Lineární programování II

Pavla Pecherková, Šárka Jozová, Ivan Nagy

Fakulta dopravní
ČVUT, Praha

Obsah

Kapitola 1

Celočíselné programování

Úloha celočíselného programování je prakticky stejná jako úloha lineárního programování. Navíc se ale požaduje, aby řešení některých proměnných ve vektoru x bylo celočíselné.

1.1 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

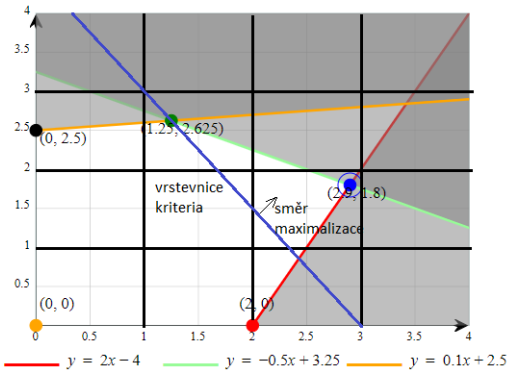
$$3x_1 + 2x_2 \to \max$$
$$2x_1 - x_2 \leq 4$$
$$2x_1 + 4x_2 \leq 13$$

$$-x_1 + 10x_2 \leq 25$$
$$x_1 \geq 0,\ x_2 \in N$$

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.

1.2 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.

1.2.1 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

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

Omezení

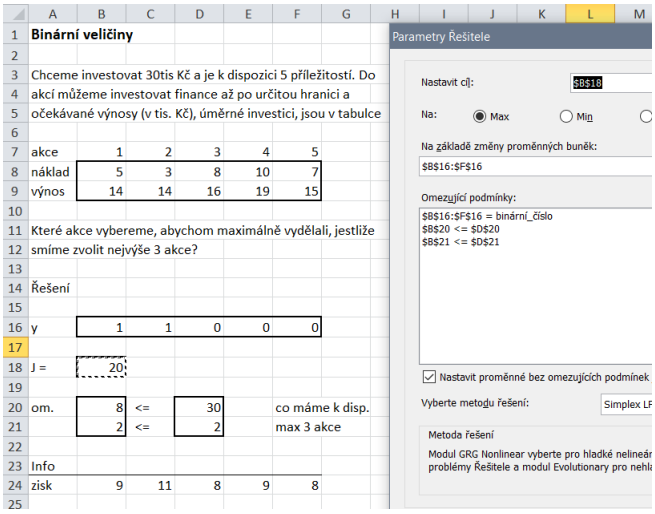
– k dispozici 30tis.

$$\sum n_i \leq 30$$

– max. 3 akce

$$\sum y_i \leq 3$$

Excel: P01_alespon.xlsx



1.2.2 Indikace nenuly

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

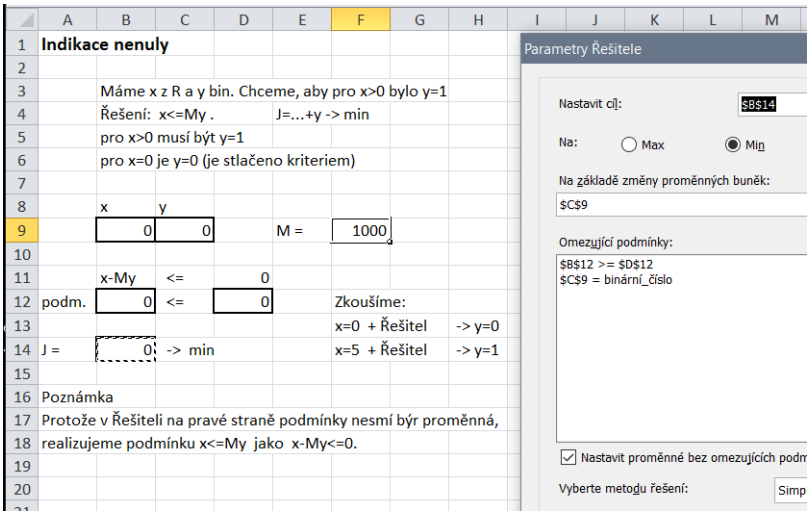
Definujeme $y \in \{0, 1\}$ - binární, indikátor nenulovosti x a zavedeme podmínku

$$My \geq x$$

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

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

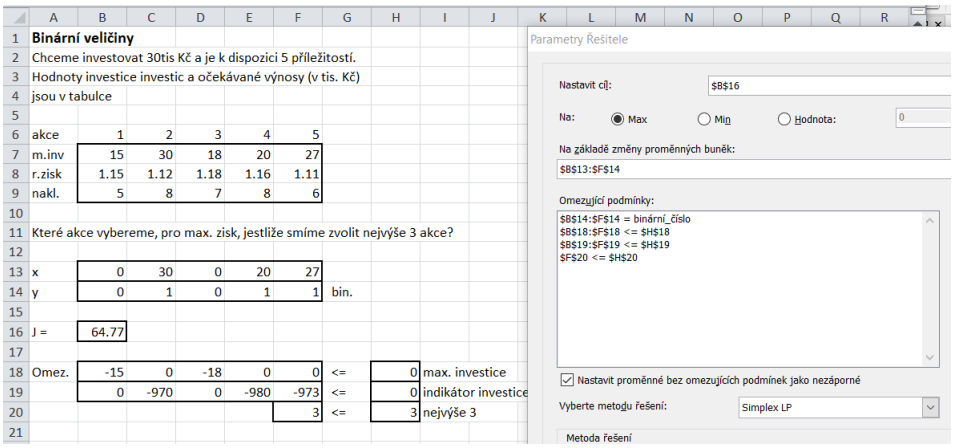
– indikátor pro $x > 0$

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

– max. 3 akce

$$\sum_i y_i \leq 3$$

Excel: Pr02_indikace2.xlsx



1.2.3 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 nebo druhou akci, dostaneme navíc ještě 5tis. Kč. Které akce vybereme, abychom maximálně vydělali, 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

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

Omezení

– finance k dispozici pro $y_1 = 0$

$$\left(\sum_i y_i n_i\right) - My_1 \leq 30$$

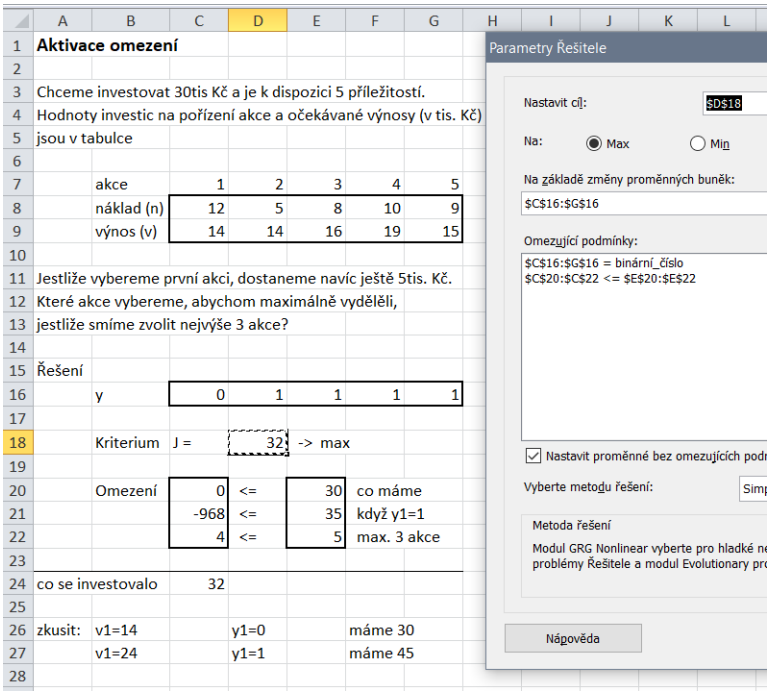
– finance k dispozici pro $y_1 = 1$

$$\left(\sum_i y_i n_i\right) - M\left(1 - y_1\right) \leq 33$$

– max. 4 akce

$$\sum_i y_i \leq 4$$

Excel: Pr03_aktivace.xlsx



	A	B	C	D	E	F	G
1	**Aktivace omezení**						
2							
3	Chceme investovat 30tis Kč a je k dispozici 5 příležitostí.						
4	Hodnoty investic na pořízení akce a očekávané výnosy (v tis. Kč)						
5	jsou v tabulce						
6							
7		akce	1	2	3	4	5
8		náklad (n)	12	5	8	10	9
9		výnos (v)	14	14	16	19	15
10							
11	Jestliže vybereme první akci, dostaneme navíc ještě 5tis. Kč.						
12	Které akce vybereme, abychom maximálně vyděleli,						
13	jestliže smíme zvolit nejvýše 3 akce?						
14							
15	Řešení						
16		y	0	1	1	1	1
17							
18		Kriterium	J =	32	-> max		
19							
20		Omezení	0	<=	30	co máme	
21			-968	<=	35	když y1=1	
22			4	<=	5	max. 3 akce	
23							
24	co se investovalo		32				
25							
26	zkusit:	v1=14		y1=0		máme 30	
27		v1=24		y1=1		máme 45	
28							

1.2.4 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

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

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 1.2.1).

Poznámka: *Podmínku „omezení platí" realizujeme jako „omezení je aktivní" - viz odstavec 1.2.3.*

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 $\sum y \leq 1$)

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

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

$$y = [y_1, y_2]$$

který bude použit v realizaci implikace.

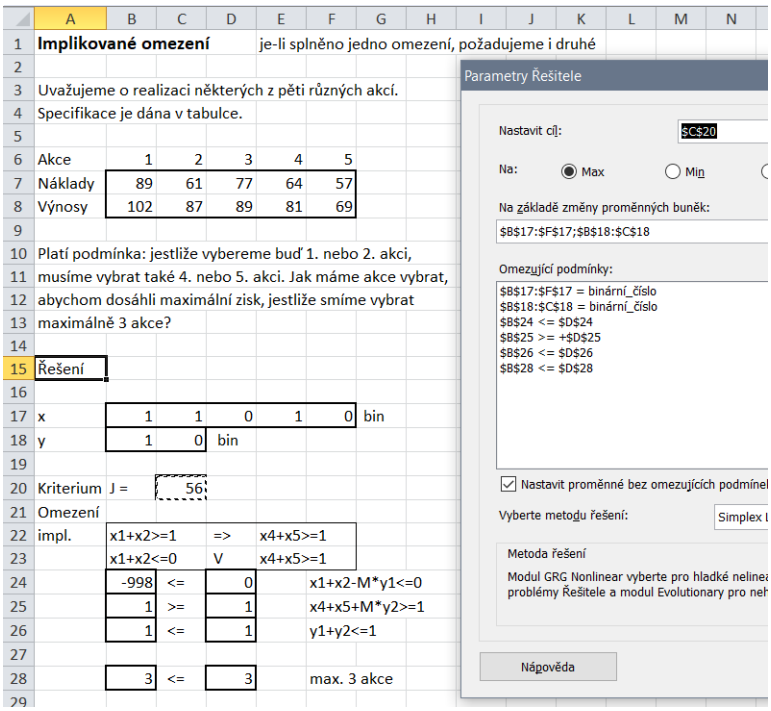
Kriterium

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

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.



1.2.5 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:

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

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

Modelujeme takto

1. oblast

$$\begin{aligned} a_1'x - M_1y_1 &\le b_1, \\ a_2'x - M_2y_1 &\le b_2, \\ a_3'x - M_3y_1 &\le b_3 \end{aligned}$$

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

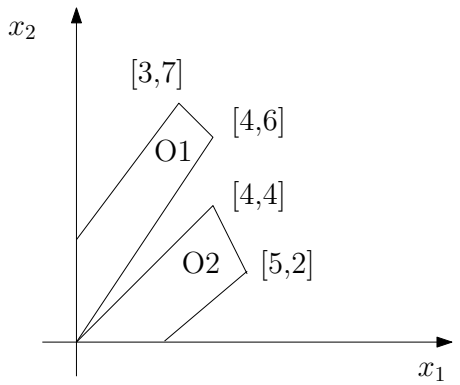
Máme dvě oblasti ohraničené přímkami

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

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

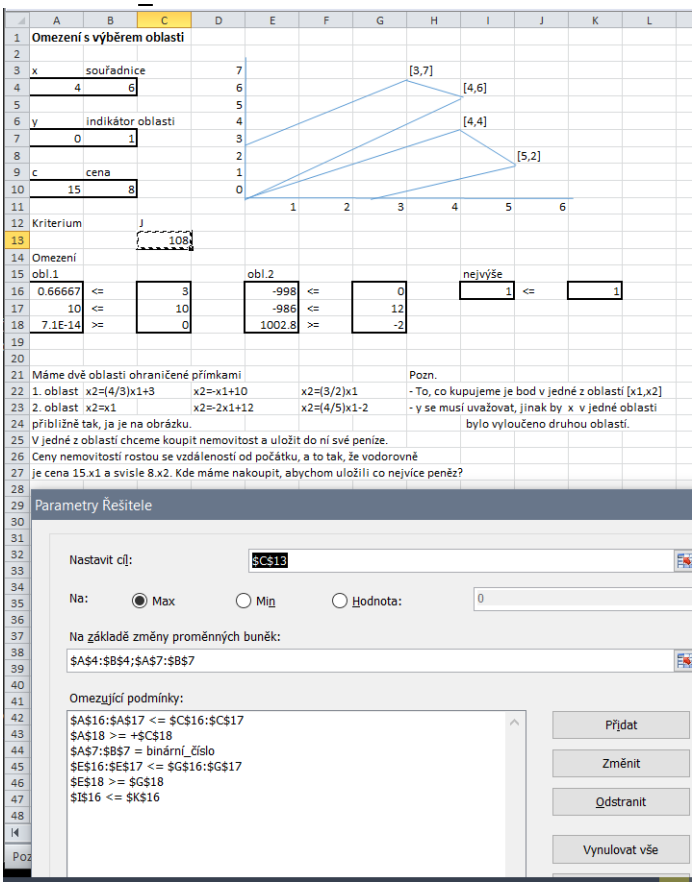
$$x_2 - My_2 \leq x_1$$
$$x_2 - My_2 \leq -2x_1 + 12$$
$$x_2 + My_2 \geq \frac{4}{5}x_1 - 2$$

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



1.3 Triky v kriteriu

1.3.1 Fixní náklady

Pokud budeme modelovat případnou realizaci určité výroby, pak náklady na výrobu se sestává 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

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

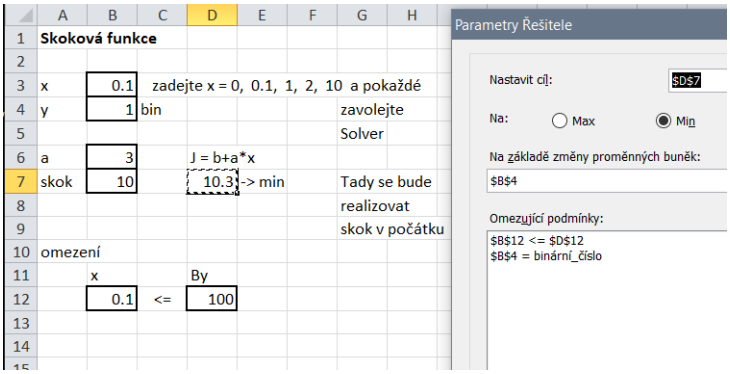
[3] chceme minimální náklady

a podmínky

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

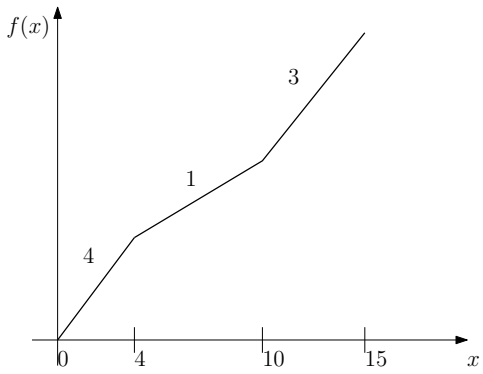
S touto podmínkou jsme se již setkali u „indikace nenuly“, viz odstavec 1.2.2. Pokud je $x > 0$ musí být $y = 1$. Penalizace v kriteriu zaručí, že pro $x = 0$ bude také $y = 0$.

Excel: Pr11_skokFunkce.xlsx



1.3.2 Po částech lineární reprezentace

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



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

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

$w_1 = w_2 = 0$	$w_1 = 1$ 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 1.1: Po částech lineární funkce

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

$$\begin{aligned} 4w_1 \leq \quad & \delta_1 \quad \leq 4 \\ 6w_2 \leq \quad & \delta_2 \quad \leq 6w_1 \\ 0 \leq \quad & \delta_3 \quad \leq 5w_2 \end{aligned} \tag{1.3.1}$$

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

Důkaz, že předpis platí je ukázán v tabulce 1.1.

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:

Máme x a chceme pro něj určit $J = f(x)$.

Zavedeme stavové proměnné $w \in \{0, 1\}$ a $d \geq 0$, které optimalizujeme.

Pomocí w (1.3.1) 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.

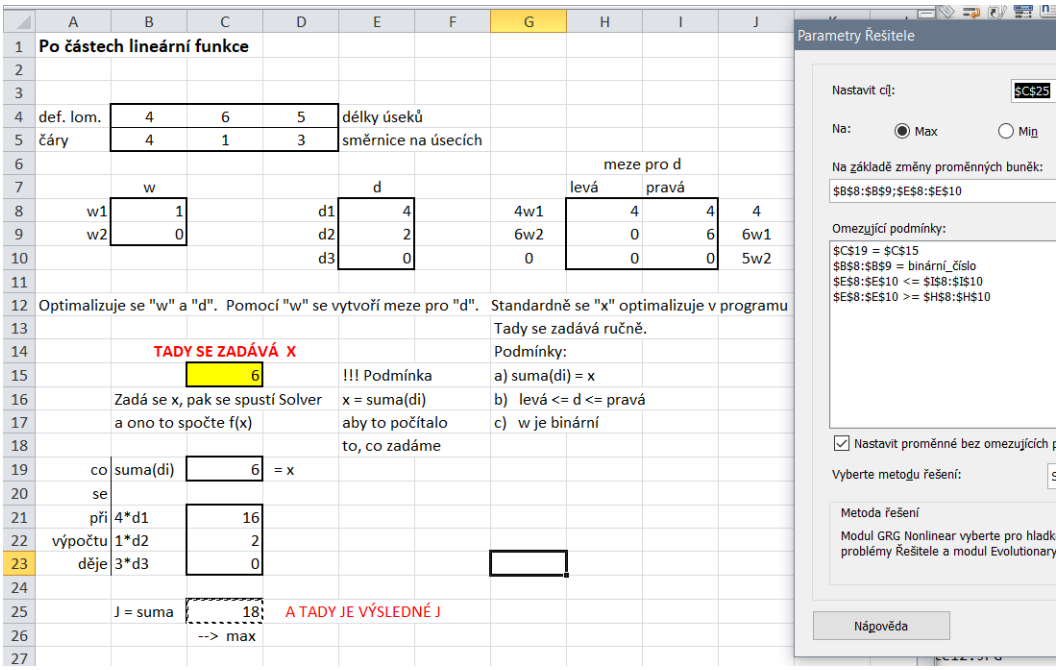
Poznámka

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)



	A	B	C	D	E	F	G	H	I	J
1	**Po částech lineární funkce**									
2										
3										
4	def. lom.	4	6	5	délky úseků					
5	čáry	4	1	3	směrnice na úsecích					
6								meze pro d		
7		w			d			levá	pravá	
8	w1	1		d1	4		4w1	4	4	4
9	w2	0		d2	2		6w2	0	6	6w1
10				d3	0		0	0	0	5w2
11										
12	Optimalizuje se "w" a "d". Pomocí "w" se vytvoří meze pro "d".						Standardně se "x" optimalizuje v programu			
13							Tady se zadává ručně.			
14		TADY SE ZADÁVÁ X					Podmínky:			
15			6		!!! Podmínka		a) suma(di) = x			
16		Zadá se x, pak se spustí Solver			x = suma(di)		b) levá <= d <= pravá			
17		a ono to spočte f(x)			aby to počítalo		c) w je binární			
18					to, co zadáme					
19	co	suma(di)	6	= x						
20	se									
21	při	4*d1	16							
22	výpočtu	1*d2	2							
23	děje	3*d3	0							
24										
25		J = suma	18	A TADY JE VÝSLEDNÉ J						
26			--> max							
27										

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 $f(x)$

$$f(x) = \begin{cases} x & \text{pro } x \in (0, 10) \\ \frac{5}{4}x - \frac{5}{2} & \text{pro } x \in (10, 50) \\ \frac{1}{2}x + 35 & \text{pro } x \in (50, 100) \end{cases} \tag{1.3.2}$$

Kolik máme vyrábět, abychom dosáhli maximálního zisku?

Řešení

Jako stavové proměnné zavedeme

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

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

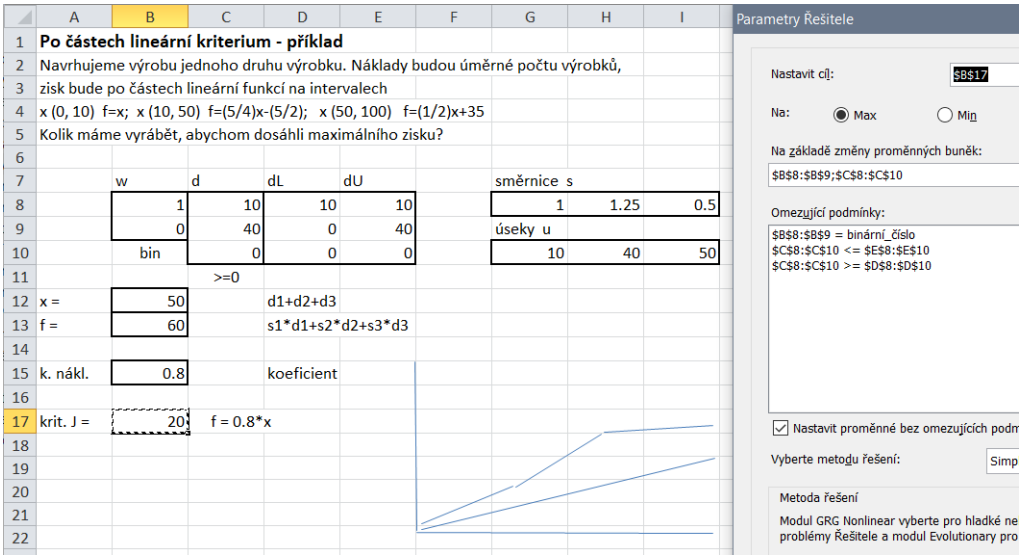
Dále sestavíme dolní a horní meze pro δ

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

Kriterium bude dáno rozdílem „výnos“ - „náklady“

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

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									

1.3.3 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 (1.3.2) 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

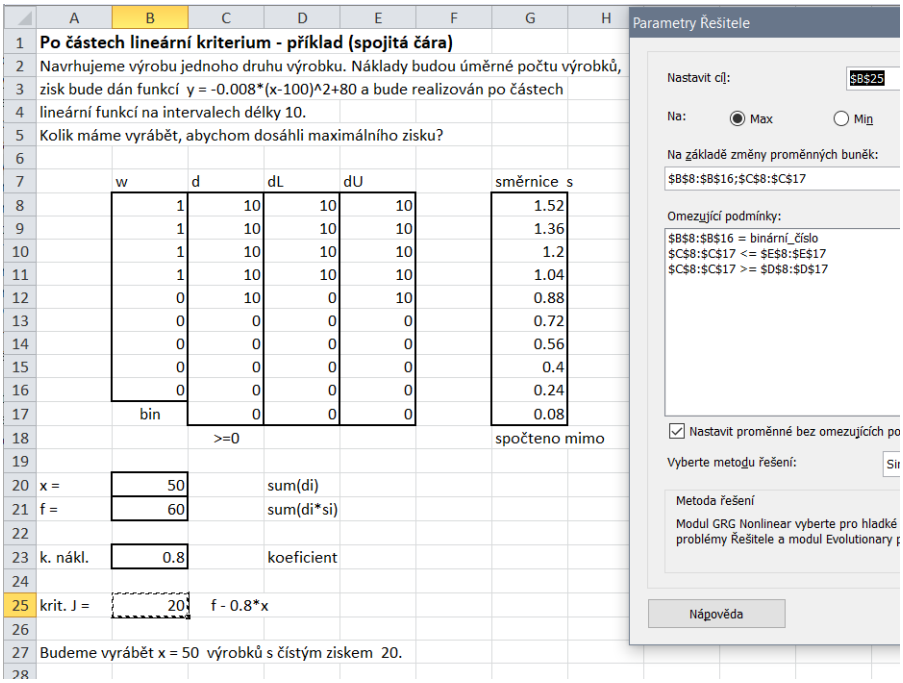
$$\frac{d}{dx}\left[-0.008(x - 100)^2 + 80\right] - 0.8x = 0$$

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

$$x - 100 = -50$$

$$x = 50$$

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								

1.3.4 Součin v kritériu

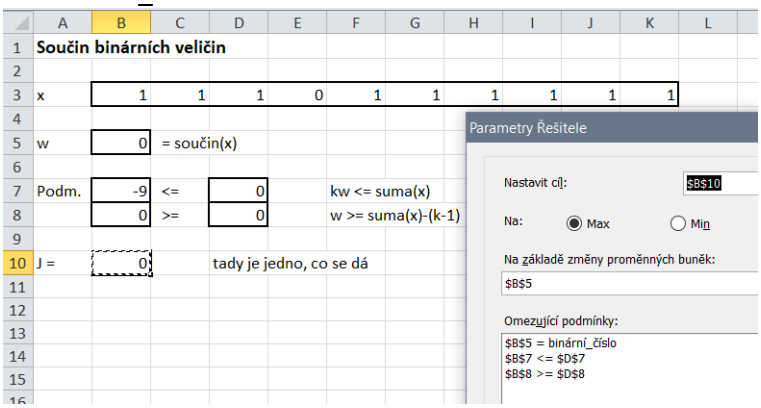
Binární veličiny

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 $0, 1, \cdots, k-1$ (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 $x_1 x_2 \cdots x_k$.

