebo se generuje z rozdělení s pravděpodobnostní funkcí p (pravděpodobnostní heuristika).

Dále algoritmus běží v iteračních krocích. Výsledkem je nejkratší dosažená cesta v průběhu celého běhu algoritmu.

Dále uvádíme program ve Scilabu

```
// THIS IS, HOW IT HOPEFULLY SHOULD RUN.
// opt. is 1 3 4 2 5 1  S=78, from Excel
// In each step of iteration Ni ants is run from various places.
// After completing their rout (after n steps) the minimal length
// and the corresponding sequence of places is recorded and according
// it the pheromone is updated. It runs for N steps (iterations).
// Initial pheromone is random (to give the ants a possibility to fork).
```

```
// The update of pheromone is given by 1/L where L is the minimal
// length of path from the current step. Forgetting is used, too.
// The way of ants is ruled by probabilities constructed from pheromone.
// Choice of path continuation is random according the probabilities.
// PHEROMON IS UPDATED BY EACH ANT.
// -----------------------------------------------------------
exec SCIHOME/ScIntro.sce, mode(0)

N=100;              // number of iterations
Ni=10;              // # of ants in one step
fi=.5;              // forgetting
D=[                 // distances
 0 17 10  3 19
17  0 32 12 21
10 32  0 16 43
 3 12 16  0 15
19 21 43 15  0
];
n=max(size(D));      // number of towns to walk round
P=1e-8*ones(n,n);                  // probs - init
F=1*rand(n,n,'u')+.1*ones(n,n);    // pheromone - init
tt=[]; St=%inf;

for i=1:N  // LOOP FOR ITERATIVE STEPS
  S0=%inf;

  for r=1:Ni   // LOOP FOR ANTS IN ONE STEP
    I=ceil(n*rand(1,1,'u'));       // starting node
    Tabu=I;                        // inital tabu list
    Free=setdiff(1:n,Tabu);        // initial free nodes

    for j=2:n  // INDIVIDUAL ANT IN ONE STEP
      p=P(I,Free)/sum(P(I,Free));  // probabilities from place I
//      [m,z]=max(p);                // maximum probabilty direction
      z=sum(rand(1,1,'u')>cumsum(p))+1; // HERE IS THE DIFFERENCE
      J=Free(z);                   // next place
      Tabu=[Tabu J];               // tabu updated
      Free=setdiff(1:n,Tabu);      // free nodes updsted
      I=J;                         // end is beginning for next step
    end                            //   in constructing the rout

    Tabu=[Tabu Tabu(1)];           // closing the cycle rout
    S=0;                           // actual length of the path
    for j=1:n
      S=S+D(Tabu(j),Tabu(j+1));
    end
    if S<S0                        // if shorter, remember it
      S0=S;                        //   within one step of iterations
      T0=Tabu;
```

```
    end

    for j=1:n
      F(Tabu(j),Tabu(j+1))=fi*F(Tabu(j),Tabu(j+1))+1/S;     // updt by best from
    end                                                     //  current step

  end // of for r (loop for ants in one step)

  P=F;    // matrix of unnormalized probabilities (= pheromone)

  tt=[tt; T0(1:n)];    // remember paths
  st(i)=S0;            // remember lengths
  if S0<St
    St=S0;
    Tt=T0;
  end

end // of for i (loop for iterative steps)

// Results
//roundd(P),
St,Tt
S1=0;
for i=1:n
  S1=S1+(D(Tt(i),Tt(i+1)));
end
S1

set(scf(1),'position',[800 200 600 400])
plot(st,'.-','markersize',3)
```

7.5 Genetické algoritmy

Napodobují děje, které se dějí u lidí kdy děti dědí vlastnosti po rodičích. Algoritmus lze použít při hledání optimální sekvence určitých prvků - u lidí jsou to alely v genu. Máme obrovskou množinu takových sekvencí a hledáme tu nejlepší podle daného kriteria. Algoritmus pracuje takto:

1. Zvolíme určitou podmnožinu sekvencí - tzv. **populaci.**
2. Každý prvek populace ohodnotíme podle kriteria a sekvence seřadíme podle kvality.
3. Seřazenou populaci rozdělíme na několik částí (tady je určitá volnost volby). Generujeme novou populaci.
   (a) Skupinu nejlepších ponecháme.
   (b) Na další skupinu aplikujeme metodu cross-over - v náhodně vybraném bodě dvě sekvence rozpojíme a spojíme křížem.
   (c) Další skupinu podrobíme mutaci - náhodně vybraný prvek zaměníme za jiný.
   (d) Skupinu nejhorších nahradíme náhodným výběrem z velké množiny všech sekvencí.

Z daného postupu je patrno, že v populacích se budou hromadit dobrá řešení. Tím ale, že do populací zahrneme i skupiny nových sekvencí, dáváme šanci dalším lepším řešením, které na začátku řešení nebyly dostupné.

Ukázka genetického algoritmu j v následujícím programu.

```
// Example of genetic algorithm
// - find sequence with the minimal sum of entries
// ------------------------------------------------
exec('SCIHOME/ScIntro.sce',-1); mode(2); format('v',6)

nX=30;                   // length of basic set
nP=10;                   // length of population
nx=15;                   // length of sequences
nite=20000;                // number of iterations
for i=1:nP
  P(i,:)=samp(nx,1:nX)'; // first population
end
for ite=1:nite           // ITERATIONS
  sx=sum(P,2);           // evaluation of population
  [sxS,jS]=gsort(sx,'g','i'); // ordering
  P=P(jS,:);             // ordered population
  Pn1=P(1:2,:);          // group 1
  x1=P(3,:);
  x2=P(4,:);
  k=8;
  Pn2=[x1(1:k) x2(k+1:nx)// group 2
       x2(1:k) x1(k+1:nx)];
  z=samp(2,1:nx);
  Pn3=P(5:6,:); Pn3(1,2)=z(1); Pn3(2,4)=z(2); // gr. 4
  for j=1:4
    Pn4(j,:)=samp(nx,1:nX)';
  end
  P=[Pn1; Pn2; Pn3; Pn4];// new population
  Su(ite)=sum(P(1:2,:)); // evaluation of the first group
end
set(scf(),'position',[600 100 400 300])
plot(Su)
title('Evolution of the criterion during iterations')
disp('Optimal sequence')
xopt=P(1,:)
disp('optimal criterion')
copt=sum(xopt)
```