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.

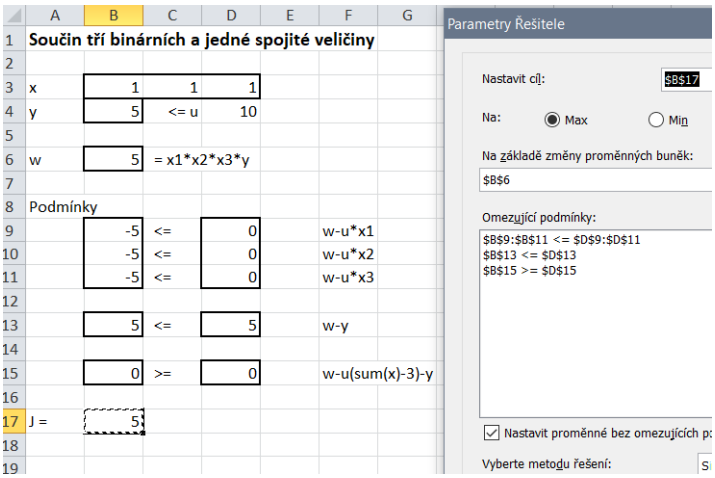
NEBO

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



1.3.5 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

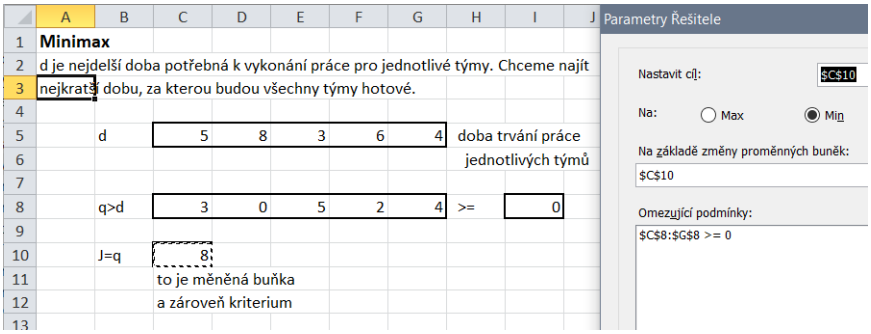
$$\begin{aligned} p_1 &\leq q \\ p_2 &\leq q \\ &\cdots \\ p_k &\leq q \end{aligned}$$

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 pětice lidí $x_1 = [x_1, x_2 \cdots x_5]_1$, $x_2\,[x_1, x_2 \cdots x_5]_2$, $x_3\,[x_1, x_2 \cdots x_5]_3$, 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 $[x_j]_i$ a $y_i = [y_1, y_2, y_3, y_4, y_5]_i \in \{0, 1\}$, $i = 1, 2, 3$ - indikátor účasti člověka $[x_j]_i$ na práci. Zavedeme je jako matice s prvky x_{ij} a y_{ij}.

Pokud člověk $[x_j]_i$ nebude vybrán na práci, bude jeho $x_{ij} = 0$ a 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.

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

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

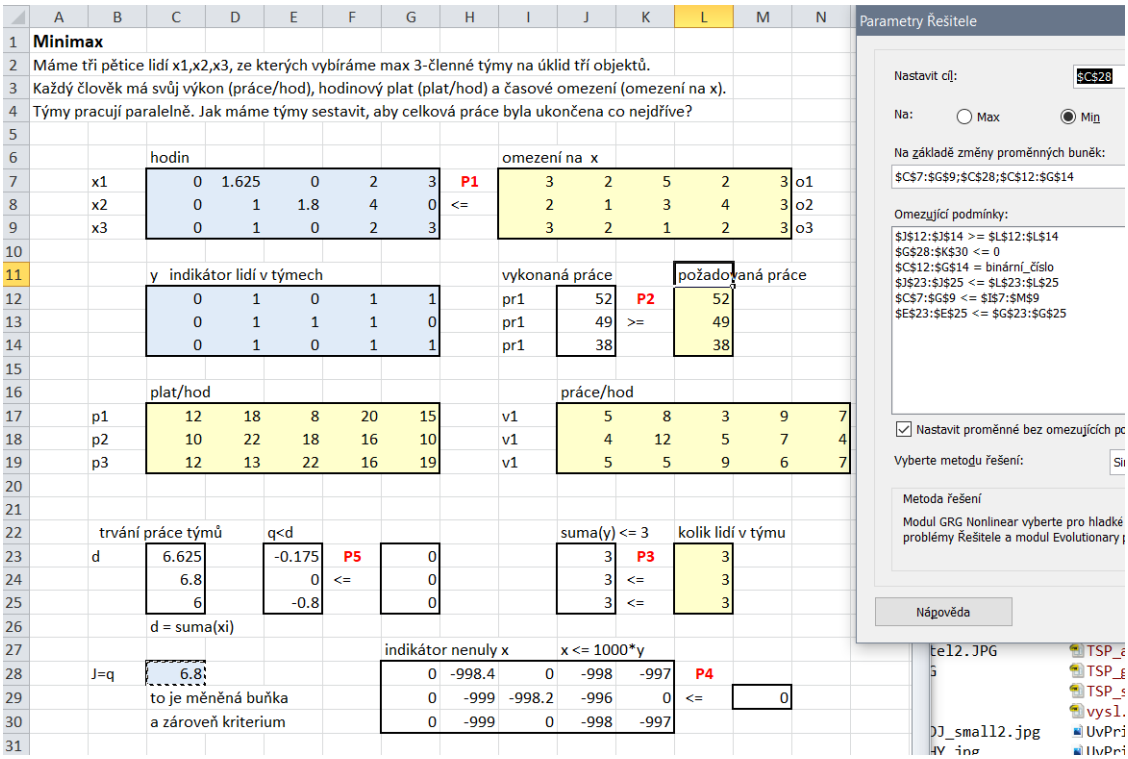
Celková doba J se bude rovnat minimální hodnotě q pro kterou platí

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

a tedy kriterium bude

$$J = q \rightarrow \min$$

Excel: Pr14_minimax2.xlsx.



Minimax

Máme tři pětice lidí x1,x2,x3, ze kterých vybíráme max 3-členné týmy na úklid tří objektů.
Každý člověk má svůj výkon (práce/hod), hodinový plat (plat/hod) a časové omezení (omezení na x).
Týmy pracují paralelně. Jak máme týmy sestavit, aby celková práce byla ukončena co nejdříve?

	hodin						omezení na x					
x1	0	1.625	0	2	3	P1	3	2	5	2	3	o1
x2	0	1	1.8	4	0	<=	2	1	3	4	3	o2
x3	0	1	0	2	3		3	2	1	2	3	o3

	y indikátor lidí v týmech						vykonaná práce			požadovaná práce
	0	1	0	1	1		pr1	52	P2	52
	0	1	1	1	0		pr1	49	>=	49
	0	1	0	1	1		pr1	38		38

	plat/hod						práce/hod				
p1	12	18	8	20	15		v1	5	8	3	9	7
p2	10	22	18	16	10		v1	4	12	5	7	4
p3	12	13	22	16	19		v1	5	5	9	6	7

	trvání práce týmů		q<d				suma(y) <= 3		kolik lidí v týmu
d	6.625		-0.175	P5	0		3	P3	3
	6.8		0	<=	0		3	<=	3
	6		-0.8		0		3	<=	3

d = suma(xi)

			indikátor nenuly x			x <= 1000*y			
J=q	6.8		0	-998.4	0	-998	-997	P4	
	to je měněná buňka		0	-999	-998.2	-996	0	<=	0
	a zároveň kriterium		0	-999	0	-998	-997		

Kapitola 2

Modely celočíselného programování

2.1 Problém batohu

(Knapsack problem)

Jedná se o úlohu z třídy investičního rozhodování.

Příklad

Jedeme na výlet stopem a balíme si s sebou věci do batohu. Objem (nosnost) batohu je omezený hodnotou M. Jednotlivé věci mají svůj objem (váhu) m_i a důležitost d_i a celkem jich je n, tedy $i = 1, 2, \cdots, n$. Chceme s sebou vzít co nejvíce důležitých věcí tak, aby nebyla překročena kapacita batohu.

Řešení

Jako stavový vektor zvolíme $x = [x_1, x_2, \cdots, x_n] \in \{0, 1\}$, jehož prvky budou indikovat vybrané věci.

Kritérium

$$J = \sum_{i=1}^{n} d_i x_i \to \max$$

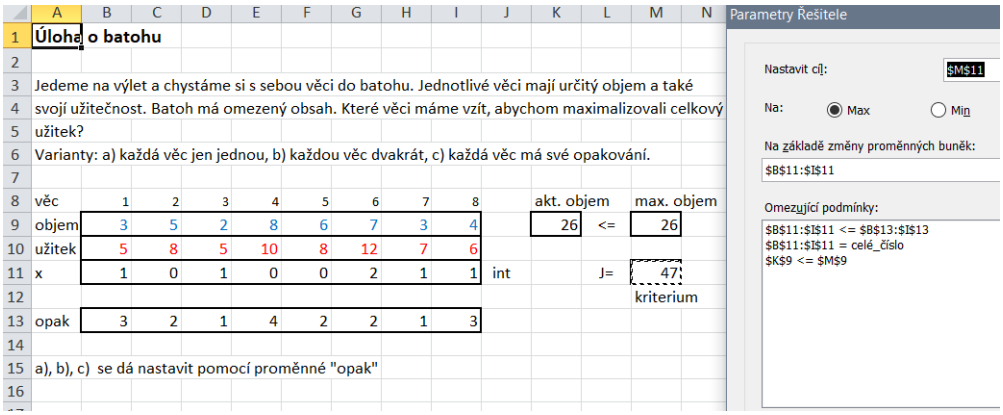
Omezení

$$\sum_{i=1}^{n} m_i x_i \leq M$$

$$x_i \in \{0, 1\}, \ \forall i$$

LIPS: Podmínky zadáme do programu LIPS (Kapsack0.lpx) nebo pro více věcí najednou (Kapsack1.lpx)

Excel: U12a_batoh.xlsx



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Úloha o batohu**													
2														
3	Jedeme na výlet a chystáme si s sebou věci do batohu. Jednotlivé věci mají určitý objem a také													
4	svojí užitečnost. Batoh má omezený obsah. Které věci máme vzít, abychom maximalizovali celkový													
5	užitek?													
6	Varianty: a) každá věc jen jednou, b) každou věc dvakrát, c) každá věc má své opakování.													
7														
8	věc	1	2	3	4	5	6	7	8		akt. objem		max. objem	
9	objem	3	5	2	8	6	7	3	4		26	<=	26	
10	užitek	5	8	5	10	8	12	7	6					
11	x	1	0	1	0	0	2	1	1	int		J=	47	
12													kriterium	
13	opak	3	2	1	4	2	2	1	3					
14														
15	a), b), c) se dá nastavit pomocí proměnné "opak"													
16														

2.2 Výběr projektu

(Project Selection)

Základní úlohou celočíselného programování je výběr projektů z několika nabízených tak, aby byly splněny dodatečné podmínky. Výběr se provádí celočíselnou proměnnou, kde 1 znamená vybrat a 0 odmítnout.

Formulace úlohy:

Máme možnost investic do několika projektů. Z každé investice máme očekávaný zisk a každá vyžaduje určité zdroje (pracovní síly, suroviny atd.), které jsou omezené. Investici buď realizujeme nebo ne. Jak vybrat investice aby přinesly maximální zisk a nebyl překročen omezený zdroj financí?

Zápis úlohy je následující:

$$c'x \to \max$$

$$ax \leq b$$

$$x \in \{0, 1\}$$

kde: $c = [c_1, c_2, \cdots c_n]$ jsou očekávané výnosy projektů, $a = [a_1, a_2, \cdots a_n]$ jsou náklady spojené s realizací projektů, b je omezení na finance, které jsou k dispozici a $x = [x_1, x_2, \cdots x_n]$ je binární proměnná, jejíž složky indikují vybraný projekt.

Příklad

K dispozici jsou 4 akce 1, 2, 3, 4. Z každé plyne očekávaný výnos

$$c = [8, 11, 6, 4]$$

a požaduje určitou investici

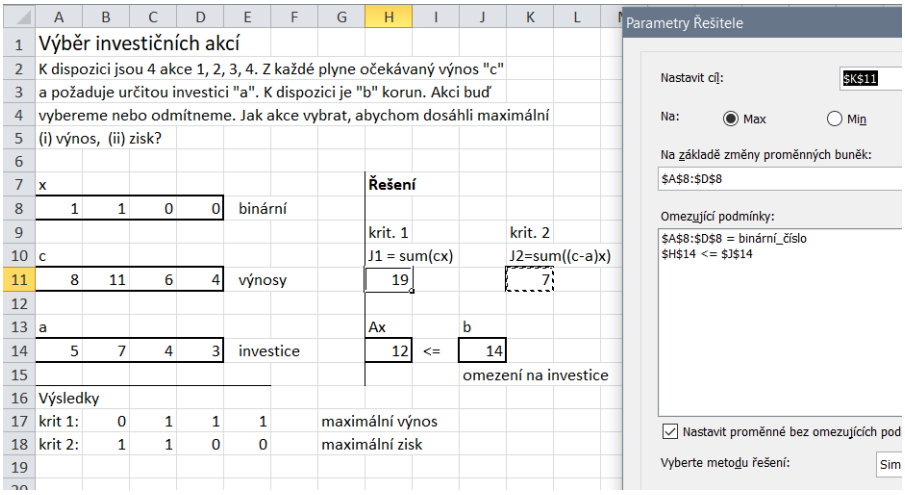
$$a = [5, 7, 4, 3]$$

K dispozici je $b = 14$tis. korun. Akci buď vybereme nebo odmítneme. Jak akce vybrat, abychom dosáhli maximální (i) výnos, (ii) zisk?

Řešení

Zavedeme binární proměnnou x_j. Dále označíme c_j - výnosy investice j, b_i - omezení na zdroje, a_j - jsou požadavky investice j na zdroj financí.

Excel: U11a_vyberAkce.xlsx



Další rozšíření příkladu jsou:

U11b_vyberAkce.xlsx - přidání podmínky „nejvýše 2 akce”,

U11c_vyberAkce.xlsx - přidání podmínky „jestliže akce jedna pak i akce 4”

U11d_vyberAkce.xlsx - přidání podmínky „buď 1 a 2 nebo 3 a 4”

2.3 Výběr projektu s několika zdroji.

Cílem je rozhodnout o určitém množství investičních akcí. Akce se buď podpoří celá, nebo se nepodpoří vůbec. Akcím přiřadíme stavový vektor

$$x = [x_1, x_2, \cdots, x_n], \quad x_j \in \{0, 1\} \ \forall j = 1, 2, \cdots, n$$

tak, že $x_i = 1$ znamená podporu akce i, $x_j = 0$ znamená, že akce j nebude podpořena.

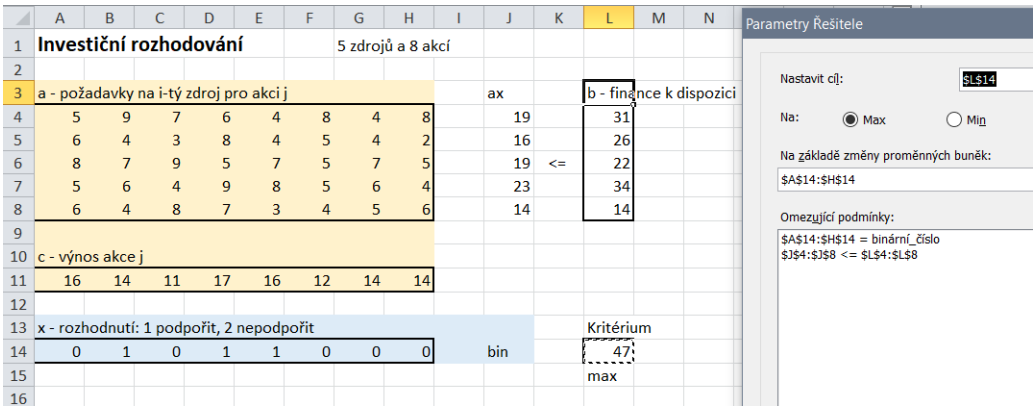
Zisk plynoucí z jednotlivých akcí označíme c_j, $j = 1, 2, \cdots, n$. Akce vyžadují m různých zdrojů (finance, lidské síly, suroviny atd.). Náklady na i-tý zdroj pro akci j označíme $a_{i,j}$ a množství jednotlivých zdrojů, které jsou k dispozici b_i (omezení jsou psány pro jednotlivé zdroje).

Zápis standardní úlohy je následující:

$$\sum_{j=1}^{n} c_j x_j \to \max$$

$$\sum_{j=1}^{n} a_{i,j}x_j \leq b_i, \quad x_j \in \{0,1\}, \; i = 1, 2, \cdots, m$$

Excel: U13a_investice.xlsx



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Investiční rozhodování**						5 zdrojů a 8 akcí							
2														
3	a - požadavky na i-tý zdroj pro akci j									ax		b - finance k dispozici		
4	5	9	7	6	4	8	4	8		19		31		
5	6	4	3	8	4	5	4	2		16		26		
6	8	7	9	5	7	5	7	5		19	<=	22		
7	5	6	4	9	8	5	6	4		23		34		
8	6	4	8	7	3	4	5	6		14		14		
9														
10	c - výnos akce j													
11	16	14	11	17	16	12	14	14						
12														
13	x - rozhodnutí: 1 podpořit, 2 nepodpořit											Kritérium		
14	0	1	0	1	1	0	0	0		bin		47		
15												max		
16														

Modifikace základní úlohy:

- Modelování závislostí projektů - realizace projektu i je závislá na realizaci projektu j. Lze modelovat jako
$$x_j \geq x_i.$$
- Výběr nejvýše jednoho ze skupiny
$$x_1 + x_2 + x_3 + x_4 \leq 1$$

2.4 Úlohy o množinách

Existuje m požadavků R_i a n aktivit A_j takových, že v celkovém sjednocení pokrývají všechny požadavky, nikoliv ale jednotlivě. Cílem je najít takovou podmnožinu aktivit, která ve sjednocení pokryje všechny požadavky a minimalizuje určité kriterium (např. náklady na využité aktivity).

Existují 3 varianty tohoto problému: (1) Set **covering**, (2) Set **partitioning** a (3) Set **packing**. Jejich formulace jsou následující (liší se v nerovnítku podmínek):

1. Covering
$$c'x \to \min$$
$$Ax \geq 1$$
$$x \in (0,1)$$
2. Partitioning
$$c'x \to \min$$
$$Ax = 1$$
$$x \in (0,1)$$

3. Packing

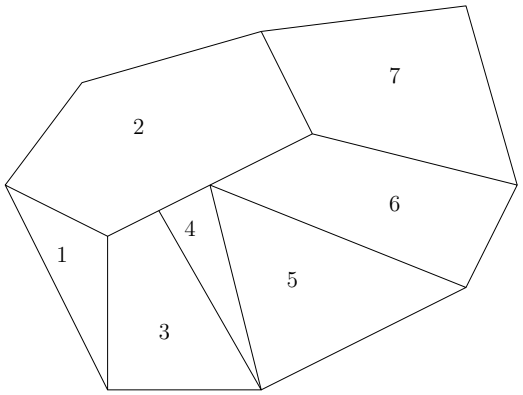
$$c'x \to \max$$

$$Ax \leq 1$$
$$x \in (0,1)$$

2.4.1 Požární stanice (set covering)

Formulace úlohy

Uvažujme město, které se skládá z několika oblastí. Např. podle obrázku



Každé oblasti je přiřazeno číslo. Ve městě chceme vybudovat požární stanice. Jedna stanice je schopna pokrýt oblast, ve které je postavena a všechny sousední oblasti (tj. takové, co sousedí hranou). Jak postavit stanice, aby jich bylo co nejméně?

Řešení

Oblastem přiřadíme binární proměnné x_1, x_2, $\cdots$, x_7. Tyto proměnné budou mít hodnotu 1, když stanice bude, jinak 0.

Kriterium

$$J = \sum_i x_i \to \min$$

– minimální počet postavených stanic.

Omezení

Nejdříve si ke každé oblasti vypíšeme její sousedy (včetně jí samé). V prvém sloupci uvedeme oblast a vpravo její sousedy. Tedy k oblasti 1 patří sama oblast 1 a její sousedí přes hranu, tj. oblasti 2 a 3.

1	1,2,3
2	2,1,3,4,6,7
3	3,1,2,4
4	4,2,3,5
5	5,4,6
6	6,2,5,7
7	7,2,6

Formulace úlohy bude:

Stav bude vektor $x \in \{0,1\}$ jehož složky budou indikovat čísla oblastí, kde postavíme stanice.

Kriterium

$$J = \sum_i x_i \to \min$$

Omezení (aby byla pokryta oblast 1, musí být v ní nebo v některém sousedovi alespoň jedna stanice; a stejně i u dalších)

$$\begin{gathered} x_1 + x_2 + x_3 \geq 1 \\ x_1 + x_2 + x_3 + x_4 + x_6 + x_7 \geq 1 \\ x_1 + x_2 + x_3 + x_4 \geq 1 \\ x_2 + x_3 + x_4 + x_5 \geq 1 \\ x_4 + x_5 + x_6 \geq 1 \\ x_2 + x_5 + x_6 + x_7 \geq 1 \\ x_2 + x_6 + x_7 \geq 1 \end{gathered}$$

$$x_1 \cdots x_7 \in \{0,1\}$$

Realizace úlohy je následující:

Zavedeme matici A tak, že řádky budou reprezentovat oblasti a v každém řádku budou jedničkami vyznačeny sousedi; zbytek budou nuly. Pro náš příklad tedy bude

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Matice A je

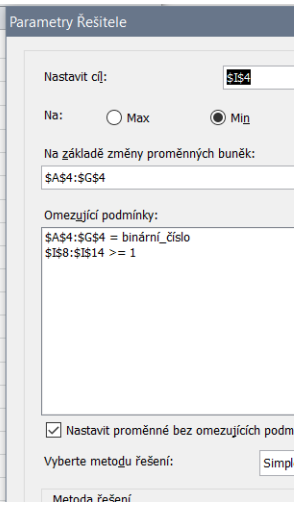
	pokryté oblasti
oblasti	

Pak úloha bude mít tvar, zmíněný na začátku

$$J = \sum x_i \to \min$$
$$Ax \geq 1$$
$$x \in \{0, 1\}$$

Excel: U15a_pozStanice.xlsx

	A	B	C	D	E	F	G	H	I
1	**Pokrytí oblastí požárními stanicemi**								
2									
3	x								Kriterium
4	0	1	0	1	0	0	0		2
5	1	2	3	4	5	6	7		
6	Omezení								
7	A								
8	1	1	1	0	0	0	0		1
9	1	1	1	1	0	1	1		2
10	1	1	1	1	0	0	0		2
11	0	1	1	1	1	0	0		2
12	0	0	0	1	1	1	0		1
13	0	1	0	0	1	1	1		1
14	0	1	0	0	0	1	1		1
15									
16	Celkem 7 oblastí. Řádek odpovídá oblasti, jedničky v řádku jsou oblasti pokryté stanicí v této oblasti.								
17	Pravý sloupec je součet součinů Ax, tj. pro danou konfiguraci x kolikrát je která oblast pokryta.								
18	Tady se vybraly 2 stanice: v oblasti 2 a 1. Pokryty jsou všechny oblasti, 2-4 dokonce dvakrát.								
19									
20									



2.4.2 Kuchařka a recepty (set packing)

Formulace úlohy

Čekáme narychlo ohlášenou návštěvu a chceme ji pohostit. Ve spíži máme 7 ingrediencí do jídla a další už nestačíme pořídit. Nevíme ale, jaké chutě návštěva má, a proto chceme připravit co největší počet jídel. Máme kuchařku a z ní jsme vybrali jídla, která obsahují ty ingredience, které máme k dispozici. Ingredience ale nelze dělit; každou lze použít jen jednou. Která jídla připravíme?

Řešení

Ingredience očíslujeme 1,2,$\cdots$,7 a vybraná jídla označíme písmeny. V tabulce seřadíme jídla a vedle píšeme ingredience které potřebují

jídlo	ingredience
A	1,2
B	1
C	2,5,6
D	3,7
E	2,7
F	3,5
G	4,5,6

Podobně jako u požárních stanic zavedeme matici A. Každému jídlu bude odpovídat jeden sloupec a v něm budou jedničkou označeny potřebné ingredience.

ingredience \ jídla

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Úloha potom bude

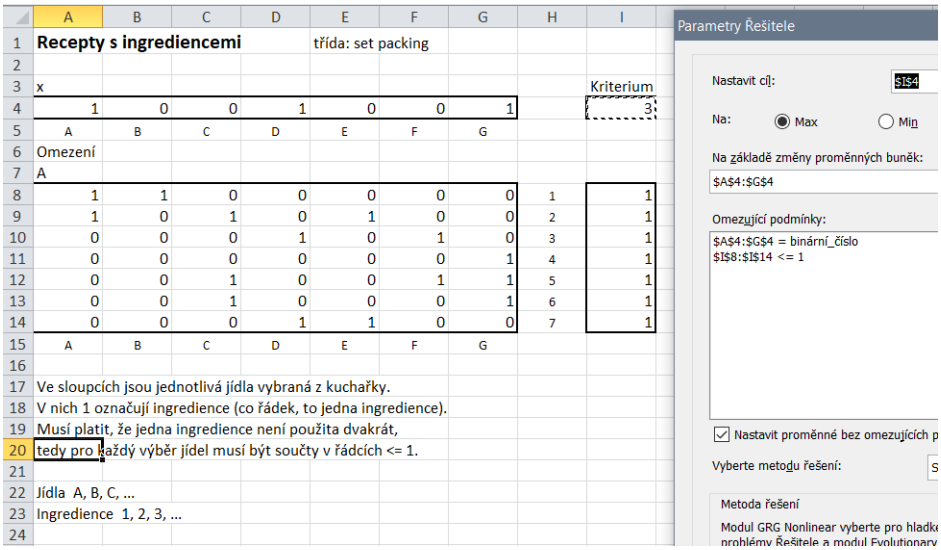
$$J = \sum x_i \to \max$$

$$Ax \leq 1$$

$$x \in \{0, 1\}$$

Program: U15b_recepty.xlsx

Řešení: A, C, D.



2.4.3 Výběr leteckých tras (set partitioning)

Formulace úlohy

Letecká doprava zajišťuje spojení mezi vybranými městy. Tyto spoje nazveme etapy letu. Jeden let (s jednou posádkou letadla) letí po určité trase sestavené z jedné nebo několika etap (pokud

se mezi ně vejde povinný odpočinek posádky a údržba letadla). Tak například let 10.00 z New Yorku do Chicaga a let 18.00 z Chicaga do Los Angeles je příkladem takových etap, které mohou tvořit jednu trasu, kterou absolvuje jedno letadlo s jednou posádkou. Každá trasa má určitou cenu (podle délky, obtížnosti, případně prostojů).

Cílem je ze všech možných (předem sestavených) tras vybrat ty, které budeme realizovat tak, aby

1. žádná etapa nezůstala neobsazena,
2. náklady na realizaci vybraných tras byly minimální.

Nejdříve sestavíme matici a, která bude svými sloupci odpovídat jednotlivým trasám. V řádcích budou všechny etapy. Sloupce matice a budou mít jedničky v řádcích těch etap, které budou trasou realizovány.

Označíme

$x_j \in \{0,1\}$ indikuje, zda trasa j bude realizována (tj. obsazena posádkou),

c_j je cena za pronajatí posádky na trase j (realizaci trasy j),

$a_{i,j} \in \{0,1\}$ označuje (jedničkou), že etapa i je zařazena do trasy j

Úloha je formulována takto

$$\sum_{j=1}^{n} c_j x_j \to \min$$

$$\sum_{j=1}^{n} a_{i,j} x_j = 1,\ i = 1, 2, \cdots, m$$

$$x_j \in \{0,1\},\ j = 1, 2, \cdots, n$$

kde kriterium vyjadřuje minimalizaci nákladů, a první podmínka vyjadřuje skutečnost, že každá etapa má právě jednu posádku.

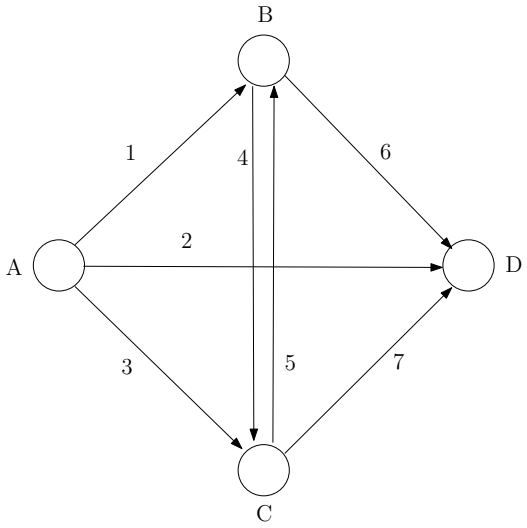
Poznámka

Jestliže připustíme, že členové některé posádky mohou určitou trasu letět jako pasažéři (převáží se na místo svého letu na další etapě), bude mít tato podmínka tvar

$$\sum_{j=1}^{n} a_{i,j} x_j \geq 1,\ \forall i$$

Příklad

Uvažujeme 4 města: A, B, C, D a lety mezi těmito městy podle následujícího obrázku



Jednotlivé šipky na obrázku značí etapy. Jde o to, jak z nich sestavit trasy (obsazené posádkou) tak, aby pronájem posádek by minimální a všechny etapy byly obslouženy. Trasy jsou předem definovány buď jako samostatné lety nebo jsou sestaveny z navazujících etap. V našem příkladě jsou: T1: 1,6; T2: 1,4,7; T3:3,5,6; T4:3,7. Trasy zapíšeme je do sloupců matice A (nejdříve samostatné etapy, potom navržené trasy)

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Řádky a sloupce matice A jsou sestaveny takto

etapy \ trasy

Ceny posádek (tras) vezmeme přibližně rovny počtu etap v příslušné trase a trochu přihodíme na samostatné lety, protože tam musí posádka dojet z domova. Ceny tedy budou

$$c = [15,\ 12,\ 16,\ 14,\ 18,\ 12,\ 16,\ 22,\ 25,\ 27,\ 26]$$

Stavový vektor bude mít 12 prvků (jako je tras), které budou buď 0 nebo 1 - trasu nerealizujeme nebo ano.

Úloha má tvar

$$J = \sum_{j=1}^{n} c_j x_j \to \min$$

$$\sum_{j=1}^{n} A_j x_j = 1$$

$$x_j \in \{0, 1\}$$

Řešení naší úlohy je

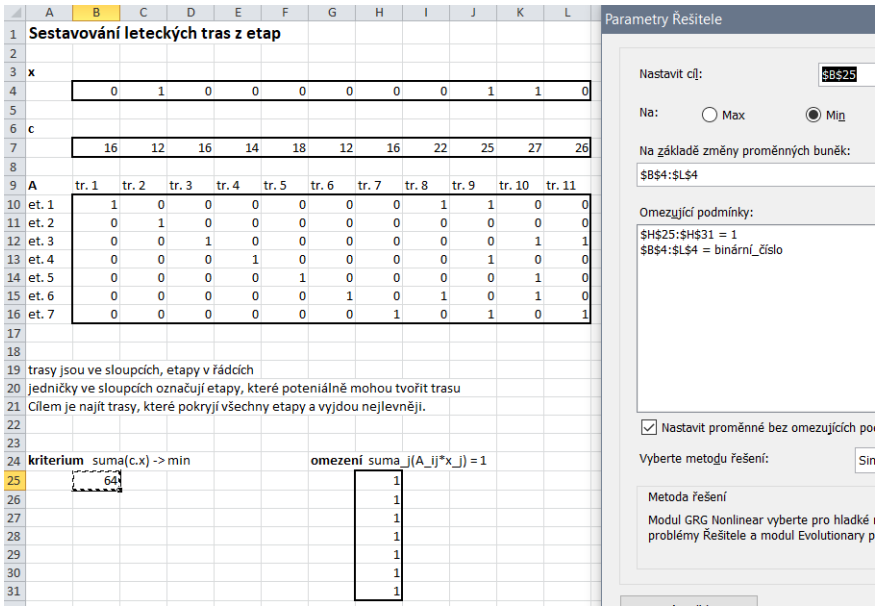
$$x = [0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 1,\ 0]$$

tedy budou realizovány lety (podle matice A)

trasa 1	etapa 2		
trasa 2	etapa 1,	etapa 4,	etapa 7
trasa 3	etapa 3,	etapa 5,	etapa 6

Tedy, všechny etapy jsou obsazeny, poletí 3 posádky a doufejme, že cena za posádky je minimální.

Excel: U15c_letadla.xlsx



2.5 Problém řezání

(Cutting Stock)

2.5.1 Řezání tyčí

Příklad

Máme tyče dlouhé 7.4 m a chceme z nich nařezat kusy dlouhé 150, 210 a 290 cm. Jednotlivé délky potřebujeme v množství 1000 ks. Jak máme postupovat, když chceme mít minimální odpad?

Řešení

Nejprve stanovíme tzv. řezné plány

Délka	Plán řezání					
	1	2	3	4	5	6
290	1	2	-	1	-	1
210	-	-	2	2	1	1
150	3	1	2	-	3	1
Odpad	0	10	20	30	80	90

Optimalizovaný stav je $x = [x_1 \cdots x_6]'$ (int), kde x_i jsou množství řezání podle jednotlivých plánů.

Kriterium

$$J = 10x_2 + 20x_3 + 30x_4 + 80x_5 + 90x_6 \to \min$$

Omezení

$$x_1 + 2x_2 + x_4 + x_6 = 1000$$
$$2x_2 + 2x_4 + x_5 + x_6 = 1000$$
$$3x_1 + x_2 + 2x_3 + 3x_5 + x_6 = 1000.$$

Označíme-li odpad jako

$$c = [0,\ 10,\ 20,\ 30,\ 80,\ 90]'$$

a zavedeme matici A

$$A = \begin{bmatrix} 1 & 2 & 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 2 & 1 & 1 \\ 3 & 1 & 2 & 0 & 3 & 1 \end{bmatrix}$$

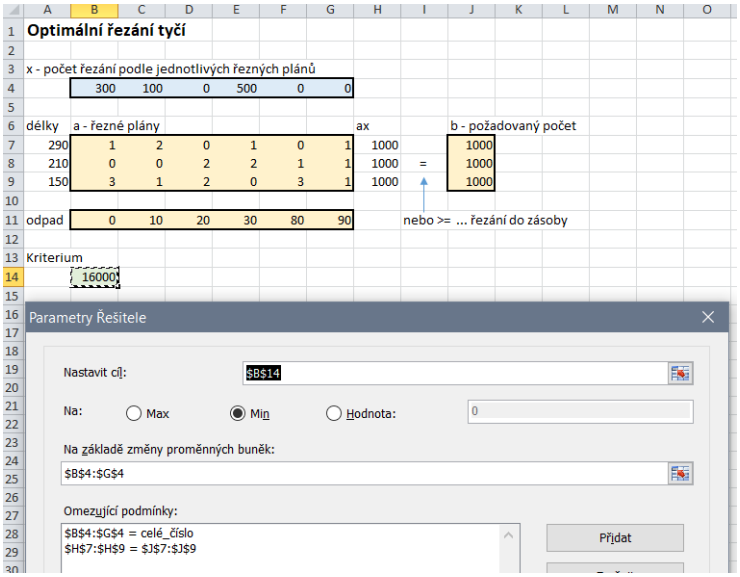
pak úloha má tvar

$$J = c'x \to \min$$
$$Ax = 1000$$
$$x \geq 0\text{-int}$$

Poznámka

Pokud zadáme omezení $Ax \geq 1000$, *pak připustíme řezání do zásoby.*

Excel: U17a_rezaniTyci.xlsx



2.5.2 Řezání papíru

Příklad

Papír se při výrobě balí do rolí o výšce W. Pro další prodej se řeže na pásy o stejné délce jako mají role o předepsaných šířkách $w_1, w_2, \cdots, w_m$. Jednotliví odběratelé si předepisují počet rolí b_j o šířce w_i, $i = 1, 2 \cdots m$. Jak role řezat, aby byl co nejmenší odpad?

Řešte pro $W = 96$, $w = [24,\ 35]$ a $b = [50,\ 65]$.

Řešení

Poznámka: *Úloha je stejná jako pro tyče. Uříznutý kus role dílky* w_i *odpovídá uříznutému kusu tyče. Plocha to toho vlastně nevstupuje.*

Nejprve zvolíme řezné plány a zbytky po řezání Z_j

Plán P	1	2	3
24cm	4	2	1
35cm	0	1	2
Zbytek Z	0	13	2

odkud

$$A = \begin{bmatrix} 4 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Poznámka: *Plány musíme vybrat sami. Zbytky po řezání podle plánu j pak mohou být počítány takto*

$$Z_j = W - \sum_i A_{ij} w_i$$

Zavedeme stav $x = [x_1, x_2, x_3]$ celočíselné, kde x_j je počet řezání podle plánu P_j.

Kriterium je minimální odpad

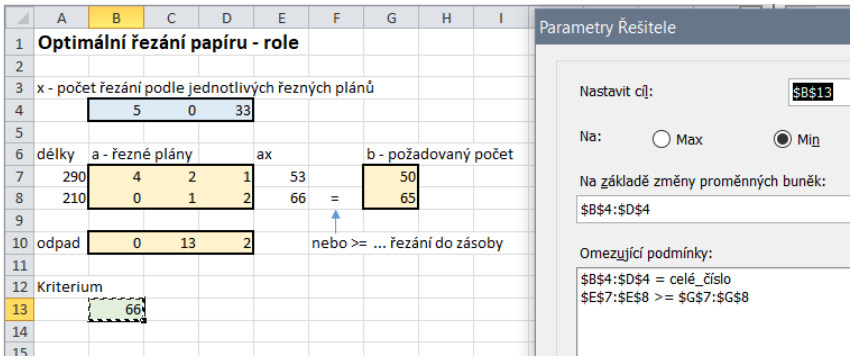
$$J = \sum_j x_j Z_j \to \min$$

Omezení

$$\sum_j A_{ij} x_j \geq b_i$$

$$x \geq 0 \text{ - int.}$$

Excel: U17b_rezaniPapiru1.xlsx



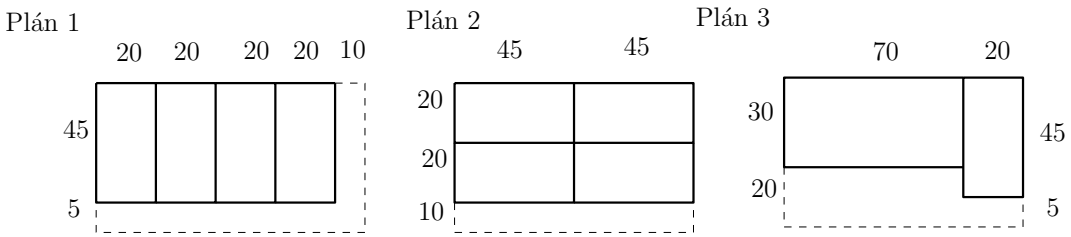
2.5.3 Řezání v ploše

Podobně lze řešit i dvourozměrnou úlohu - vykrajování obdélníků z ploch vyrobeného papíru.

Příklad

Vyráběné papírové obdélníky mají rozměr 50×90 cm. Máme z nich vyříznout b_1 =50 obdélníků o rozměrech 20×45 cm^2 a b_2 =20 obdélníků 30×70 cm^2.

Plány, které přichází v úvahu jsou na obrázku



Dále můžeme uvažovat dva druhy kriteria. U prvního budeme standardně počítat odpad po řezání, u druhého celkovou délku řezu (v případě, kdy řezání je drahé). Dostáváme tak tabulku

Plán P	1	2	3
20×45	4	4	1
30×70	0	0	1
Zbytek Z	900	900	1500
Délka řezu D	260	220	135

Kriterium 1 - zbytek

Protože první a druhý plán má stejné výsledky, dostáváme jen dva plány P_1 a P_3. Tedy stav bude

$$x = [x_1, x_2] \text{ - int.}$$

a kriterium bude

$$J = \sum_{i=1}^{2} Z_i x_i \to \min$$

Omezení

$$Ax \geq b$$

kde $A = \begin{bmatrix} 4 & 1 \\ 0 & 1 \end{bmatrix}$

Kriterium 2 - řezy

Tady se uplatní všechny tři plány, a tedy

$$x = [x_1, x_2, x_3]$$

kriterium

$$J = \sum_{i=1}^{3} D_i x_i \to \min$$

Omezení

$$Ax \geq b$$

kde $A = \begin{bmatrix} 4 & 4 & 1 \\ 0 & 0 & 1 \end{bmatrix}$.

Poznámka

Protože plán 1 a 2 mají stejný počet výsledných obdélníků a plán 1 má delší řez, stačilo by uvažovat jen plány 2 a 3.

Excel: U17c_rezaniPapiru2.xlsx

2.6 Problém obchodního cestujícího

(Travelling Salesman Problem – TSP)

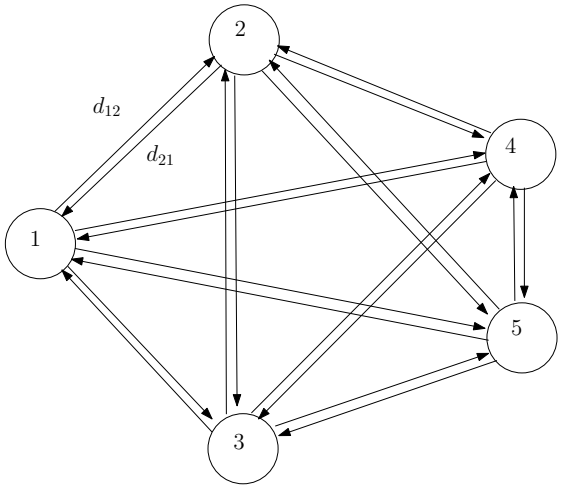
2.6.1 Asymetrický TSP

Obecná formulace úlohy

Uvažujeme n měst spojených navzájem cestami. Každou cestu uvažujeme jako jednosměrnou, obousměrné jsou dvě jednosměrky tam a zpět. Každá z jednosměrek má svou délku (obecně se délka cesty tam nerovná délce zpět). Cílem je najít cestu, kterou má projít obchodní cestující tak, aby každé město navštívil právě jednou a vrátil se do stejného města, ze kterého vyšel. Celá cesta má být nejkratší možná.

Příklad

Úlohu budeme demonstrovat pro 5 měst.



$c_{ij} \geq 0$ značí délky cest (obecně $c_{ij} \neq c_{ji}$),

$x_{ij} \in \{0, 1\}$ znamená $x_{ij} = 1$ cestující půjde z města i do města j; $x_{ij} = 0$ nepůjde.

Stavová matice x bude

$$x = [x_{ij}]\,,\ i \text{ z města}, j \text{ do města}$$

s maticí vzdáleností mezi městy

$$c = \begin{bmatrix} M & 5 & 1 & 8 & 4 \\ 2 & M & 9 & 1 & 1 \\ 1 & 8 & M & 7 & 6 \\ 4 & 9 & 6 & M & 1 \\ 3 & 1 & 6 & 5 & M \end{bmatrix},$$

kde M znamená, že daná dráha neexistuje a tedy ji nebudeme v matici x počítat. Stejně dobře, ale s jednodušší realizací můžeme danou vzdálenost zadat jako nekonečno (v praxi - velké číslo). Potom x bude celá matice, včetně diagonály, ta ale nikdy nebude vybrána, protože cesta je tam příliš dlouhá (tzv. metoda s penalizací).

Řešení

Pro nejkratší cestu musí platit:

Kriterium

$$J = \sum_{ij} c_{ij} x_{ij} \to \min$$

Omezení 1: „města se nesmí opakovat“ → každé město má jen jednu vstupní cestu a jednu výstupní, tj.

$$\sum_{i} x_{ij} = 1, \ \forall j \quad \text{výstupní}$$

$$\sum_{j} x_{ij} = 1, \ \forall i \quad \text{vstupní}$$

Vstupy a výstupy pro uzel 2 jsou znázorněny na následujícím obrázku. Zde je vidět, že skutečně součet ve sloupci je to, co příslušný uzel obdržel od ostatních a součet v řádku je to, co příslušný uzel rozdal mezi ostatní uzly.

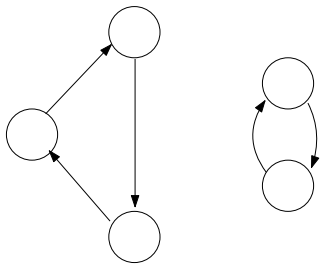
	1	2	3	4	5
1	·	x_{12}	·	·	·
2	·	x_{22}	·	·	·
3	·	x_{32}	·	·	·
4	·	x_{42}	·	·	·
5	·	x_{52}	·	·	·

Vstup do uzlu 2

	1	2	3	4	5
1	·	·	·	·	·
2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
3	·	·	·	·	·
4	·	·	·	·	·
5	·	·	·	·	·

Výstup z uzlu 2

Omezení 2: „cesta se nesmí skládat z oddělených cyklů“. To by, i při zachování předchozí podmínky, mohlo nastat například tak, jak je uvedeno na následujícím obrázku



Zajištění této podmínky je problematické a řeší se různě. My budeme postupovat následovně:

- *Kombinace*

Sestavíme všechny k-krokové neorientované cykly pro $k = 2, 3, \cdots, \left[\frac{n}{2}\right]$, kde $[\cdot]$ značí celou část čísla. Tedy např. pro 5 uzlů kontrolujeme jen pro $k = 2$.[1] Součet x_{ij} v těchto cyklech musí být $\leq k - 1$. Tyto cykly jsou určeny k-prvkovými podmnožinami uzlů, které odpovídají výběru k

[1] Obecně bychom měli kontrolovat do $n - 1$. Kdyby se v grafu vytvořil sub-cyklus délky větší nebo rovno $\left[\frac{n}{2}\right]$, musely by být zbylé uzly také v cyklu nebo několika cyklech - pro každý uzel požadujeme jednu vstupní a jednu výstupní hranu. Tyto cykly jsme již ale dříve kontrolovali. Podrobněji viz Dodatky 2.12.1

prvků z n-prvkové množiny, bez opakování, přičemž nezáleží na pořadí vybraných prvků - jejich počet tedy bude

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{n \cdot (n-1) \cdots (n-k+1)}{k \cdot (k-1) \cdots 2 \cdot 1}$$

pro každé k.

Pro $n = 5$ bude $k = 2, \cdots \left[\frac{5}{2}\right] = 2$, a tedy stačí kontrolovat pouze 2-krokové cykly.

2-krokové: 12, 13, 14, 15, 23, 24, 25, 34, 35, 45 (a vždy návrat do prvního, tedy 121, 131, $\cdots$ pro celé cykly). Je jich $\binom{5}{2} = 10$.

Návod na generování kombinací

- Začínáme od 1 a postupně zvyšujeme 12
- Pak zvětšujeme poslední až do n 13, 14, 15
- Pak zvýšíme předposlední, následuje o jedna větší 23
- A zase zvyšujeme poslední.
- A tak dál

Sestavení všech kombinací je ale jen pomocnou úlohou. Ty cykly, které je třeba blokovat jsou dány pomocí permutací v následujícím bodě.

• Permutace

V orientovaném grafu musíme vzít ještě v úvahu pořadí vybíraných uzlů - tedy vzít všechny dráhy mezi vybranou k-ticí uzlů. Přitom ale musíme vynechat všechny dráhy, které jsou po stejných hranách, jen jejich začátek je posunut. Tedy 123, 231,312 - ty tvoří jediný cyklus.

Postupujeme takto:

- v každé k-tici zafixujeme jeden uzel - třeba ten první,
- z ostatních vytváříme permutace.

Tak například pro trojici 123 fixujeme 1 a permutace 23 jsou 23, 32. Dostaneme tedy cykly 1231 a 1321.

Poznámka

Všimněme si, že pro tří-krokové cykly tento postup znamená, že vezmeme cyklus tam a ještě zpět: 1231, 1321. Pro více-krokové cykly je už situace složitější.

V generovaných permutacích jsou již původní kombinace obsaženy. Výsledné cykly, které je třeba kontrolovat, jsou tedy dány právě těmi cykly, které dostaneme při generování permutací. Celkový počet cyklů délky k (v n-uzlovém grafu) je dán počtem kombinací násobeném počtem $(k-1)$permutací

$$\binom{n}{k}(k-1)! = \frac{n!}{(n-k)!k!}\frac{k!}{k} = \frac{n!}{(n-k)!}\frac{1}{k} = \frac{V_k(n)}{k}$$

kde $V_k(n) = \frac{n!}{(n-k)!}$ jsou variace k-té třídy z n prvků.

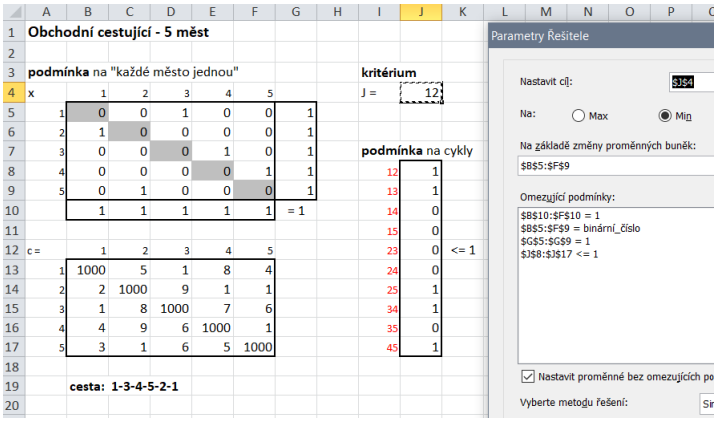
Pro náš příklad grafu s pěti uzly tedy výsledné cykly, které je třeba blokovat jsou:

2-krokové: 12, 13, 14, 15, 23, 24, 25, 34, 35, 45, počet: $\frac{V_2(5)}{2} = \frac{5 \cdot 4}{2} = 10$; (tady permutace odpadnou)

Na konci bude vždy ještě počáteční uzel, abychom cyklus uzavřeli.

Podrobný návod, jak obecně generovat cykly je uveden v Dodatcích 2.12.2 a 2.12.3

Excel: U19a_tspAsym.xlsx



Poznámka

V podmínkách na cykly musíme hlídat, aby nenastala situace, že se v grafu utvoří dva oddělené cykly. 1-krokové cykly nemohou nastat nikdy, protože smyčky v grafu jsou vyloučeny (prvky na diagonále incidenční matice se neuvažují). 2-krokové cykly tam a zpět jsou stejné - jen posunuté. 3-krokové cykly bychom museli hlídat v obou směrech - tedy „po směru“ i „proti směru“ - např. 1231 a ještě zpět 1321.

Protože ale 3-krokový cyklus muže být v kombinaci jedině s 2-krokovým (a ty jsou již hlídány) nemusíme je již v našem případě (5 měst) vůbec hlídat!

2.6.2 Symetrický TSP

V prvé řadě si musíme upřesnit, co znamená incidenční matice v neorientovaném grafu. Po každé hraně tohoto grafu můžeme jít ve směru ij (od uzlu i k uzlu j) a stejně tak i ve směru ji. Prvky ij a ji jsou transponované. Nad diagonálou (běžné orientované incidenční matice) jsou indikovány jedničkou hrany „tam” a transponovaně pod diagonálou hrany „zpět”. Když tuto matici překlopíme podél hlavní diagonály, obdržíme horní trojúhelníkovou matici, kde nuly označují neexistující hrany, jedničky označují hrany, po kterých se přešlo (nebo smí přejít) jen v jednom směru a dvojky hrany, po kterých se přešlo (nebo smí přejít) tam i zpět. Tuto matici můžeme nazvat incidenční maticí neorientovaného grafu a budeme s ní dále pracovat.

Platí

- Pro každý prvek incidenční matice na diagonále (to je to město) se sečtou prvky nad ním a vpravo od něho.

- Tento součet (počet cest ve trase, která jím prochází) musí být ≤ 2.

Sčítané prvky jsou na příkladě grafu s pěti uzly pro uzly 1 a 2 na následujícím obrázku

$$\begin{bmatrix} 1 & - & - & - & - \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix} \quad \begin{bmatrix} \cdot & | & \cdot & \cdot & \cdot \\ \cdot & 2 & - & - & - \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

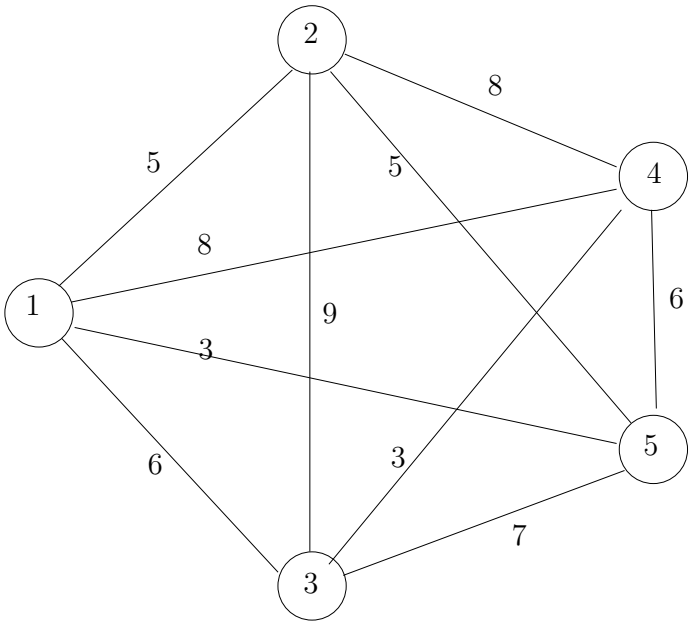
Budeme požadovat, aby její prvky byly binární - tj. ihned zakážeme cesty tam a zpět po jedné hraně.

Kontrola cyklů:

- 2-krokové cykly se nemusí kontrolovat, protože jsme vyloučili cestu tam a zpět po jedné hraně.
- k-krokové cykly, pro $k > 2$, stačí generovat jen jako kombinace (není potřeba pokračovat s permutacemi jako v orientovaném grafu).

Příklad

Řešte úlohu TSP pro následující graf



Řešení

Matice vzdáleností bude horní trojúhelník

$$c = \begin{bmatrix} 0 & 5 & 6 & 8 & 3 \\ - & 0 & 9 & 8 & 5 \\ - & - & 0 & 3 & 7 \\ - & - & - & 0 & 6 \\ - & - & - & - & 0 \end{bmatrix}$$

Stavovou matici zavedeme jako binární matici x rozměru 5×5 (5 uzlů) kde její prvek $x_{ij} = 1$ říká, že cesta povede po hraně ij (nebo ji, což je totéž) ale jen jednou. Tam a zpět by to bylo 2. $x_{ij} = 0$ znamená, že tudy cesta nevede. Tedy bude

$$x = \begin{bmatrix} \cdot & x_{12} & x_{13} & x_{14} & x_{15} \\ - & \cdot & x_{23} & x_{24} & x_{25} \\ - & - & \cdot & x_{34} & x_{35} \\ - & - & - & \cdot & x_{45} \\ - & - & - & - & \cdot \end{bmatrix}$$

kde x_{ij} jsou počítané veličiny, $-$ a $\cdot$ jsou neexistující hrany, přičemž $\cdot$ označují uzly 1, 2, 3, 4, 5.

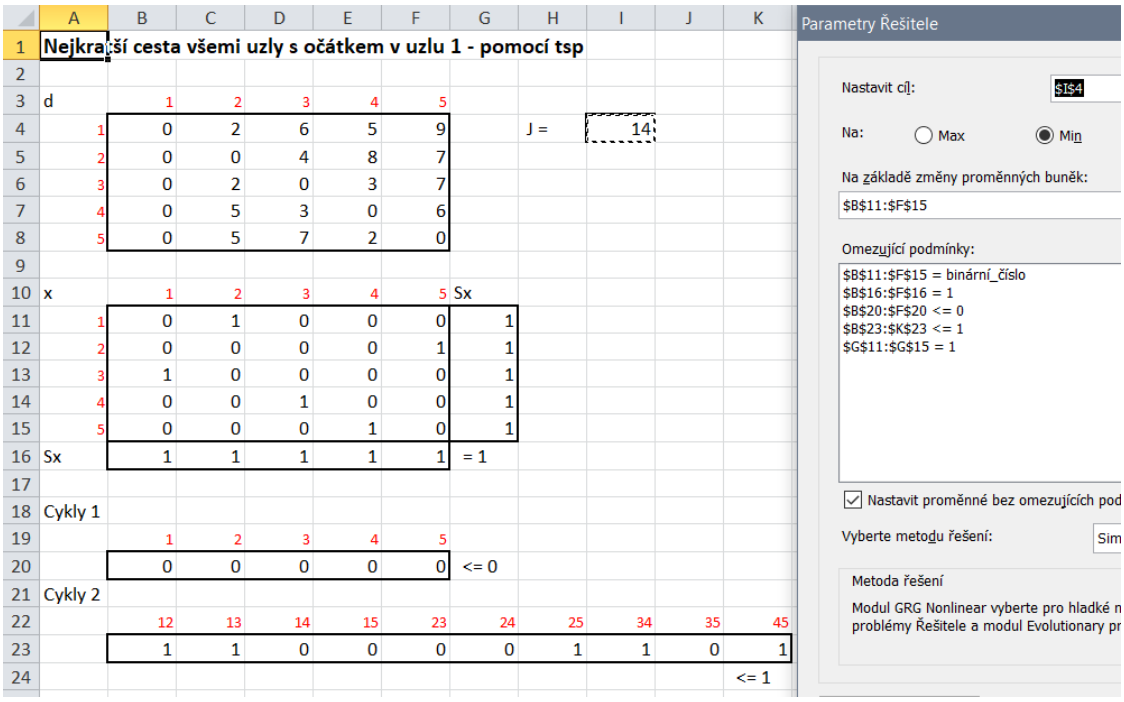
Nejprve definujeme podmínky pro maximálně 2 hrany v uzlu.

pro uzel 1: $x_{12} + x_{13} + x_{14} + x_{15} \leq 2$
pro uzel 2: $x_{12} + x_{23} + x_{24} + x_{25} \leq 2$
pro uzel 3: $x_{13} + x_{23} + x_{34} + x_{35} \leq 2$
pro uzel 4: $x_{14} + x_{24} + x_{34} + x_{45} \leq 2$
pro uzel 5: $x_{15} + x_{25} + x_{35} + x_{45} \leq 2$

Další podmínky by se týkaly vyloučení sub-cyklů. Protože ale náš graf je neorientovaný, jsou 2-krokové cykly automaticky vyloučeny. Takový cyklus by v matici x představoval dvojku a matice $x$je binární.

Výsledek je dole na obrázku v matici x. Vypíšeme si hrany podle jedniček. Je to 1-3, 1-5, 2-4, 2-5, 3-4. Nakreslíme si 5 uzlů a do nich vložíme nalezené hrany. To, co dostaneme je hledaná nejkratší cesta.

Program: U20b_tspSym.xlsx



	A	B	C	D	E	F	G	H	I	J	K
1	**Nejkra**ší cesta všemi uzly s počátkem v uzlu 1 - pomocí tsp										
2											
3	d	1	2	3	4	5					
4	1	0	2	6	5	9		J =	14		
5	2	0	0	4	8	7					
6	3	0	2	0	3	7					
7	4	0	5	3	0	6					
8	5	0	5	7	2	0					
9											
10	x	1	2	3	4	5	Sx				
11	1	0	1	0	0	0	1				
12	2	0	0	0	0	1	1				
13	3	1	0	0	0	0	1				
14	4	0	0	1	0	0	1				
15	5	0	0	0	1	0	1				
16	Sx	1	1	1	1	1	= 1				
17											
18	Cykly 1										
19		1	2	3	4	5					
20		0	0	0	0	0	<= 0				
21	Cykly 2										
22		12	13	14	15	23	24	25	34	35	45
23		1	1	0	0	0	0	1	1	0	1
24											<= 1

Úloha obchodního cestujícího má řadu modifikací.

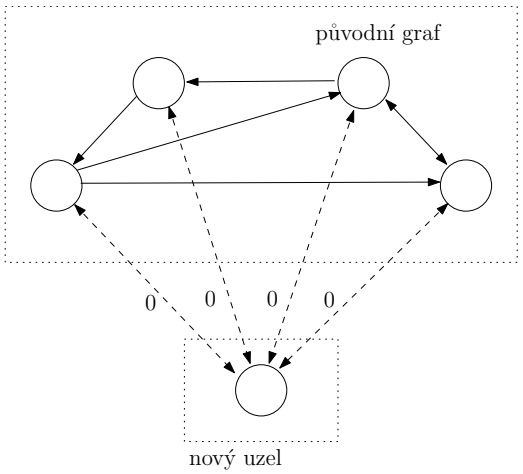
2.6.3 Vůbec nejkratší cesta grafem

Zadání úlohy :

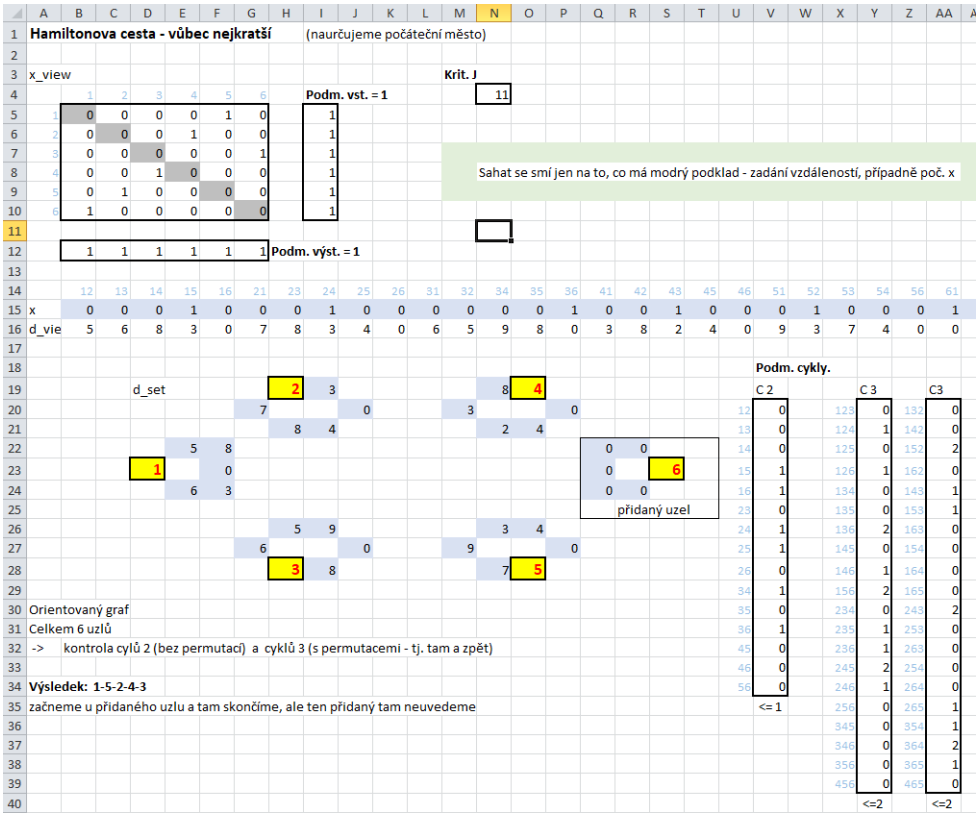
Vůbec nejkratší cesta všemi uzly bez ohledu na to, kde začíná a kde končí.

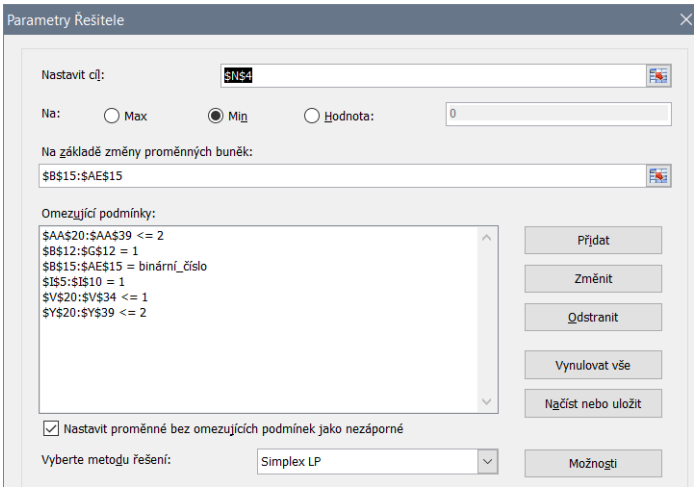
Řešení

Vezmeme graf a přidáme jeden uzel. Z něj vedeme cesty tam a zpět do všech uzlů grafu s nulovou délkou. Na tento rozšířený graf aplikujeme TSP a to, co vznikne v původním grafu je nejkratší cesta.



Excel: U20a_tspAplik1.xlsx





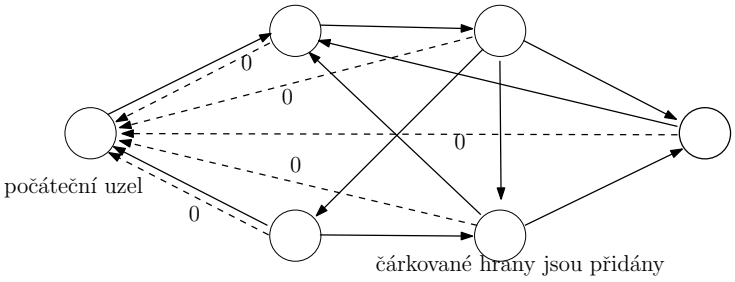
2.6.4 Nejkratší cesta grafem z daného uzlu

Zadání úlohy

Počáteční uzel cesty je dán. Hledáme nejkratší cestu z tohoto uzlu všemi ostatními uzly.

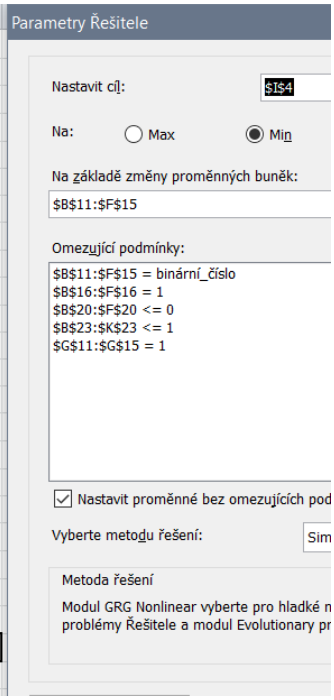
Řešení

Vezmeme počáteční uzel, a hranám, které do něj vedou dáme nulová ohodnocení. Na tento upravený graf aplikujeme TSP.



Excel: U20b_tspAplik2.xlsx

	A	B	C	D	E	F	G	H	I	J	K
1	**Nejkra**	**ší cesta všemi uzly s očátkem v uzlu 1 - pomocí tsp**									
2											
3	d	1	2	3	4	5					
4	1	0	2	6	5	9		J =	14		
5	2	0	0	4	8	7					
6	3	0	2	0	3	7					
7	4	0	5	3	0	6					
8	5	0	5	7	2	0					
9											
10	x	1	2	3	4	5	Sx				
11	1	0	1	0	0	0	1				
12	2	0	0	0	0	1	1				
13	3	1	0	0	0	0	1				
14	4	0	0	1	0	0	1				
15	5	0	0	0	1	0	1				
16	Sx	1	1	1	1	1	= 1				
17											
18	Cykly 1										
19		1	2	3	4	5					
20		0	0	0	0	0	<= 0				
21	Cykly 2										
22		12	13	14	15	23	24	25	34	35	45
23		1	1	0	0	0	0	1	1	0	1
24											<= 1



2.7 Propuknutí infekčního onemocnění

Obecná formulace úlohy

V n místech se vyskytlo infekční onemocnění. K dispozici je m lékařských týmů, které mohou onemocnění prošetřit. Tým $i \in \{1, 2, \cdots, m\}$ bude místo $j \in \{1, 2, \cdots, n\}$ vyšetřovat t_{ij} hodin. Každý tým může obsloužit 0, 1 nebo 2 místa. Jestliže má tým obsloužit ještě druhé místo (z místa k do místa l), musí se počítat s dobou přejezdu d_{kl}. Jakmile jsou všechna místa prošetřena, může se s infekcí začít bojovat. Cílem je navrhnout plán vyšetřování tak, aby cela akce byla co nejkratší.

Řešení

Definujeme veličinu $x_{ij} \in \{0,1\}$ s hodnotou 1 jestliže tým i bude vyšetřovat místo j a 0, když nebude.

Kriterium J

$$\mathcal{P}_i = \sum_j t_{ij} x_{ij}, \ \ \forall i$$

– doba na vyšetřování

$$\mathcal{Q}_i = \sum_{k,l;\, k \neq l} d_{kl} x_{ik} x_{il}, \ \ \forall i$$

– doba na přejezdy (jestliže bude tým i vyšetřovat obě místa k i l, bude $x_{ik}x_{il} = 1$, jinak je to nula)

$$J = \min \max_i \{\mathcal{P}_i + \mathcal{Q}_i\}$$

– hledáme, který tým i bude nejdelší (ostatní už budou hotovy) a tuto dobu chceme minimalizovat.

Omezení

$$\sum_i x_{ij} = 1, \ \ \forall j$$

– každé místo bude navštíveno právě jednou

$$\sum_j x_{ij} \leq 2, \ \ \forall i$$

– žádný tým nenavštíví více než dvě místa

$$x_{ij} \in \{0,1\}, \ \ \forall i,j$$

Tahle úloha ale nejde přímo realizovat (i) nelinearita v součinu, (ii) minimax. Musíme použít triky, zavedené v odstavcích 1.3.4 a 1.3.5.

Nelinearitu vyřešíme tak, že zavedeme novou proměnnou

$$w_{i;kl} = x_{ik} x_{il}$$

kde i je tým a kl přejezd z místa k do místa l. Index kl kódujeme takto $kl = 12 \to 1$, $kl = 13 \to 2$, $kl = 23 \to 3$ (je to podobné, jako při hledání cyklů v grafu viz 2.12.2 nebo pro jednosměrky 2.12.3)

Podmínky pro součin jsou (viz 1.3.4)

$$w_{i;kl} \leq x_{ik}, \ \ w_{i;kl} \geq 0, \ \ w_{i;kl} \leq x_{il}, \ \ w_{i;kl} \geq x_{ik} + x_{il} - 1.$$

Minimax vyžaduje definovat novou proměnnou q, která bude zároveň minimalizovaným kriteriem, a zavést podmínky

$$p_1 \leq q, \ \ p_2 \leq q, \cdots, p_n \leq q,$$

kde $p_1, p_2, \cdots, p_n$ jsou doby trvání práce jednotlivých týmů.

Realizace minimaxu je taková trochu tajemná:

- Zvolíme buňku pro kriterium J, nic tam nevkládáme a zadáme ji jako optimalizovanou.
- V listě Excelu vyjádříme doby trvání všech týmů: $J_1 = \mathcal{P}_1 + \mathcal{Q}_1$,$J_2 = \mathcal{P}_2 + \mathcal{Q}_2$ a případně další.
- Zadáme podmínky $J_1 \leq J$, $J_2 \leq J$ atd. nejlépe ve tvaru $J_1 - J \leq 0$, $J_2 - J \leq 0$ atd.
- Kritérium J budeme minimalizovat.

Příklad

2 týmy (i=1,2), 3 místa (j=1,2,3)

– stav (x_{ij} tým i vyšetřuje místo j)

$$x = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \end{bmatrix} \quad \text{binární}$$

– doby potřebné k vyšetření (tým i, místo j)

$$t = \begin{bmatrix} 5 & 3 & 4 \\ 4 & 2 & 5 \end{bmatrix}$$

– veličina w (cesta tam a zpět je stejně dlouhá - $kl = 12$ je totéž jako $kl = 21$)

$$w = \begin{bmatrix} w_{1;12} & w_{1;13} & w_{1;23} \\ w_{2;12} & w_{2;13} & w_{2;23} \end{bmatrix}$$

– délky přejezdu

$$d = [8,\ 4,\ 5]$$

Omezení (1., 2., 3. - součin; 4. minimax)

1. $w_{i;kl} \geq 0,\ \forall i, k, l$
2. $w_{i;kl} - x_{ik} \leq 0, \quad w_{i;kl} - x_{il} \leq 0,\ \forall i, k, l$
3. $w_{i;kl} - x_{ik} - x_{il} + 1 \geq 0$
4. $\sum_j t_{ij} x_{ij} + \sum_{kl;\, k \neq l} d_{kl} w_{i;kl} - q \leq 0,\ \forall i$

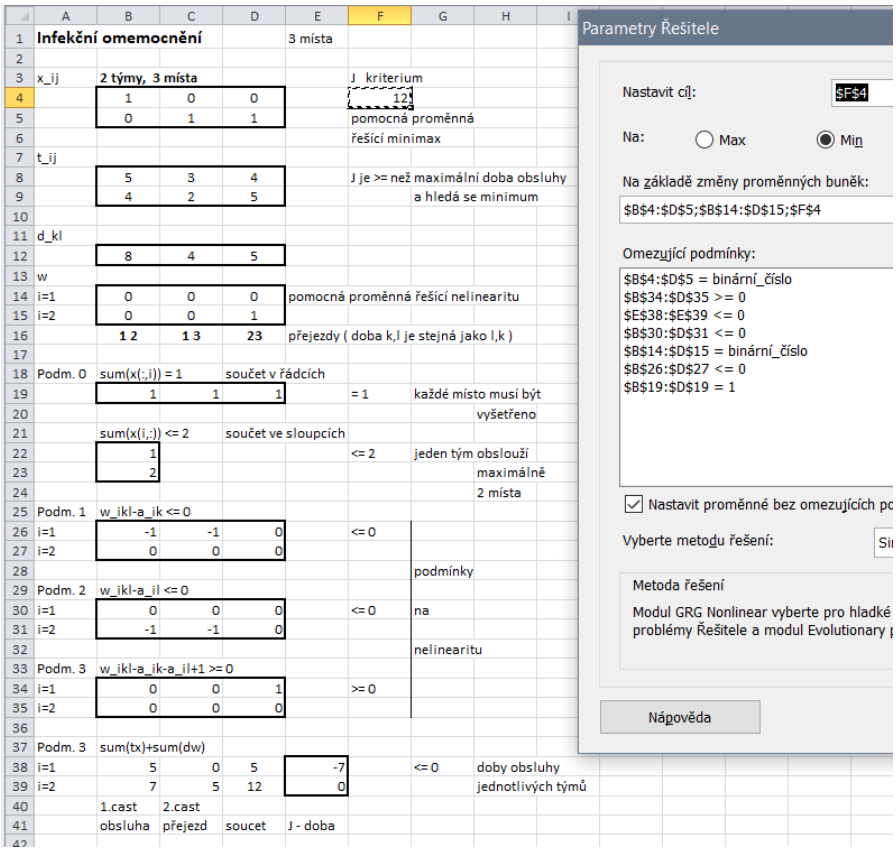
kde q je nová veličina s významem „doba trvání nejdelší akce".

Kriterium

$$J = q \to \min$$

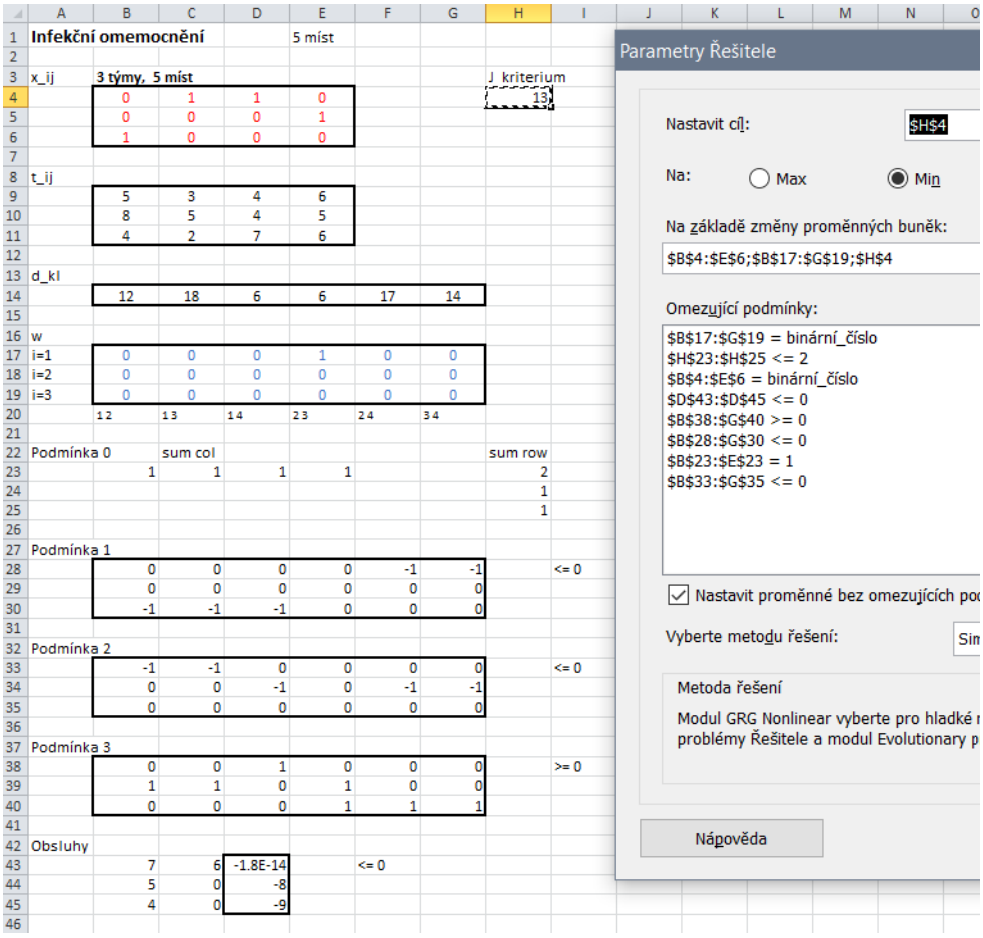
Nastavované veličiny jsou: $x \in \{0, 1\}$, $w \in \{0, 1\}$, $J = q \leq 0$

Excel: U21a_infectionSmall.xlsx,



	A	B	C	D	E	F	G	H
1	**Infekční omemocnění**				3 místa			
2								
3	x_ij	**2 týmy, 3 místa**				J kriterium		
4		1	0	0		121		
5		0	1	1		pomocná proměnná		
6						řešící minimax		
7	t_ij							
8		5	3	4		J je >= než maximální doba obsluhy		
9		4	2	5		a hledá se minimum		
10								
11	d_kl							
12		8	4	5				
13	w							
14	i=1	0	0	0	pomocná proměnná řešící nelinearitu			
15	i=2	0	0	1				
16		**1 2**	**1 3**	**23**	přejezdy (doba k,l je stejná jako l,k)			
17								
18	Podm. 0	sum(x(:,i)) = 1		součet v řádcích				
19		1	1	1		= 1	každé místo musí být	
20							vyšetřeno	
21		sum(x(i,:)) <= 2		součet ve sloupcích				
22		1				<= 2	jeden tým obslouží	
23		2					maximálně	
24							2 místa	
25	Podm. 1	w_ikl-a_ik <= 0						
26	i=1	-1	-1	0		<= 0		
27	i=2	0	0	0				
28							podmínky	
29	Podm. 2	w_ikl-a_il <= 0						
30	i=1	0	0	0		<= 0	na	
31	i=2	-1	-1	0				
32							nelinearitu	
33	Podm. 3	w_ikl-a_ik-a_il+1 >= 0						
34	i=1	0	0	1		>= 0		
35	i=2	0	0	0				
36								
37	Podm. 3	sum(tx)+sum(dw)						
38	i=1	5	0	5	-7	<= 0	doby obsluhy	
39	i=2	7	5	12	0		jednotlivých týmů	
40		1.cast	2.cast					
41		obsluha	přejezd	soucet	J - doba			

A příklad větších rozměrů je v U16b_ infectionBig.xlsx)



2.8 Dynamické plánování

(Production Planning)

Obecný návod

Zadáno: Požadavky P, ceny c, omezení L.

Optimalizuje se: x výroba (nákup), y indikátor pro $x > 0$, případně co zbude (g odpad).

Počítá se: z zásoba a musí se zadat $z \geq 0$.

Výpočet zásob: $z_{nové} = z_{staré} +$ výroba $-$ požadavek $(-$ odpad$)$

Podmínky:

(i) $x > 0 \to y = 1$, tj. `x-M*y<=0` indikátor výroby

(ii) minimální odběr `x-L*y>=0` (pro léky - L jsou min. odběry)

(iii) další omezení: maximální zásoby, na začátku a konci nulové zásoby atd.

Poznámka

Zásoby na konci roku určují platbu pro příští rok. Jsou-li roky označeny 1, 2, $\cdots$, pak začneme se z_0, tj. zásobami na konci nultého roku a natvrdo položíme nulové. Pro 3 roky tedy bude

rok	*0*	*1*	*2*	*3*
zásoba	z_0	z_1	z_2	z_3
hodnota z.	*0*	$z_1 = z_0 + x_1 - P_1$	$z_2 = z_1 + x_2 - P_2$	$z_3 = z_2 + x_3 - P_3$

kde z jsou zásoby, x je výroba a P jsou požadavky. Sloupec pro rok 0 je pomocný a umožní jednotný zápis pro zásoby (možnost kopírovat).

Abychom dosáhli nulové závěrečné zásoby, dáme podmínku $z_3 = 0$.

2.8.1 Plánování produkce

Pro budoucí časová období máme navrhnout objem produkce tak, aby celkové náklady a produkci a skladování byly minimální a aby požadavky na produkci v jednotlivých etapách byly splněny. Předpokládáme neomezenou kapacitu produkce v každém období. Předpokládáme dále, že

- náklady na produkci jsou úměrné její velikosti,
- náklady na skladování v každém období jsou úměrné úrovni zásob z konce předešlého období.

Rozhodujeme, zda vyrábět y_t a pokud ano, kolik vyrábět x_t, kde t je indikátor periody.

Zavedeme proměnné

c_t je jednotka nákladů na produkci, f_t je počáteční náklad v periodě t; h_t je jednotka nákladů na přechování uskladněných výrobků, z_t je velikost zásoby na konci periody t; d_t je požadavek na odběr zboží v periodě t, $t = 1, 2, \cdots, n$, (n je počet period).

Zápis standardní úlohy je následující:

Kriterium

$$J = \sum_{t=1}^{n} (c_t x_t + f_t y_t) + \sum_{t=1}^{n-1} h_t z_{t-1} \to \min$$

minimalizace nákladů na produkci a skladování.
Poznámka: *na začátku jsou zásoby 0 a na úplném konci se předpokládají také 0:* $z_0 = z_n = 0$. z_0 *se jen zadá nulové;* z_n *se počítá a musí se dát podmínka* $z_n = 0$.

Výpočet

$$z_t = z_{t-1} + x_t - d_t, \ t = 1, 2, \cdots, \text{počet period}$$

– kde zásoby na začátku první periody jsou $z_0 = 0$.

Omezení

$$x_t \leq M y_t$$

– y_t je indikátor nenulovosti x_t: $y_t = 1$ když $x > 0$, tj. když se vyrábí.

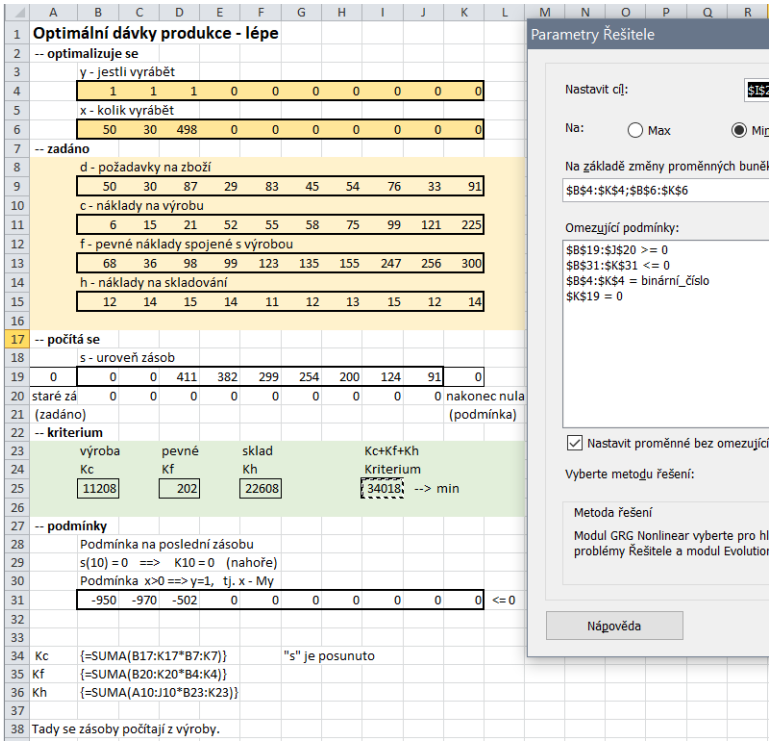
$$z \geq 0 \text{ a } z_n = 0$$

– zásoby jsou nezáporné a na konci jsou nula.

Optimalizované veličiny jsou

$$x_t \geq 0,\ y \in \{0, 1\}$$

Excel: U18a_dynProd.xlsx



Poznámka

1. *Při objednávkách je*

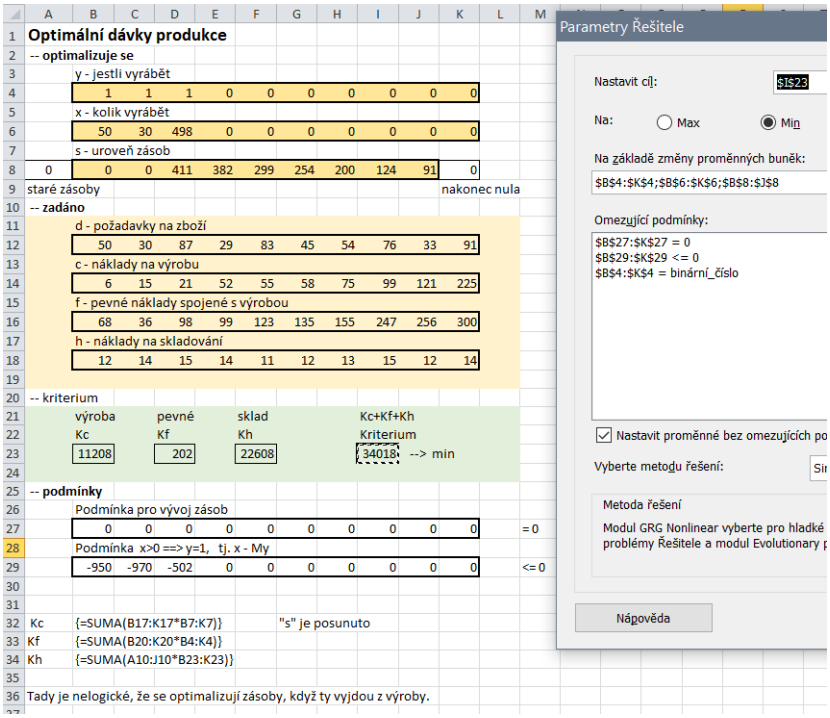
   $$z_t = z_{t-1} + x_t - d_t + b_t - b_{t-1}$$

   kde (asi) b jsou objednávky, které musíme splnit. Ty jdou mimo normální prodej a musí být připraveny v zásobách. Stará objednávka je dodána a proto se ze zásob odečte.

2. *Úlohu lze velice snadno rozšířit na případ, kdy výroba je v jednotlivých etapách omezena. Zavedeme veličinu u_t - omezení výroby v období t. Úloha je pak stejná, jen podmínku $x_t \leq My_t$ nahradíme podmínkou $x_t \leq u_t y_t$ s omezeními.*

3. *Stejně jednoduše můžeme respektovat omezené sklady. Pro omezení U zadáme $z \leq U$.*

Jiné možné řešení



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Optimální dávky produkce**												
2	-- **optimalizuje se**												
3		y - jestli vyrábět											
4		1	1	1	0	0	0	0	0	0	0		
5		x - kolik vyrábět											
6		50	30	498	0	0	0	0	0	0	0		
7		s - uroveň zásob											
8	0	0	0	411	382	299	254	200	124	91	0		
9	staré zásoby										nakonec nula		
10	-- **zadáno**												
11		d - požadavky na zboží											
12		50	30	87	29	83	45	54	76	33	91		
13		c - náklady na výrobu											
14		6	15	21	52	55	58	75	99	121	225		
15		f - pevné náklady spojené s výrobou											
16		68	36	98	99	123	135	155	247	256	300		
17		h - náklady na skladování											
18		12	14	15	14	11	12	13	15	12	14		
19													
20	-- kriterium												
21		výroba		pevné		sklad			Kc+Kf+Kh				
22		Kc		Kf		Kh			Kriterium				
23		11208		202		22608			34018	--> min			
24													
25	-- **podmínky**												
26		Podmínka pro vývoj zásob											
27		0	0	0	0	0	0	0	0	0	0	= 0	
28		Podmínka x>0 ==> y=1, tj. x - My											
29		-950	-970	-502	0	0	0	0	0	0	0	<= 0	
30													
31													
32	Kc	{=SUMA(B17:K17*B7:K7)}				"s" je posunuto							
33	Kf	{=SUMA(B20:K20*B4:K4)}											
34	Kh	{=SUMA(A10:J10*B23:K23)}											
35													
36	Tady je nelogické, že se optimalizují zásoby, když ty vyjdou z výroby.												

Tady se optimalizují také zásoby a jejich vývoj se uvažuje jen jako podmínka.

2.8.2 Objednávka léku

Obecné zadání úlohy

Lékařské středisko má rozhodnout o tom, jak objednat určitý lék od m potenciálních dodavatelů. Středisko má pro jednotlivé měsíce požadavky d_t, $t = 1, 2, \cdots, T$ a cena léku od dodavatele i v měsíci t je c_{it}. Každý dodavatel i má určitou minimální dávku odběru l_i. Nespotřebovaná balení léku je možno schovat do dalšího období jako zásobu z_t, která je však omezena velikostí Z. Zbylá balení léků (které nebyly spotřebovány a už se nevejdou do zásoby) se musí vyhodit. Cílem je minimalizovat náklady na pořízení léku v daném časovém úseku.

Řešení

Zavedeme proměnné:

c_{it} cena za jedno balení léku od dodavatele i v měsíci t

d_t požadavek na měsíc t

l_i minimální odběr od dodavatele i

x_{it} kolik balení léku bereme od dodavatele i v měsíc t

y_{it} jestli vůbec bereme (0 ne, 1 ano)

z_t zásoba v měsíci t (z minulého měsíce)

g_t kolik toho v měsíci t vyhodíme

První, co je třeba udělat, je vypočítat zásoby z_t.

$$z_z = z_{t-1} + \sum_i x_{it} - d_t - g_t,\ t = 1, 2, \cdots, \text{počet period}$$

To zadáme do Excelu jako počítané buňky.

Optimalizované veličiny jsou $x \geq 0$, int. $y \in \{0, 1\}$ a $g \geq 0$

Omezení:

$$x_{it} \geq l_i y_{it}$$

$\cdots$ když se nebere, tak se nebere; když ano, tak minimum od dodavatele i je l_i

$$x_{it} \leq 1000 y_{it}$$

$\cdots$ když se nebere, tak x_{it} musí být nula, jinak libovolné (nebo naopak: když se bere, tj. $x > 0$, pak bude indikátor $y = 1$).

$$x_{it} \in \{0, 1\}, \quad 0 \leq z_t \leq Z, \quad g_t \geq 0$$

$\cdots$ další podmínky.

Kriterium:

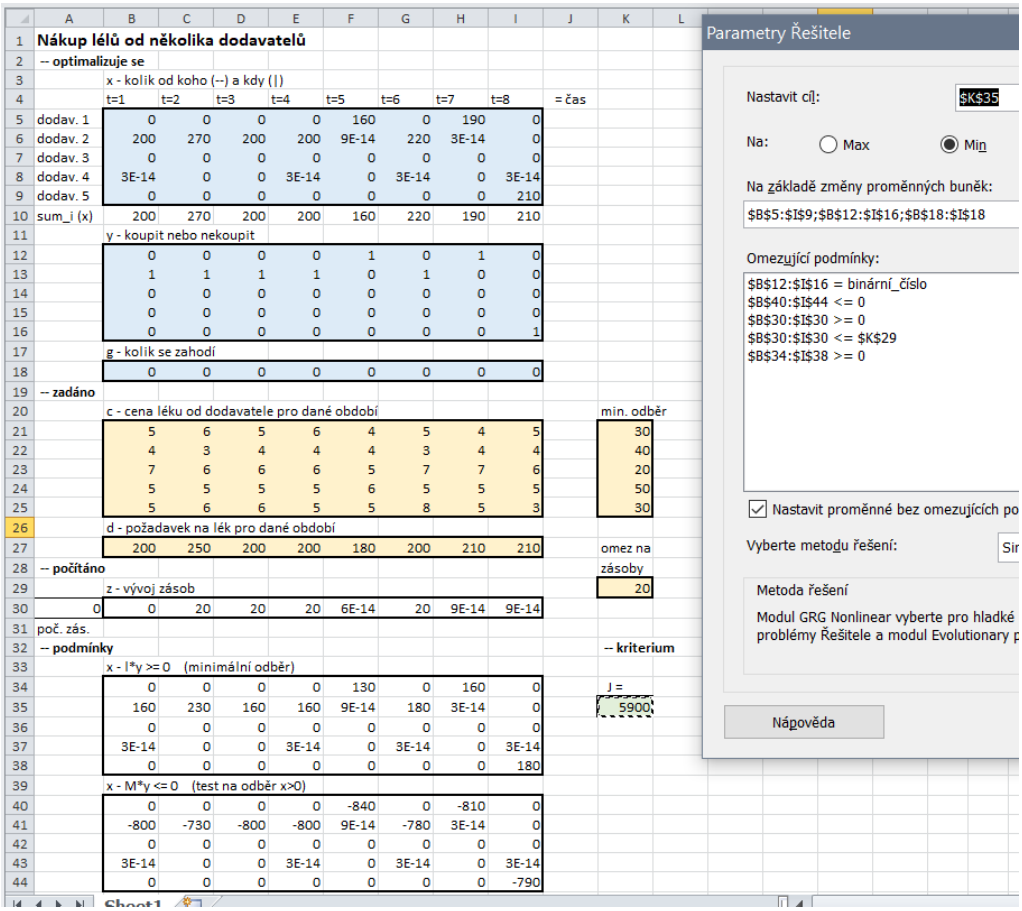
$$\sum c_{it} x_{it} \to \min$$

Příklad

Uvažujeme $m = 5$, $n = 8$,

$$c = \begin{bmatrix} 5 & 6 & 5 & 6 & 4 & 5 & 4 & 5 \\ 4 & 3 & 4 & 4 & 4 & 3 & 4 & 4 \\ 7 & 6 & 6 & 6 & 5 & 7 & 7 & 6 \\ 5 & 5 & 5 & 5 & 6 & 5 & 5 & 5 \\ 5 & 6 & 6 & 5 & 5 & 8 & 5 & 3 \end{bmatrix}, \quad l = \begin{bmatrix} 20 \\ 40 \\ 10 \\ 20 \\ 20 \end{bmatrix}$$

Excel: U18b_prodejLeku.xlsx



	A	B	C	D	E	F	G	H	I	J	K
1	**Nákup lélů od několika dodavatelů**										
2	-- optimalizuje se										
3		x - kolik od koho (--) a kdy (|)									
4		t=1	t=2	t=3	t=4	t=5	t=6	t=7	t=8	= čas	
5	dodav. 1	0	0	0	0	160	0	190	0		
6	dodav. 2	200	270	200	200	9E-14	220	3E-14	0		
7	dodav. 3	0	0	0	0	0	0	0	0		
8	dodav. 4	3E-14	0	0	3E-14	0	3E-14	0	3E-14		
9	dodav. 5	0	0	0	0	0	0	0	210		
10	sum_i (x)	200	270	200	200	160	220	190	210		
11		y - koupit nebo nekoupit									
12		0	0	0	0	1	0	1	0		
13		1	1	1	1	0	1	0	0		
14		0	0	0	0	0	0	0	0		
15		0	0	0	0	0	0	0	0		
16		0	0	0	0	0	0	0	1		
17		g - kolik se zahodí									
18		0	0	0	0	0	0	0	0		
19	-- zadáno										
20		c - cena léku od dodavatele pro dané období									min. odběr
21		5	6	5	6	4	5	4	5		30
22		4	3	4	4	4	3	4	4		40
23		7	6	6	6	5	7	7	6		20
24		5	5	5	5	6	5	5	5		50
25		5	6	6	5	5	8	5	3		30
26		d - požadavek na lék pro dané období									
27		200	250	200	200	180	200	210	210		omez na
28	-- počítáno										zásoby
29		z - vývoj zásob									20
30	0	0	20	20	20	6E-14	20	9E-14	9E-14		
31	poč. zás.										
32	-- podmínky										-- kriterium
33		x - l*y >= 0 (minimální odběr)									
34		0	0	0	0	130	0	160	0		J =
35		160	230	160	160	9E-14	180	3E-14	0		5900
36		0	0	0	0	0	0	0	0		
37		3E-14	0	0	3E-14	0	3E-14	0	3E-14		
38		0	0	0	0	0	0	0	180		
39		x - M*y <= 0 (test na odběr x>0)									
40		0	0	0	0	-840	0	-810	0		
41		-800	-730	-800	-800	9E-14	-780	3E-14	0		
42		0	0	0	0	0	0	0	0		
43		3E-14	0	0	3E-14	0	3E-14	0	3E-14		
44		0	0	0	0	0	0	0	-790		

2.9 Řazení úkolů

(Scheduling)

2.9.1 Obecná formulace

Základní pojmy jsou "úkol" a "pozice". Tedy máme n úkolů, které lze zpracovat jen postupně (nikoli paralelně). Zpracování i-tého úkolu trvá dobu p_i a má určeno, kdy má být úkol dokončen d_i. Jak máme zpracování úkolů uspořádat, aby součet zpoždění při dokončení jednotlivých úkolů (tardiness) t_i byl minimální.

Úloha tedy zní

Zadáno

$$p = [p_1, p_2, \cdots, p_n]$$
$$d = [d_1, d_2, \cdots, d_n]$$

minimalizujte celkové zpoždění

$$J = \sum_{i=1}^{n} t_i \to \min$$

kde t_i je definována jako rozdíl

$$t_i = (\text{skutečné dokončení ukolu } i\,) - (\text{požadované dokončení úkolu } i\,)$$

Poznámka

t_i označuje pouze zpoždění (je to nezáporná veličina). Pro předstih je nula.

2.9.2 Formulace s definicí "předcházení"

Definujeme binární veličinu y, tak, že $y_{ij} = 1$ znamená, že úkol i předchází úkol j. Předchází znamená, že je je někde před - nemusí být těsně před.

Poznámka

Z matice y stačí brát jen horní trojúhelník (bez diagonály), tedy y_{ij} pro $i < j$, protože platí $y_{ji} = 1 - y_{ij}$.

Zavedeme veličinu s_i - začátek zpracování úkolu i a dále $e_i = s_i + p_i$, což je skutečný čas dokončení i-tého úkolu (p_i je jeho trvání).

Formulace úlohy je následující:

Kriterium

$$J = \sum_{i=1}^{n} t_i \to \min$$

Omezení 1

$$e_i - d_i \leq t_i, \quad \forall i \ \text{(úkoly)} \tag{2.9.1}$$

nepřímo definuje zpoždění t_i.

Konec zpracování úkolu i, a tedy jeho odevzdání, je e_i. Ten se předpokládá větší, než požadovaná doba odevzdání d_i. Rozdíl $e_i - d_i$ je tedy kladný a představuje zpoždění při odevzdání úkolu i. Ten je shora omezen veličinou zpoždění t_i. Protože se ale součet všech t_i minimalizuje, bude, pro $t_i \geq 0$, zřejmě $e_i = d_i + t_i$. Dokončení úkolu je tedy požadované dokončení + zpoždění.

Navíc podmínka $t_i \geq 0$ se nedostane do rozporu s definicí (2.9.1) pro $e_i < d_i$ (předčasné splnění) jako by tomu bylo, kdybychom přímo požadovali rovnost $t_i = e_i - d_i$. V tomto případě bude $t_i = 0$.

Omezení 2,3

uspořádávají úkoly podle veličiny y_{ij}. Podle její hodnoty platí právě jedno z omezení

$$e_i \leq s_j + M\overbrace{(1 - y_{ij})}^{y_{ji}}, \quad \forall i < j$$

$$e_j \leq s_i + My_{ij}, \quad \forall i < j$$

První z omezení je pro $y_{ij} = 1$ aktivní, zatímco druhé bude vyřazeno (automaticky vždy splněno). Platí tedy

$$e_i \leq s_j$$

což znamená, že konec i-tého úkolu e_i bude před začátkem j-tého úkolu s_j. Pro $y_{ij} = 0$ to bude naopak. Aktivní je druhé omezení a to říká, že

$$e_j \leq s_i$$

a tedy, konec úkolu j bude před začátkem úkolu i.

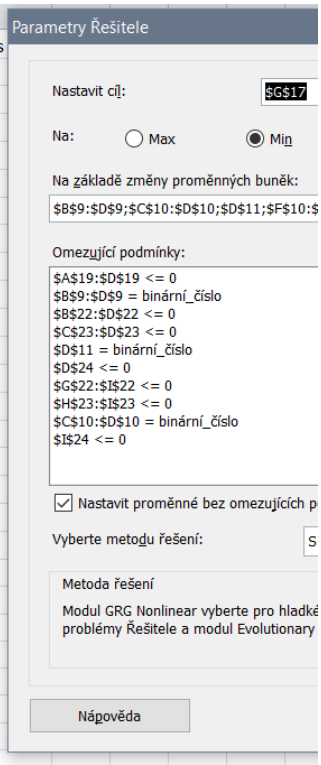
Výsledek se asi nejlépe přečte ze začátků úkolů s_i.

Poznámka

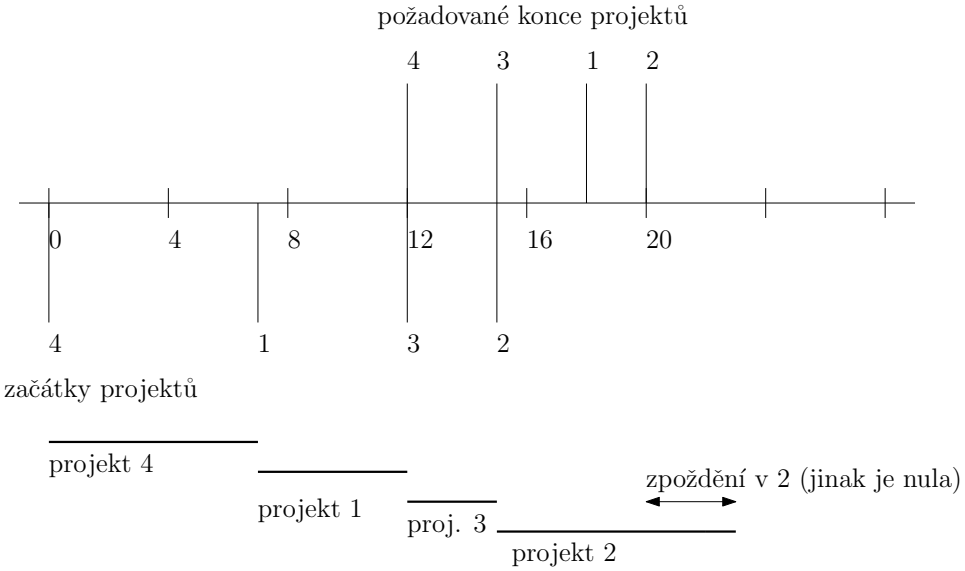
Je zajímavé, že často některé úkoly začínají stejně. Je to OK nebo je tam chyba?

Program je U23a_schedulingDJ.xlsx

	A	B	C	D	E	F	G	H	I
1	**Scheduling - tardiness problem**						disjunctive constraints		
2									
3	pi - processing times								
4	5	8	3	7					
5	di - due times					zadání			
6	18	20	15	12					
7									
8	yij - job j follows job i								
9		1	1	0		si - starting time of i			
10	0		0	0		7	15	12	0
11	0	1		0		ti - tardiness of i			
12	1	1	1			0	3	0	0
13	modré se optimalizuje					ei - ending time of i			
14	dol. troj. je yji = 1-yij					12	23	15	7
15									
16	**Podmínky**					**Kriterium**			
17						J =	3		
18	ei-di-ti								
19	-6	0	0	-5	<= 0				
20									
21	si+pi-sj-1000*(1-yij)					sj+pj-si-1000*yij			
22	1	-3	0	-988		1	-984	-992	0
23	2		-989	-977		2		0	-8
24	3			-985		3			-5
25	i / j	2	3	4		i / j	2	3	4
26			<= 0					<= 0	
27									
28	Výsledek řazení:						Takže řazení je:		
29	1. řádek y říká: 1. předchází 2. a 3.						4., 1., 3., 2.		
30	2. řádek: 2. nepředchází nic								
31	3. řádek: 3. předchází 2.								
32	4. řádek: 4. předchází 1., 2. a 3.								



Výsledek řazení je znázorněn na následujícím obrázku



Stručnější zápis programu v Excelu je U23a_strucne.xlsx

Úloha s osmi úkoly je řešena v U23a_pokus1.xlsx

2.10 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 $x \geq 0$.*

2.10.1 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	7	4
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 je, když vybereme nejnižší cenu a provedeme odpovídající přiřazení. Pak vezmeme další nejnižší cenu a pokud je to možné, přiřadíme. Tak dostaneme A→1 (1), C →3 (2), B → 2 (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 ↔ C: C → 1 (3), A → 3 (4), B → 2 (7). Celkem 14 - horší

A ↔ B: B → 1 (3), C → 3 (2), A → 2 (3). Celkem 8 - lepší.

2.10.2 Heuristiky obecné

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.

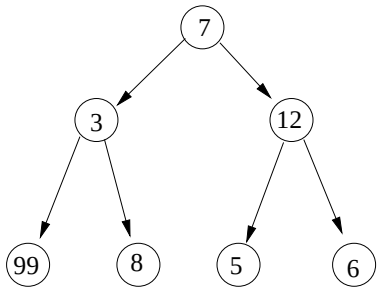
Greedy algoritmy mají následující části (*ilustrujeme na obchodním cestujícím*)

- 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,

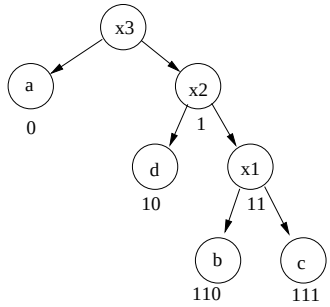
písmeno	$x1$	d	a
váha×100	31	34	35

   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.

Lagrangean relaxation

Může se stát, že řešíme LP problém, při kterém je několik omezení jednoduchých, a několik, které úlohu výpočetně komplikují. Potom můžeme použít metodu Lagrangean relaxation která hledá přibližné řešení přičemž aplikuje jen jednoduchá omezení. Druhá část omezení se přesouvá do kriteria podobně jako u metody lagrangeových multiplikátorů.

Tedy, řešíme problém

$$c'x \to \max$$
$$Ax \leq b$$

kde A je matice $m \times n$.

Jestliže matici A rozdělíme na A_1 $(m_1 \times n)$ a A_2 $(m_2 \times n)$, $m = m_1 + m_2$ a podobně i b, dostaneme

$$c^T x \to \max$$
$$A_1 x \leq b_1$$
$$A_2 x \leq b_2.$$

Problém je pak aproximativně zapsat takto

$$c^T x + \lambda^T (b_2 - A_2 x) \to \max$$
$$A_1 x \leq b_1$$

kde se požaduje $\lambda \geq 0$.

Připouští se, že podmínka $A_2 x \leq b_2$ není zcela splněna, ale její nesplnění se penalizuje. Naopak, čím "více" je splněna, tím dostaneme větší odměnu.

Pro optimální řešení $\hat{x}$ a řešení aproximované $\bar{x}$ platí

$$c^T \hat{x} \leq c^T \hat{x} + \lambda^T (b_2 - A_2 \hat{x}) \leq c^T \bar{x} + \lambda^T (b_2 - A_2 \bar{x})$$

protože

1. $c^T \hat{x} \leq c^T \hat{x} + \lambda^T (b_2 - A\hat{x})$ - λ je nezáporné a pro optimální řešení $\hat{x}$ platí $b_2 - A\hat{x} \geq 0$ (druhá podmínka),

2. $\lambda^T (b_2 - A_2\hat{x}) \leq c^T\bar{x} + \lambda^T (b_2 - A_2\bar{x})$ aproximované řešení $\bar{x}$ je optimální pro toto kriterium a první omezení je společné a optimální řešení je navíc ještě vázáno druhou soustavou omezení.

Z této nerovnosti plyne, že optimální řešení je nejblíže nejmenšímu řešení z aproximovaných. Při řešení postupujeme takto:

- Najdeme všechny přípustné hodnoty λ a k nim aproximovaná řešení $\bar{x}$.
- Jako optimální vezmeme nejmenší z nalezených aproximovaných řešení.

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

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

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.

2.11 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.

2.11.1 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 ebo 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

a program

2.11.2 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

2.11.3 Simulated annealing

Jedná se o metodu hledání globálního extrému u funkce s více lokálními extrémy. Metoda využívá metodu Monte Carlo a prohledává prostor řešení s postupně se zmenšujícími náhodnými odskoky od současného 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 z okolí x_{k-1}
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 $k \to \infty$)

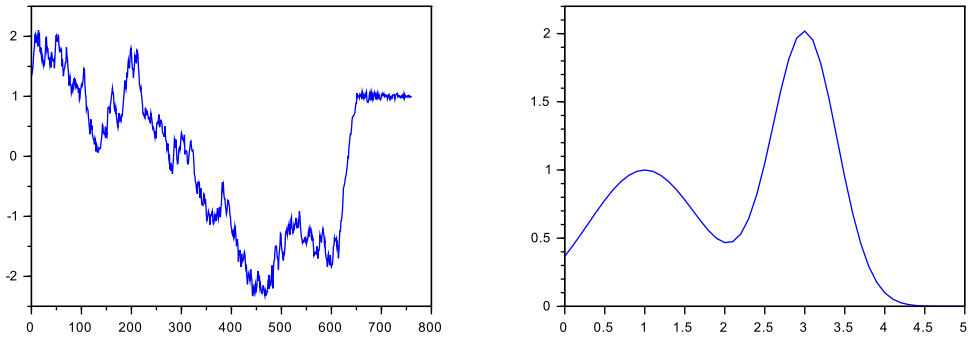
Hodnoty x_k by měly konvergovat ke globálnímu extrému.

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:

Výsledek je



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

2.11.4 Ant colony optimization

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.[2] Když projdou n městy (tj. vytvoří nějakou cestu pro obchodního cestujícího) 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

2.12 Dodatky

2.12.1 Kontrolované cykly v úloze TSP

V úloze TSP hledáme nejkratší cestu n městy. Základní podmínka je, že každý uzel grafu, popisující města a cesty mezi nimi, má jen jednu vstupní a jednu výstupní hranu (nebo 2

[2] O tvorbě této matice bude zmínka dále.

hrany v neorientovaném grafu). Tato podmínka ale nezaručí, že budou všechny města navzájem propojena. Této podmínce totiž vyhovují dva nebo více oddělených sub-cyklů. Tyto sub-cykly musíme eliminovat. Pro k-krokový sub-cyklus to provedeme podmínkou, že nesmí mít více než $k-1$ hran.

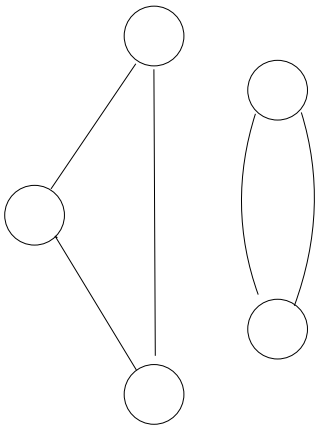
Které cykly je třeba hlídat?

1-krokové cykly jsou vyloučeny přímo zadáním stavové matice x jejíž prvky indikují použité hrany. Ta má vždy diagonálu rovnu 0.

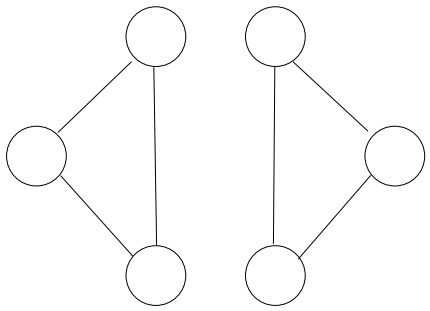
2-krokové cykly u symetrické úlohy (neorientovaný graf) také není třeba hlídat. Použité prvky matice x jsou definovány jak binární veličiny. V případě 2-krokového cyklu by tato matice měla některé prvky rovny dvěma

U asymetrické úlohy tyto cykly kontrolovat musíme.

Dále pak stačí cykly kontrolovat do stupně $\left[\frac{n}{2}\right]$,kde $[\cdot]$ značí celou část čísla. Lze totiž dokázat, že pokud se v grafu utvoří sub-cyklus, pak také ostatní uzly leží v cyklu nebo několika cyklech. Tedy pokud jsme již zkontrolovali cykly až do řádu $\left[\frac{n}{2}\right]$, vyšší řády už jsou vyloučené, protože by byly doplněny nižšími, které jsme ale vyloučili. Nejlépe je to vidět na příkladu grafu s pěti a šesti uzly



graf s pěti uzly



graf s šesti uzly

Pro pět uzlů je $\left[\frac{n}{2}\right] = \left[\frac{5}{2}\right] = 2$. A skutečně, sub-cyklus délky 3 už nemusíme kontrolovat, protože je doplněn sub-cyklem délky 2, který už byl kontrolován.

Pro 6 uzlů je $\left[\frac{n}{2}\right] = \left[\frac{6}{2}\right] = 3$ a vidíme, že tento cyklus je poslední, který je třeba kontrolovat.

To, jak cykly vyhledat, ukážeme v následujících odstavcích.

2.12.2 k-krokové cykly v neorientovaném grafu

Úkolem je vyjmenovat všechny k-krokové cykly v neorientovaném n-grafu. Tato úloha odpovídá výběru všech k-prvkových podmnožin uzlů z grafu o n uzlech, a tedy v určení všech kombinací k prvků z množiny o n prvcích.

Jak se dělají kombinace k z n?

Ukážeme pro 4 krokové cykly v pěti uzlech[3].

Začneme 1 2 3 4 a odzadu přičítáme jedna. Když to přeleze, uděláme přenos.

1 2 3 4
1 2 3 5
uděláme přenos přes 5
1 2 4 5
po přenosu pokračujeme ne od začátku, ale další vyšší číslicí
dále bude zase přenos, ale přes celé 4 5
1 3 4 5
a pokračujeme zase další vyšší číslicí (5)
a zase přenos přes 3 4 5
2 3 4 5
a hotovo

Dostali jsme 5 kombinací - $C_4(5) = \frac{5 \cdot 4 \cdot 3 \cdot 2}{4 \cdot 3 \cdot 2 \cdot 1} = 5$.

Poznámka: *Celý cyklus bude končit vždy stejným uzlem, jakým začínal. Proto např. 1 2 3 4 vlastně znamená 1 2 3 4 1. A stejně i pro ostatní.*

2.12.3 k-krokové cykly v orientovaném grafu

Úloha detekce cyklů v orientovaném grafu navazuje na předchozí úlohu o grafu neorientovaném. Základem jsou opět kombinace uzlů, ale tentokrát se cykly v jedné kombinaci mohou lišit ve směru hran. Např. kombinace 123 obsahuje 2 různé cykly: 1231 a 1321. Naopak, dráhy 1231, 2312 a 3123 tvoří jeden a tentýž cyklus, jen počátek se posouvá. Abychom určili všechny různé cykly a vyhnuli se těm, které jsou jen posunuté, budeme postupovat takto:

V každé kombinaci fixujeme jeden uzel a vypíšeme všechny permutace ostatních. Tyhle permutace by měly značit právě ty cykly, které v tomto případě hledáme.

Poznámka

Počet všech cyklů v k-grafu je dán počtem všech kombinací krát počet variací, dělený počtem posunutí

$$\binom{n}{k}\frac{k!}{k} = \frac{n!}{(n-k)!k!}\frac{k!}{k} = \frac{\frac{n!}{(n-k)!}}{k}\frac{V_k(n)}{k}$$

Jak se dělají permutace k?

Ukážeme pro graf se 4 uzly.

Permutace lze generovat rekurzivně. Označíme $P(a,b,c)$ jako permutace z a,b,c. Potom

$P(1,2,3,4) = [\{1, P(2,3,4)\}, \{2, P(1,3,4)\}, \{3, P(1,2,4)\}, \{4, P(2,3,4)\}]$ kde

$P(2,3,4) = [\{2, P(3,4)\}, \{3, P(2,4)\}, \{4, P(2,3)\}]$ atd. pro $P(1,3,4)$, $P(1,2,4)$ a $P(1,2,3)$.

$P(3,4) = [\{3,4\}, \{4,3\}]$ a stejně pro ostatní.

Konstrukce je tady

[3] Jak jsme právě ukázali, tento cyklus bychom v úloze TSP neřešili. Postup generování kombinací na něm ale ukázat lze, a bude docela krátký (což se nám hodí).

		12P(34)	1234	1243
	1P(234)	13P(24)	1324	1342
		14P(23)	1423	1432
		21P(34)	2134	2143
	2P(134)	23P(14)	2314	2341
		24P(13)	2413	1431
P(1234)				
		31P(24)	3124	3142
	3P(124)	32P(14)	3214	3241
		34P(12)	3412	3421
		41P(23)	4123	4132
	4P(123)	42P(13)	4213	4231
		43P(12)	4312	4321
			P(1234) = 4! = 24	

Příklad

Vyjmenujte všechny sub-cykly orientovaného grafu se 7 uzly.

Budeme kontrolovat cykly pro $k = 2, \cdots \left[\frac{7}{2}\right] = 2, 3$.

2-krokové cykly:

Kombinace - 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27, 34, 35, 36, 37, 45, 46, 47, 56, 57, 67. Je jich $\binom{7}{2} = 21$. Pozn. na konci bude vždy ještě počáteční uzel. Tedy 12 znamená 121. Atd.

Permutace - nic

3-krokové cykly:

Kombinace - 123, 124, 125, 126, 127, 134, 135, 136, 137, 145, 146, 147, 156, 157, 167, 234, 235, 236, 237, 245, 246, 247, 256, 257, 267, 345, 346, 347, 356, 357, 367, 456, 457, 467, 567. Je jich $\binom{7}{3} = 35$.

Permutace

– pro první kombinaci 123. Fixujeme uzel 1 a permutace zbylých je 23 a 32. Tedy dostaneme dva 3-krokové cykly 123 a 132.

– pro další kombinaci 124. Fixujeme 1 a dostaneme 124 a 142 atd.

V Excelu jsou dva programy pro graf s šesti uzly. První je U20c_tsp6_bad.xlsx a v něm se ukazuje, že když se nehlídají permutace (tady 3-krokové cykly tam i zpět), tak se skutečně v mohou objevit 2 tří-krokové cykly. To je špatně.

Stejný program, ale správně (i s permutacemi) je v programu U20d_tsp6_good.xlsx.

2.12.4 TSP jako nelineární problém

Zadání úlohy :

Obchodní cestující má navštívit n měst, každé z nich právě jednou a vrátit se do stejného města, ze kterého vyšel. Cena ca cestování má být co nejmenší. Ceny tras z města i do města j jsou dány jako prvky $c_{i,j}$ matice c.

Řešení

Zavedeme stavovou matici s prvky $x_{i,j} \in \{0,1\}$ (nepůjde, půjde z *i do j*) a úloha je zadána následovně

$$\sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}x_{i,j} \to \min$$

$$\sum_{i=1}^{n} x_{i,j} = 1, \ \ \forall j \ \ \text{(tj. sloupce)}$$

$$\sum_{j=1}^{m} x_{i,j} = 1, \ \ \forall i, \ \ \text{(tj. řádky)}$$

Podmínky na sub-cykly řešíme následujícím způsobem

$$\sum \text{diag}\,(X) = 0$$

$$\sum \text{diag}\left(X^2\right) = 0$$

$$\cdots$$

$$\sum \text{diag}\left(X^{n-1}\right) = 0$$

kde n je počet měst (uzlů) a X je čtvercová matice stavových proměnných $x_{i,j}$.

Poznámka

Místo podmínky $\sum diag\,(X) = 0$ by také měla jít veliká penalizace na diagonále matice c. S hodnotou 1000 to ale numericky selhávalo.

Excel: U20e_tsp10_Nonlin.xlsx

2.12.5 Nejkratší cesty mezi všemi uzly (nelineární)

Velice elegantní algoritmus, podobný algoritmu pro vyhledávání cyklů grafu, který ale podobně není lineární.

Máme čtvercové matice A a B s nezápornými prvky. Zavedeme operaci minimální sčítání minS(A,B) pomocí formule

$$\text{minS}(A,B)_{i,j} = \min_k \left(A_{ik} + B_{kj}\right), \forall i,j$$

Pro matici vzdáleností d orientovaného n-grafu zavedeme

$$d^{(2)} = \text{minS}\left(d,d\right)$$

je matice nejkratších drah tvořených dvěma hranami.

Podobně dále

$$d^{(3)} = \text{minS}\left(d,d^{(2)}\right), d^{(4)} = \text{minS}\left(d,d^{(3)}\right), \cdots$$

jsou matice nejkratších drah tvořených třemi, čtyřmi atd. hranami.

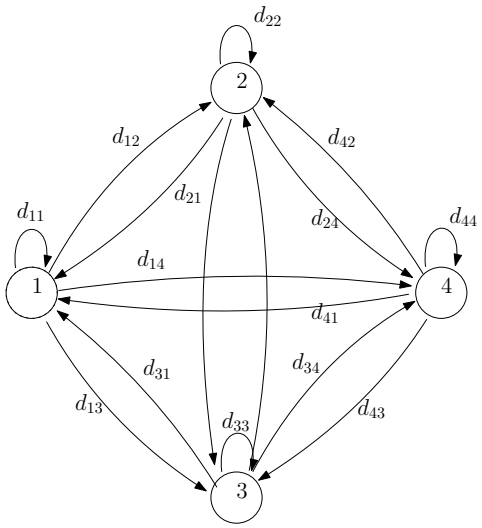
Nás bude zajímat matice všech nejkratších drah v grafu, tvořených daným počtem hran, z libovolného uzlu do jiného libovolného, tj. matice

$$d^{(k)} = \text{minS}\left(d,d^{(k-1)}\right)$$

kde $k = 1,2,\cdots n-1$ je počet uzlů v grafu.

Příklad

Uvedený algoritmus budeme demonstrovat na příkladě grafu se čtyřmi uzly



Matice délek hran bude

$$d = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{bmatrix}$$

Prvky této matice ukazují délky jedno-krokových drah mezi uzly. Tak např. d_{12} je délku cesty z uzlu 1 do uzlu 2 v jednom kroku.

Označíme $d^{(2)} = \min \mathrm{S}\{d, d\}$. Např jeho prvek $d_{12}^{(2)}$ bude

$$d_{12}^{(2)} = \min \left\{ [d_{11}, d_{12}, d_{13}, d_{14}] \oplus \begin{bmatrix} d_{12} \\ d_{22} \\ d_{32} \\ d_{42} \end{bmatrix} \right\} =$$

$$= \min\{d_{11} + d_{12}, d_{12} + d_{22}, d_{13} + d_{32}, d_{14} + d_{42}\}$$

kde operace $\oplus$ znamená sčítání odpovídajících prvků vektorů.

Členy součtu v minimu posledního výrazu představují délky všech dvou-krokových cest z uzlu 1 do uzlu 2:
– první: smyčka v 1 a přechod z 1 do 2,
– druhý: přechod z 1 do 2 a smyčka v 2,
– třetí: přechod 1 do 3 a z 3 do 2,
– čtvrtý: přechod z 1 do 4 a ze 4 do 2. A z nich se vezme minimální.

Prvky celé matice $d^{(2)}$ pak představují minimální dvou-krokové dráhy mezi příslušnými uzly.

Matice $d^{(3)} = \mathrm{minS}\left(d, d^{(2)}\right)$ je pak tvořena všemi nejkratšími tří-krokovými drahami mezi příslušnými uzly.

Delší dráhy už v grafu neexistují.

Vezmeme-li nyní matici D, jejíž libovolný prvek D_{ij} je

$$D_{ij} = \min\left\{d_{ij}, d_{ij}^{(2)}, d_{ij}^{(3)}\right\}$$

kde nuly jsme zaměnili na velké číslo, pak tato matice obsahuje délky minimálních drah z z uzlu i do uzlu j , $i, j = 1, 2, \cdots, n-1$ (tady $i, j = 1, 2, 3$) a to bez ohledu na počet hran, které ji tvoří.

Program ve Scilabu

2.12.6 TSP s opakovanými návštěvami měst

Zadání úlohy :

Úloha je dána jako asymetrický TSP, ale připouštíme více návštěv jednotlivých měst. Typická aplikační úloha je rozvoz zboží.

Řešení

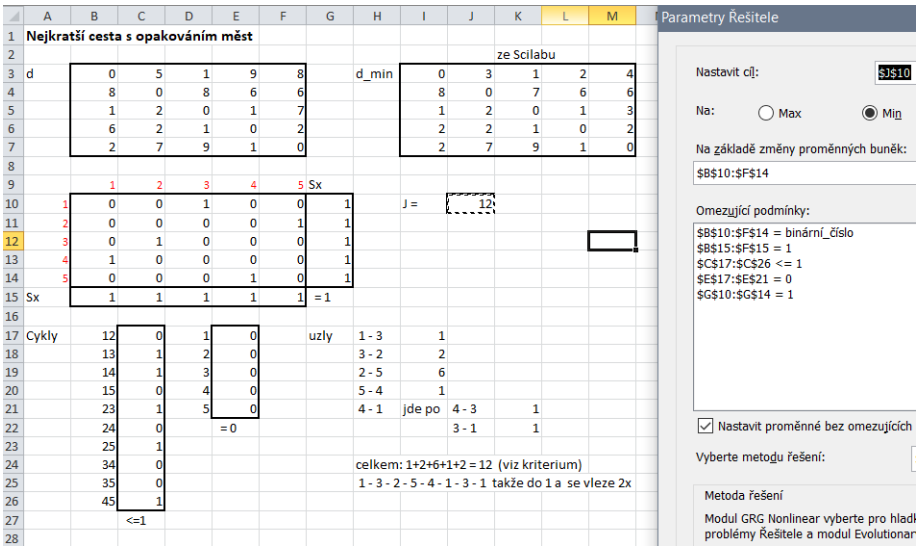
Místo matice vzdáleností jednotlivých dvojic uzlů (ohodnoceni hran grafu) zadáme matici nejkratších vzdáleností mezi jednotlivými dvojicemi uzlů. Může být: buď je vzdálenost spojovací hrany nejmenší, pak ji necháme. Nebo existuje kratší cesta, pak ji přepíšeme. Nebo hrana neexistuje, pak ji přidáme s ohodnocením nejkratší vzdálenosti mezi těmito uzly.

Poznámka

*Matice nejkratších vzdáleností se velice jednoduše spočte pomocí operace „***minimální sčítání matic***“, uvedené v předchozím odstavci.*

Na upravenou matici (nyní minimálních vzdáleností mezi uzly grafu) aplikujeme TSP. Dostaneme navazující dvojice uzlů, které v původním grafu vyznačíme. Každou hranu ale musíme prozkoumat. Pokud je délka hrany minimální délkou, pak ji na cestě ponecháme. Pokud není minimální, musíme hledat, po kterých hranách minimální cesta mezi aktuální dvojicí bodů vede a místo aktuální hrany vyznačíme tuto minimální cestu. Tím se může stát, že některým uzlem projdeme vícekrát, ale můžeme dostat kratší cestu než když trváme na podmínce jediného průchodu každým městem.

Excel: U21c_tspOpak.xlsx



Kapitola 3

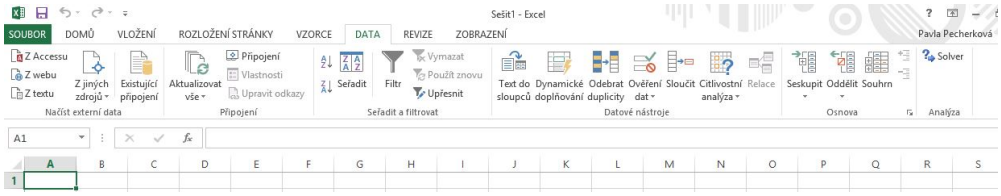
Programové vybavení

3.1 Excel

3.1.1 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 3.1.1.

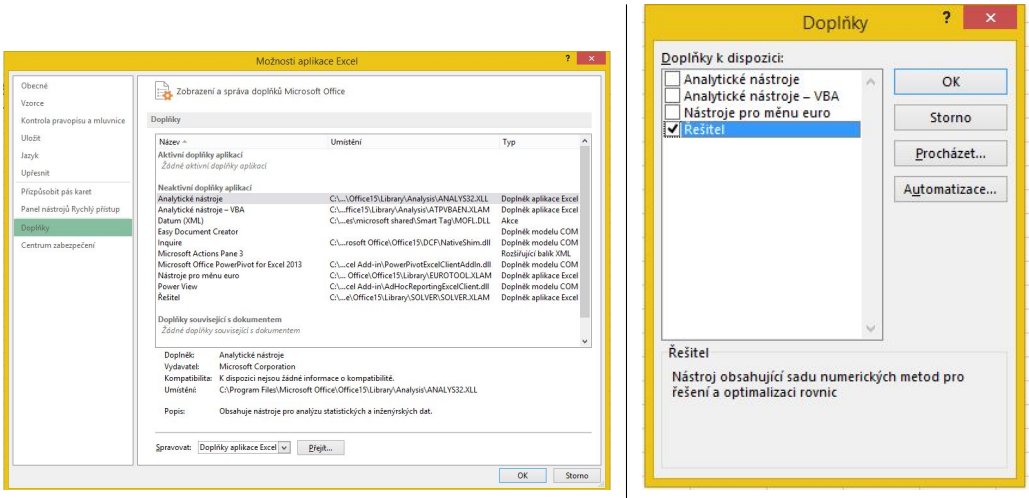


Obrázek 3.1.1: Excel - Data - Řešitel (Solver)

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

1. otevřete *"Soubor - Možnosti - Doplňky"* a otevře se vám nové okno, viz obrázek 3.1.2 (a),
2. v otevřeném okně dole klikneme na *"Spravovat: Doplňky aplikace Excel"* a otevře se další okno, viz obrázek 3.1.2 (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 3.1.2: Aktivace Řešitele

3.1.2 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.

3.1.3 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.)

3.1.4 Příklad v Excelu

Řešíme obecný příklad lineárního programování

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

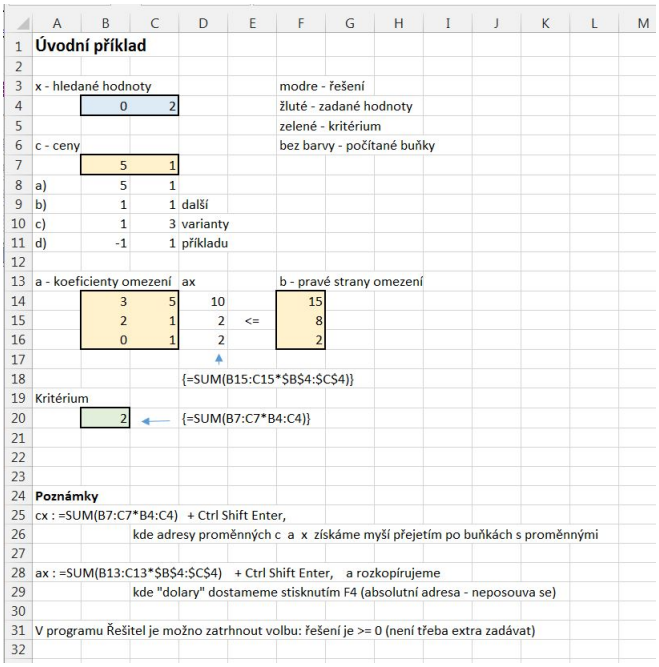
$$\begin{aligned} 3x_1 + 5x_2 &\leq 15 \\ 2x_1 + x_2 &\leq 8 \\ x_2 &\leq 2 \end{aligned}$$

$$x_1, x_2 \geq 0$$

Pro přehlednost, je hned na začátku uvedeno řešení, které je označeno modrým pozadím (obrázek 3.1.3). 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 3.1.3: Příklad v Excelu

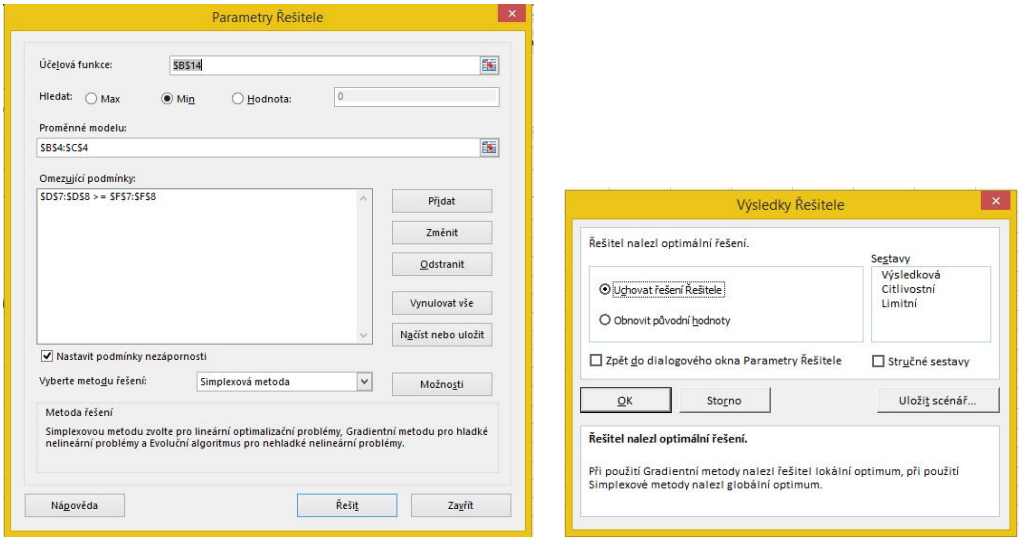
V případě, že již máme nastavené základní hodnoty, přecházíme k Řešiteli (Solver). Spustíme DATA-Řešitel (Solver) a objeví se nám tabulka znázorněna na obrázku 3.1.4 (a). Pro správnou funkci je potřeba nastavit následující:

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 3.1.4: Řešitel

Poté se již jen zadá funkce řešit a objeví se následující tabulka 3.1.4 (b). Tam necháme zaškrtnutou možnost **Uchovat řešení Řešitele** a potvrdit tlačítkem **OK**.

3.2 LiPS

LiPS - Linear Program Solver je free software, který provádí výpočet úloh lineárního (i celočíselného) programování. Navíc obsahuje také řešení úloh citlivosti a stability. Program není třeba instalovat, lze ho spustit přímo z .exe souboru, staženého na adrese

https://sourceforge.net/projects/lipside/

Tento software je doplňkový a budeme jej používat jednak jako ukázku simplexové metody, kterou ukazuje krok po kroku, jednak pro demonstraci citlivosti a stability úloh.

Poznámka

Podobný, poněkud silnější ale také složitější, je program LPSolve

https://sourceforge.net/projects/lpsolve/files/lpsolve/

Hodí se na rozměrné reálné úlohy.

Po spuštění programu LiPS se objeví aplikace s pracovní plochou. V ní je možno otevřít nové nebo již dříve uložené okno „model". To může mít dvojí podobu: *text* nebo *tabulka,* a to jak u nového, tak i u otevíraného okna (u otevíraného je typ okna možno vybrat v poli dole pod okénkem Název souboru).

Model se nejlépe zadá ve formě tabulky. Potom je možno model uložit a otevřít v textové podobě. Tím se zjistí, jaká jsou pravidla pro textové zadávání.

Řešení modelu se spustí ikonkou s bílým trojúhelníkem na zeleném poli. Po spuštění se objeví výsledkové okno „report", ve kterém je uložen celý postup řešení v simplexové tabulce. Výsledky jsou v posledních dvou tabulkách nadepsaných jako RESULTS.

První tabulka ukazuje řešení (Value), zadané ceny (Objective cost) a koeficienty stability (Reduced cost).

Druhá tabulka uvádí pravé strany omezení (RHS), rozdíly mezi levou a pravou stranou omezení (Slack) a koeficienty citlivosti (Dual price).

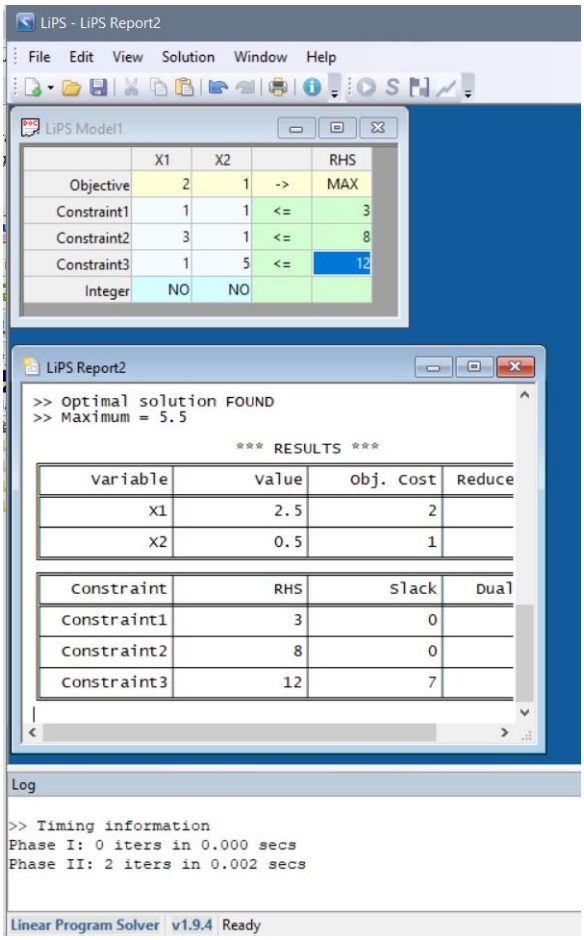
Po spuštění modelu a s kurzorem v okně modelu lze spustit další ikony vpravo od ikony Solve. Jsou to Sensitivity analysis, Matrix map a Solving history.

Po spuštění Sensitivity analysis (modré S) se objeví tabulka, kde vybereme RHS Range (All) a COST Range (All) a přesuneme je šipkou do pravého pole. Po OK se objeví citlivostní analýza.

Zbylé volby v předchozího menu provádí analýzu v situaci, kdy zvolíme jiné pravé strany nebo jiné koeficienty matice omezení A. Ty je třeba zadat tak, že se kurzorem postavíme na příslušný řádek v pravém okně a zvolíme dole Settings...

Zadaný model, stejně jako řešení nebo citlivost je možno uložit na disk.

Základní pohled na LiPS je v následujícím obrázku



3.3 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

Enter the linear programming problem here:

Maximize
Minimize
$z =$ 2x+y subject to the constraints:

Show only the region defined by the following contraints:

```
x+y<=3
3x+y<=8
x+5y<=12
```

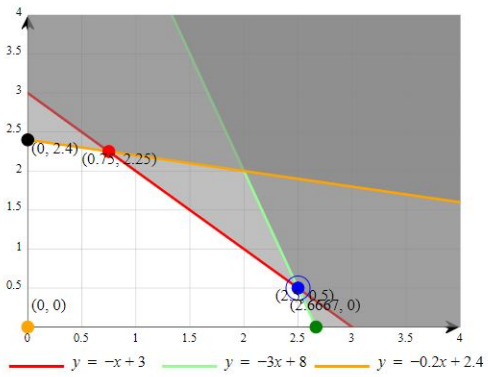
LP Examples | Graphing Examples | Solve

Rounding: 4 decimal places Fraction Mode

Erase Everything

The solution will appear below.

Vertex	Lines through vertex	Value of objective
(2.5, 0.5)	$x + y = 3$ $3x + y = 8$	5.5 Maximum
(0.75, 2.25)	$x + y = 3$ $x + 5y = 12$	3.75
(2.6667, 0)	$3x + y = 8$ $y = 0$	5.3333
(0, 2.4)	$x + 5y = 12$ $x = 0$	2.4
(0, 0)	$x = 0$ $y = 0$	0



Xmin: 0 Xmax: 4

Ymin: 0 Ymax: 4

Gridlines X: 0.5 Y: 0.5

Graph Show vertex coordinates

Kapitola 4

Sbírka příkladů

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

4.1 Set covering

4.1.1 Plánování směn - zaměstnanci

Zadání úlohy

Plánujeme obsazení 4 směn obsluhy v restauraci s rychlým občerstvením. Aby jednotlivé směny na sebe dobře navázaly, musí se na začátku a na konci překrývat. Proto celý den (24 hodin) rozdělíme na okna po 3 hodinách a směny sestavujeme podle následující tabulky (X označuje přítomnost příslušné směny v příslušném okně)

Okno	6-9	9-12	12-15	15-18	18-21	21-24	0-3	3-6	Plat
Směna 1	X	X	X						135
Směna 2			X	X	X				140
Směna 3					X	X	X		190
Směna 4	X						X	X	188
Požadavek	55	46	59	23	60	38	20	30	

Cílem je sestavit směny tak, aby celková vyplacená odměna byla co nejmenší

Poznámka

Směna bude mít určitý počet lidí tak, aby se v každém okně vyhovělo požadavku. Plat ve směně je pro všechny stejný a je dám dobou, kdy směna pracuje.

Řešení

Zavedeme: d_i - potřební pracovníci pro okno $i = 1, 2, \cdots, 8$

$$d = [55, 46, 59, 23, 60, 38, 20, 30]'$$

c_j - plat pro směnu $j = 1, 2, 3, 4$ (v rámci jedné směny je stejný plat),

$$c = [135, 140, 190, 188]$$

y_j - počet pracovníků ve směně $j = 1, 2, 3, 4,$ (optimalizovaný stav)

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

a matici a, transponovanou k uvedené tabulce s jedničkami na pozicích X a jinde nulami

$$A_{i,j} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

tedy řádky i odpovídají směnám, sloupce j oknům.

Zápis standardní úlohy je následující:

– kriterium (minimální celkové odměny)

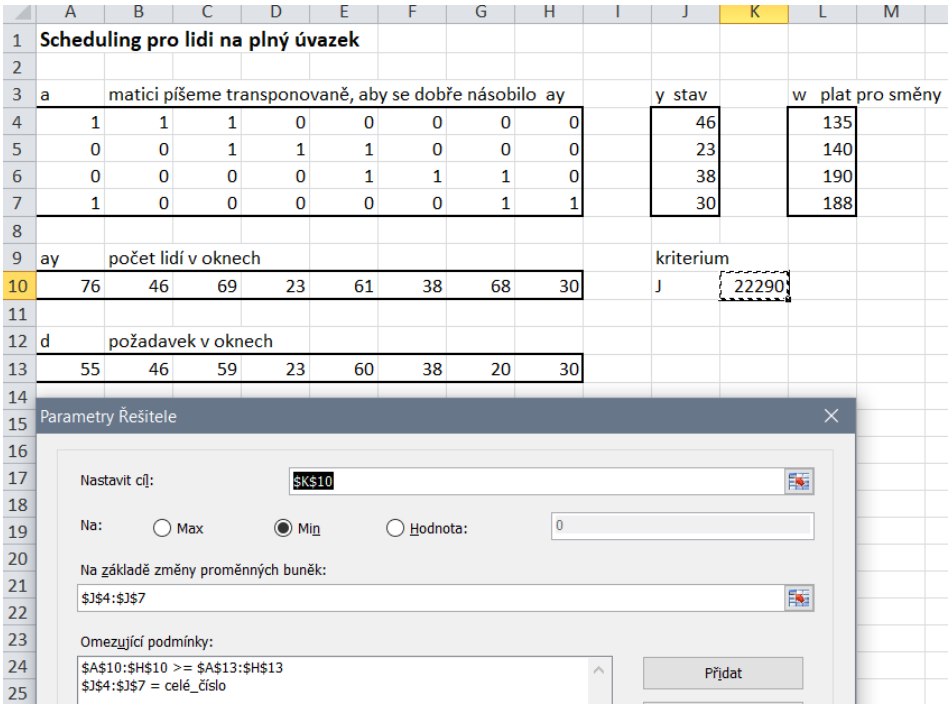
$$J = \sum_i c_i y_i \to \min$$

– podmínka (splnění požadavků na obsazení v oknech)

$$\sum_i y_i A_{i,j} \geq d_j$$

$$y_i \geq 0, \text{ int}$$

Excel: U22a_fullSchedul.xlsx



4.1.2 Plánování směn - zaměstnanci + brigádníci

Zadání úlohy

Zase jako před tím, ale najímáme i brigádníky. Ty přijímáme na divoko (do jednotlivých časových oken), platíme mzdou m_j a jejich počet označíme x_j pro každé okno j. Navíc musí platit, že brigádníci nesmí být ve směně sami. Vždy s mini musí být alespoň jeden zaměstnanec.

Řešení

Kriterium

$$J = \sum_i c_i y_i + \sum_j m_j x_j \to \min$$

kde $\sum_i c_i y_i$ je to, zaplatíme zaměstnancům a $\sum_j m_j x_j$ je plat brigádníkům.
Dále je vhodné vyjádřit počet zaměstnanců v oknech z

$$z_j = \sum_i y_i A_{i,j}$$

Podmínky

– uspokojení počtu pracovníků v okně j (zaměstnanci + brigádníci)

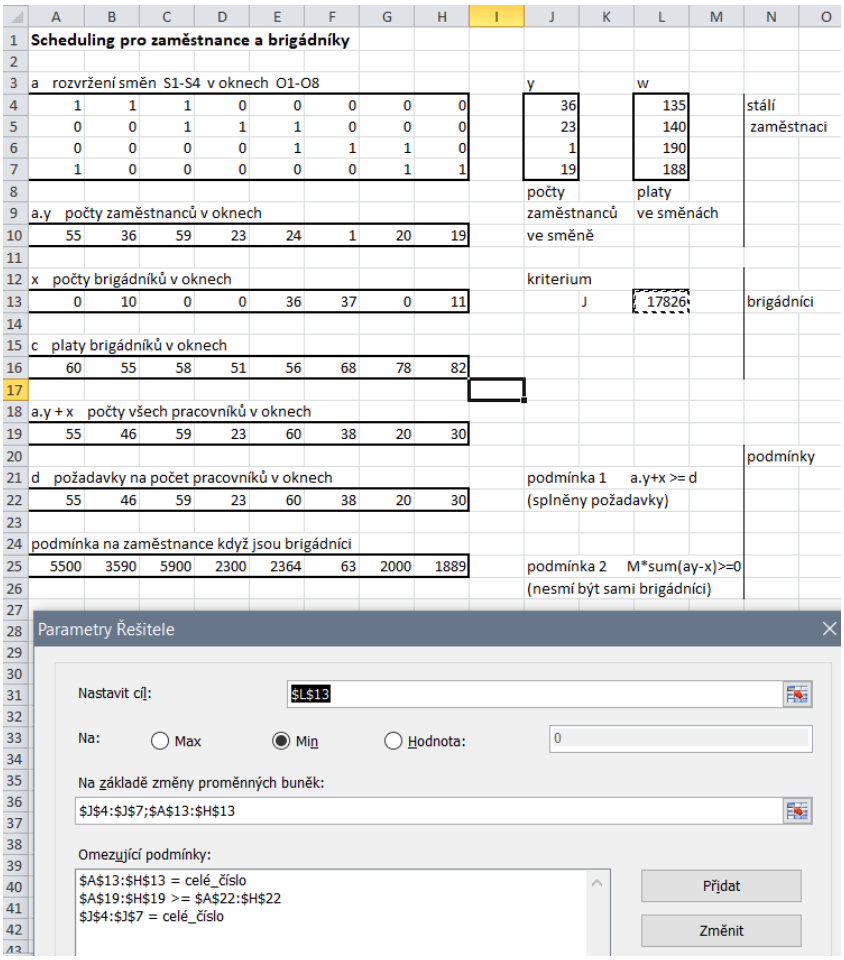
$$z_j + x_j \geq d_j$$

– brigádníci nesmí být sami[1]

$$x_j - Mz_j \leq 0$$

$$x_i, y_j \geq 0, \text{ int}$$

Excel: U22b_partSchedul.xlsx



4.2 Řazení úkolů

4.2.1 Formulace s definicí "pozice"

Definujeme binární veličinu x tak, že $x_{ij} = 1$ jestliže úkol i je na pozici j. Veličina zpoždění t_i je definována stejně - zpoždění při odevzdání úkolu i, tedy doba od požadovaného termínu odevzdání do skutečného odevzdání nebo nula, je-li úkol splněn včas (nebo s předstihem).

[1] Tohle je zajímavé. Je to jako indikátor nenuly, ale x může být, a bude, i větší než jedna. Není to tedy jen indikátor. Podmínka jen vylučuje, aby nastalo $x > 0$ a $z = 0$.

Kriterium

$$J = \sum_{i=1}^{n} t_i \to \max$$

Omezení 1 a 2

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall \text{pozice } j \tag{4.2.1}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall \text{úkoly } i \tag{4.2.2}$$

tj. na každé pozici musí být právě jeden úkol.

Příklad matice x pro konkrétní seřazení úkolů 1, 3, 2 bude

x_{ij}	1	2	3
1	1	0	0
2	0	0	1
3	0	1	0

Omezení 3

$$\sum_{i=1}^{n} p_i \underbrace{(x_{i1} + x_{i2} + \cdots + x_{ij})}_{=0 \text{ nebo } 1} - \sum_{i=1}^{n} d_i x_{ij} \leq t_j, \quad \forall \text{pozice } j \tag{4.2.3}$$

Tento výraz je poněkud nepřehledný, proto jej rozepíšeme podrobně (pro tři úkoly, tj. $n = 3$). Jedná se o n podmínek pro pozice $j = 1, 2, \cdots, n$ pro úkoly i

pozice $j = 1$

$$p_1 x_{11} + p_2 x_{21} + p_3 x_{31} - d_1 x_{11} - d_2 x_{21} - d_3 x_{31} \leq t_1$$

Poznámky
1. Celý tento výraz se týká první pozice.
2. Všechny x_{i1} jsou z prvního sloupce matice x. Vzhledem k podmínce (4.2.1) bude nenulové právě jedno z nich.
3. Nenulové x_{i1} bude označovat úkol na první pozici. Tedy doba jeho trvání bude znamenat také termín jeho ukončení (začínal v 0).
4. V druhé sumě nenulové x_{i1}vybere příslušný požadovaný konec úkolu.
5. Tím je dáno zpoždění na první pozici t_1

pozice $j = 2$

$$p_1 (x_{11} + x_{12}) + p_2 (x_{21} + x_{22}) + p_3 (x_{31} + x_{32}) -$$
$$-d_1 x_{12} - d_2 x_{22} - d_3 x_{32} \leq t_2$$

Poznámky
1. Pojednává se zde výhradně o druhé pozici.
2. Součty x_{ij} v kulatých závorkách se týkají postupně 1., 2., a 3. úkolu a prohledávají pozice před druhou pozicí (včetně).
3. Každá ze závorek může být buď nula (když příslušný úkol před pozicí 2 neleží) nebo jedna (když leží). Před pozicí 2 (včetně) budou ležet právě dva úkoly. Ty budou označeny jedničkami v x a jimi budou vybrány jejich délky. Jejich součet určí, kdy bude úkol na

pozici $j = 2$ skutečně ukončen.

4. Druhá suma opět vybere požadovaný konec úkolu na druhé pozici.
5. Tím je dáno zpoždění na druhé pozici t_2.

pozice $j = 3$

$$p_1 (x_{11} + x_{12} + x_{13}) + p_2 (x_{21} + x_{22} + x_{23}) + p_3 (x_{31} + x_{32} + x_{33}) - \\ -d_1 x_{13} - d_2 x_{23} - d_3 x_{33} \leq t_3$$

Poznámka
Pokračujeme stejně s třetí (poslední) pozicí. První suma detekuje konec úkolu na třetí pozici a druhá požadovaný konec tohoto úkolu. Tím je dáno zpoždění na třetí pozici t_3.

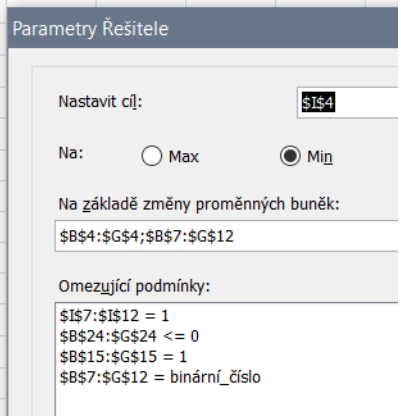
V Excelu úlohu realizujeme podle vzorce (4.2.3) ve tvaru

$$\sum_{i=1}^{n} \left(p_i \sum_{k=1}^{j} x_{ik} - d_i x_{ij} \right) \leq t_j, \quad \forall \text{pozice } j$$

tak, že vytvoříme matici $i \times j$ ve které realizujeme vnitřek kulaté závorky. Ta má indexy i a j, k je jen sčítací index. Tuto matici pak sečteme ve sloupcích. Součty budou mít indexy j. Součty pak porovnáme s t_j. Pozor! Index j je horní mez sčítání $\sum_{k=1}^{j}$, takže součty se postupně prodlužují. Zmíněnou matici lze vytvářet kopírováním. Zkonstruujeme první řádek a ve sloupcích kopírujeme. Ve vzorcích je třeba adresu pro t fixovat pomocí F4.

Excel: U23b_schedulingSP.xlsx

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Scheduling using positions**											
2												
3		t							J		p - proc.	d - due
4		0	0.16667	0.33333	0.5	3.83333	1.16667		6		8	7
5											5	19
6		x	2	3	4	5	6		Podm1		7	10
7	1	0	0	0	0	1	0		1		4	15
8	2	0	0	0	1	0	0		1		9	32
9	3	1	0	0	0	0	0		1	= 1	6	12
10	4	0	0	1	0	0	0		1			
11	5	0	0	0	0	0	1		1			
12	6	0	1	0	0	0	0		1			
13												
14		Podm2										
15		1	1	1	1	1	1	= 1				
16												
17	Podm3	k=1	k=2	k=3	k=4	k=5	k=6					
18	i=1	0	-0.1667	-0.3333	-0.5	-2.8333	6.83333					
19	i=2	0	-0.1667	-0.3333	-14.5	1.16667	3.83333					
20		-3	6.83333	6.66667	6.5	3.16667	5.83333					
21		0	-0.1667	-11.333	3.5	0.16667	2.83333					
22		0	-0.1667	-0.3333	-0.5	-3.8333	-24.167					
23	i=6	0	-6.1667	5.66667	5.5	2.16667	4.83333					
24	sum	-3	0	0	0	0	-7E-14	<= 0				
25												



Poznámka

Uvedené omezení (4.2.3) lze upravit takto

$$\sum_{k=1}^{j}\sum_{i=1}^{n} p_i x_{ik} - \sum_{i=1}^{n} d_i x_{ij} \leq t_j, \quad \forall \, pozice \; j,$$

kde $\sum_{i=1}^{n} p_i x_{ik}$ *a* $\sum_{i=1}^{n} d_i x_{ij}$ *jsou součiny vektoru* p *a matice* x*. Ten lze realizovat maticově. Tento tvar je tedy vhodný pro jednodušší implementaci úlohy v Excelu.*

4.2.2 Formulace jako "lineární třídění"

Definujeme binární x tak, že $x_{ij} = 1$ znamená, že úkol i leží někde před úkolem j. Dále zavedeme zpoždění t_j jako kladnou dobu od požadovaného do skutečného odevzdání úkolu. Předstih je nula.

Optimalizované veličiny jsou x a t.

Kriterium

$$J = \sum t_j \to \min$$

Podmínka 0

Místo toho, abychom v Řešiteli vynechávali diagonálu x, ponecháme x celé a na diagonále vynutíme nuly

$$x_{ii} = 0, \; \forall i$$

Podmínka 1

$$x_{ij} + x_{ji} = 1, \; \forall \, (i,j) \, , \; i > j$$

což znamená, že úkol i je buď před úkolem j nebo za ním (právě jedno z x_{ij} nebo x_{ji} musí být nula).

Podmínka 2

$$x_{ik} + x_{kj} + x_{ji} \leq 2, \; \forall \, (i,k,j) \; \text{různé}$$

Tato podmínka říká, že když je úkol i před úkolem k a úkol j před úkolem k pak musí být také úkol i před úkolem j (je to jakási analogie uspořádání: je-li $a < b$ a $b < c$, pak je $a < c$). Podmínka musí platit pro všechny trojice i, k, j včetně permutací.

Poznámka

V Příloze na straně 74 je popsáno jak trojice i, k, j *generovat.*

Podmínka 3 (definice zpoždění)

$$p_j + \sum_{i=1}^{n} p_i x_{ij} - d_j \leq t_j, \quad \forall \text{ úloly } j$$

kde $p_j + \sum_{i=1}^{n} p_i x_{ij}$ je okamžik dokončení j-tého úkolu a d_j je požadovaná doba. Rozdíl je tedy zpoždění v j-tém úkolu.

Vysvětlení: Sloupec matice x v sumě obsahuje jedničky tam, kde jsou úkoly i (řádky x) před úkolem j - j-tý sloupec. Každá z těchto jedniček se násobí příslušnou dobou trvání i-tého úkolu. Suma tedy představuje dobu, ve které může j-tý úkol začít. Po přičtení p_j dostaneme dobu ukončení j-tého úkolu.

V programu Excel úlohu realizujeme takto:

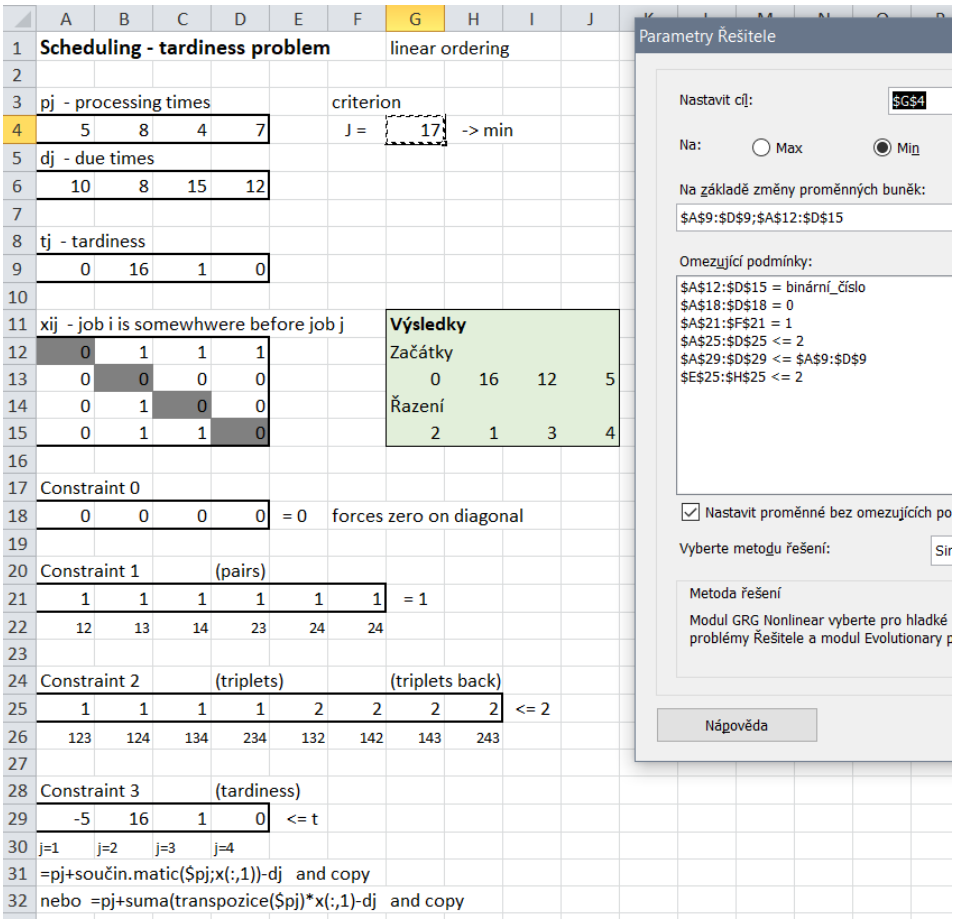
Podmínky 0, 1 a 2 se musí vypsat prvek po prvku. Doporučuje se, předepsat si kombinace indexů i, k, j. V podmínce 2 je třeba realizovat všechny trojice indexů, a to včetně pořadí :-(. Naštěstí se ale jedná o trojice, takže stačí vyjmenovat všechny kombinace a vzít je tam a zpět. Tedy např. tam 123, zpět 132, tam 124, zpět 142 atd.

Podmínku 3 lze kopírovat podle vzoru

pj+součin.matic($pj;x(:,1))-dj a kopie

Funkce součin.matic() je anglicky mmult().

Program je U23c_schedulingLO.xlsx



	A	B	C	D	E	F	G	H	I	J
1	**Scheduling - tardiness problem**						linear ordering			
2										
3	pj - processing times					criterion				
4	5	8	4	7		J =	17	-> min		
5	dj - due times									
6	10	8	15	12						
7										
8	tj - tardiness									
9	0	16	1	0						
10										
11	xij - job i is somewhwere before job j						**Výsledky**			
12	0	1	1	1			Začátky			
13	0	0	0	0			0	16	12	5
14	0	1	0	0			Řazení			
15	0	1	1	0			2	1	3	4
16										
17	Constraint 0									
18	0	0	0	0	= 0	forces zero on diagonal				
19										
20	Constraint 1			(pairs)						
21	1	1	1	1	1	1	= 1			
22	12	13	14	23	24	24				
23										
24	Constraint 2			(triplets)			(triplets back)			
25	1	1	1	1	2	2	2	2	<= 2	
26	123	124	134	234	132	142	143	243		
27										
28	Constraint 3			(tardiness)						
29	-5	16	1	0	<= t					
30	j=1	j=2	j=3	j=4						
31	=pj+součin.matic($pj;x(:,1))-dj and copy									
32	nebo =pj+suma(transpozice($pj)*x(:,1)-dj and copy									

Postačitelnost podmínek

V Podmínce 2 kontrolujeme vždy tři úkoly. Musíme vyloučit uspořádání, kdy první je následován druhým, druhý třetím a třetí zase prvním. V grafu se jedná o to, zda trojice uzlů (úkolů), a hran (následností) netvoří cyklus. Tady vznikají dvě otázky:

1. *Záleží při výběru uzlů na pořadí nebo ne?*
   Zkusíme tři uzly 1, 2, 3 (v tomto pořadí). Pro ně podmínka je

   $$x_{12} + x_{23} + x_{31} \leq 2$$

   Jestliže $x_{12} = 1$ a $x_{23} = 1$,pak musí být $x_{31} = 2$ (podmínka). Podobně, při cyklické záměně: Jestliže $x_{23} = 1$ a $x_{31} = 1$, pak musí být $x_{12} = 0$ - stejná podmínka. A stejně pro x_{31}, x_{12} a x_{23}.
   Ale! Když zaměníme pořadí, např. 1, 3, 2 (což není jen cyklická záměna), dostaneme $x_{13} = 1$, $x_{32} = 1$ a požadujeme, aby $x_{21} = 0$. To ale uvedenou podmínkou pokryto není a musíme formulovat podmínku novou

   $$x_{13} + x_{32} + x_{21} \leq 2.$$

   Taro podmínka pak vyhovuje i pro cyklickou záměnu uzlů. Vidíme tedy, že pro výběr platí pravidla pro tří-krokové cykly v asymetrickém TSP.

2. *Stačí kontrolovat vždy jen tři úkoly nebo se musí brát i více?*
   Zkusíme pro čtyři úkoly 1, 2, 3, 4. Tady by se požadovalo: Jestliže $x_{12} = 1$, $x_{23} = 1$ a $x_{34} = 1$, pak musí být $x_{41} = 0$. Tato čtveřice ale obsahuje čtyři trojice, které musíme kontrolovat. Jsou to 1,2,3 a 1,2,4 a 1,3,4 a 2,3,4. Z třetí trojice dostaneme $x_{12} = 1$, $x_{24} = 1$ $\rightarrow x_{41} = 0$. Protože ale $x_{12} = 1$ a $x_{23} = 1$ znamená, že úkoly jdou v pořadí 1, 2, 3, je $x_{13} = 1$. A tedy

   $$(x_{12} = 1 \,\mathrm{a}\, x_{23} = 1 \rightarrow x_{13} = 1) \,\mathrm{a}\, x_{34} = 1 \rightarrow x_{41} = 0.$$

   Stačí tedy kontrolovat vždy jen tři úkoly.

4.2.3 Hybridní formulace

V této formulaci se využívají proměnné obojího typu: (i) proměnné x definující pozici úkolu a (ii) proměnné y určující pořadí úkolů. Máme tedy

- $x_{ik} = 1$ jestliže úkol i je na pozici k,
- $y_{ij} = 1$ jestliže úkol i je naplánován někdy před úkolem j

Dále opět definujeme zpoždění t_i jako rozdíl mezi dobou dokončení úkolu i a jeho plánovaným dokončením.

Kriterium

$$J = \sum_i t_i \rightarrow \min$$

Podmínky 1 a 2

$$\sum_{i=1}^{n} x_{ik} = 1, \;\; \forall \,\text{pozice } k$$

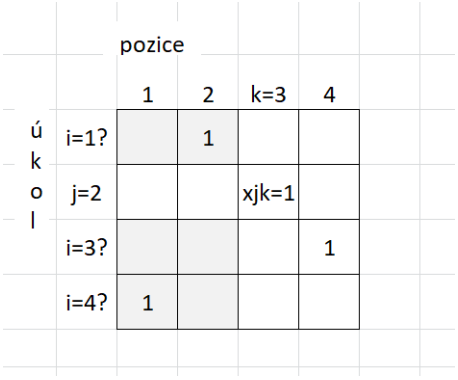
$$\sum_{k=1}^{n} x_{ik} = 1, \ \forall \text{úkoly } i$$

říkají, že na každé pozici musí být právě jeden úkol a každý úkol musí mít právě jednu pozici.

Podmínka 3

$$x_{jk} + \sum_{m=1}^{k-1} x_{im} \leq 1 + y_{ij}, \ \forall k > 1, \ j, \ i \neq j$$

Podmínku vysvětlíme na následujícím obrázku pro $n = 4$



Podmínka říká že, jestliže je $x_{jk} = 1$ (úkol j je na pozici k) a existuje nějaký úkol i na pozici menší než k (tedy je před ním), pak bude $\sum_{m=1}^{k-1} x_{im} = 1$. A tedy $x_{jk} + \sum_{m=1}^{k-1} x_{im} = 2$. Podmínka pak bude $2 \leq 1 + y_{ij}$,tedy $y_{ij} = 1$, což znamená, že úkol i bude ležet někde před úkolem j.

Podmínka bude na obrázku aktivní pro $i = 1$ a určí, že $y_{12} = 1$ a dále pro $i = 4$ a ta určí, že $y_{42} = 1$.

Podmínka 4

$$p_j + \sum_{i=1}^{n} p_i y_{ij} - d_j \leq t_j$$

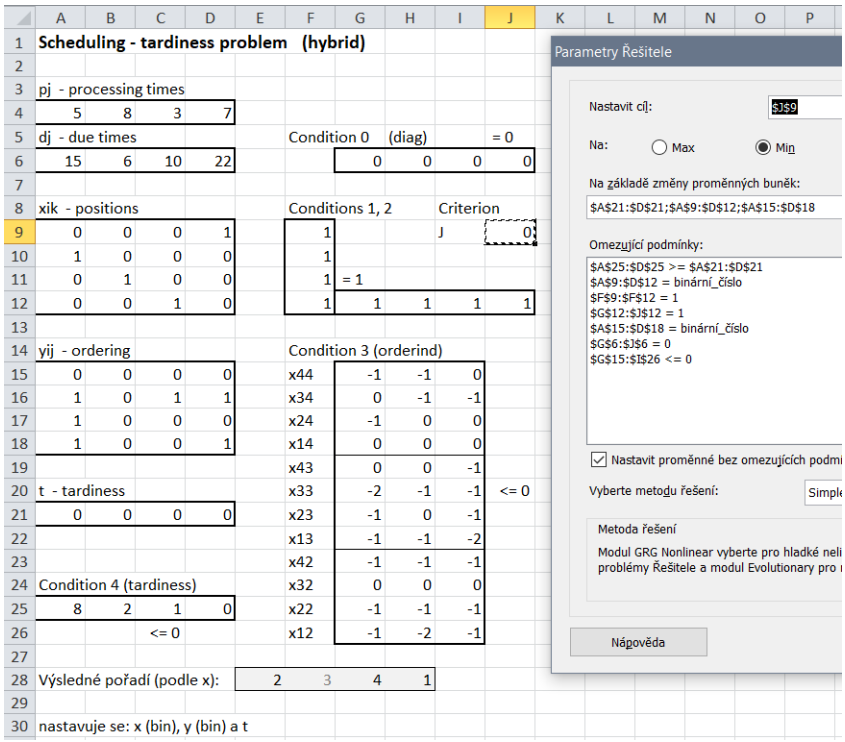
Tato podmínka určuje zpoždění t_j a je stejná jako ve formulaci s lineárním tříděním.

Realizace úlohy v programu Excel je dosti náročná. Je to proto, že podmínku 3 nelze vytvořit kopírováním a těchto podmínek je velmi mnoho.

Jinak, podobně jako v předchozí úloze, s maticí x lze pracovat vcelku, když zavedeme podmínku 0: diagonála$(x) = 0$. Podmínku 4 lze realizovat kopírováním - viz předchozí úloha.

Nastavují se veličiny x (binární), y (binární) a t.

Pro $n = 4$ je program U23d_schedulingHY.xlsx následující



	A	B	C	D	E	F	G	H	I	J
1	**Scheduling - tardiness problem (hybrid)**									
2										
3	pj - processing times									
4	5	8	3	7						
5	dj - due times					Condition 0	(diag)			= 0
6	15	6	10	22			0	0	0	0
7										
8	xik - positions					Conditions 1, 2			Criterion	
9	0	0	0	1		1			J	0
10	1	0	0	0		1				
11	0	1	0	0		1	= 1			
12	0	0	1	0		1	1	1	1	1
13										
14	yij - ordering					Condition 3 (orderind)				
15	0	0	0	0		x44	-1	-1	0	
16	1	0	1	1		x34	0	-1	-1	
17	1	0	0	0		x24	-1	0	0	
18	1	0	0	1		x14	0	0	0	
19						x43	0	0	-1	
20	t - tardiness					x33	-2	-1	-1	<= 0
21	0	0	0	0		x23	-1	0	-1	
22						x13	-1	-1	-2	
23						x42	-1	-1	-1	
24	Condition 4 (tardiness)					x32	0	0	0	
25	8	2	1	0		x22	-1	-1	-1	
26			<= 0			x12	-1	-2	-1	
27										
28	Výsledné pořadí (podle x):				2	3	4	1		
29										
30	nastavuje se: x (bin), y (bin) a t									

4.2.4 Formulace jako "cesty grafem"

Při této formulaci uvažujeme cesty grafem s incidenční maticí u_{ij}. Hledáme cesty s minimálním zpožděním $\sum_i t_i$. Protože chceme testovat všechny cesty, uvažujeme virtuální pozici 0 (viz triky v TSP). Úloha je postavena takto:

Definujeme

$$y_{ij} = 1, \text{ jestliže úkol } i \text{ je před úkolem } j \text{ (kdekoli)}$$

$$u_{ij} = 1, \text{ jestliže } i \text{ a } j \text{ jsou bezprostředně ua sebou}$$

$$t_i \text{ je zpoždění při odevzdání úkolu } i$$

Kriterium

$$J = \sum_i t_i \to \min$$

Podmínka 1 a 2

$$\sum_{i=1}^{n} u_{ij} = 1, \ \forall \text{úkoly } j \geq 0$$

tedy, každý předchozí má jediného následníka (první pozice je 0), a

$$\sum_{j=0}^{n} u_{ij} = 1, \ \forall \text{pozice } i \geq 1$$

tedy, každý následník má jediného předchozího.

Podmínka 3

$$\sum_{i=1}^{n} y_{ij} + \sum_{k=1}^{n} y_{jk} = n - 1, \ \forall \text{úkoly } j \geq 1$$

tedy, těch úkolů, co jsou před j, a těch úkolů, co jsou za j, je dohromady právě $n - 1$.

Podmínka 4

$$p_j + \sum_{i=1}^{n} p_i y_{ij} - d_j \leq t_j, \ \forall \text{úkoly } j \geq 1$$

což je definice zpoždění t_i.

Podmínka 5

$$\sum_{i=1}^{n} y_{ij} - \sum_{i=1}^{n} y_{ik} + (n+1)\, u_{kj} \leq n, \ \forall (j,k), \ j > 0, \ k > 0, \ j \neq k$$

která říká: Pokud hrana (i, j) leží na cestě uspořádání, pak úkol i má o jedna méně předchůdců než úkol j: $\sum_i y_{ij} + 1 = \sum_i y_{ik}$. Pokud neleží, tak podmínka je vždy platná (a tedy se neuvažuje).

Poznámka

Trochu lepší než podmínka 5 je tato

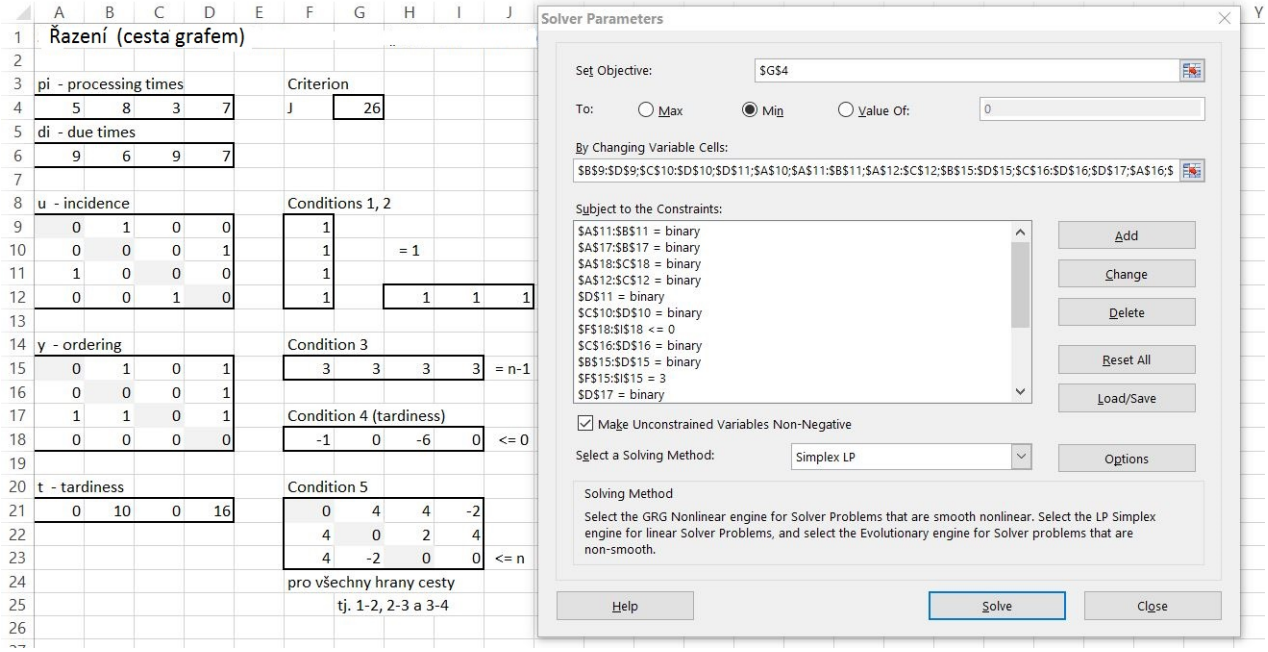
$$\sum_{i=1}^{n} y_{ij} - \sum_{i=1}^{n} y_{ik} + (n+1)\, u_{jk} + (n-1)\, u_{kj} \leq n, \ \forall (j,k), \ j > 0, \ k > 0, \ j \neq k$$

Řešení v programu EXCEL

je přímo úmorné! Diagonály matic u a y se musí zadat nulové, vyjmout z nastavovaných buněk a také z podmínky na binární veličiny.

Podmínka 5 se zadává do matice $(n-1) \times n$ - pro všechny hrany cest, jdoucích od vstupu do výstupu (cesta o délce 4 má 3 hrany).

Excel: U23e_schedulingTR.xlsx je tady



	A	B	C	D	E	F	G	H	I	J
1	Řazení (cesta grafem)									
2										
3	pi - processing times					Criterion				
4	5	8	3	7		J	26			
5	di - due times									
6	9	6	9	7						
7										
8	u - incidence					Conditions 1, 2				
9	0	1	0	0		1				
10	0	0	0	1		1		= 1		
11	1	0	0	0		1				
12	0	0	1	0		1		1	1	1
13										
14	y - ordering					Condition 3				
15	0	1	0	1		3	3	3	3	= n-1
16	0	0	0	1						
17	1	1	0	1		Condition 4 (tardiness)				
18	0	0	0	0		-1	0	-6	0	<= 0
19										
20	t - tardiness					Condition 5				
21	0	10	0	16		0	4	4	-2	
22						4	0	2	4	
23						4	-2	0	0	<= n
24						pro všechny hrany cesty				
25						tj. 1-2, 2-3 a 3-4				
26										