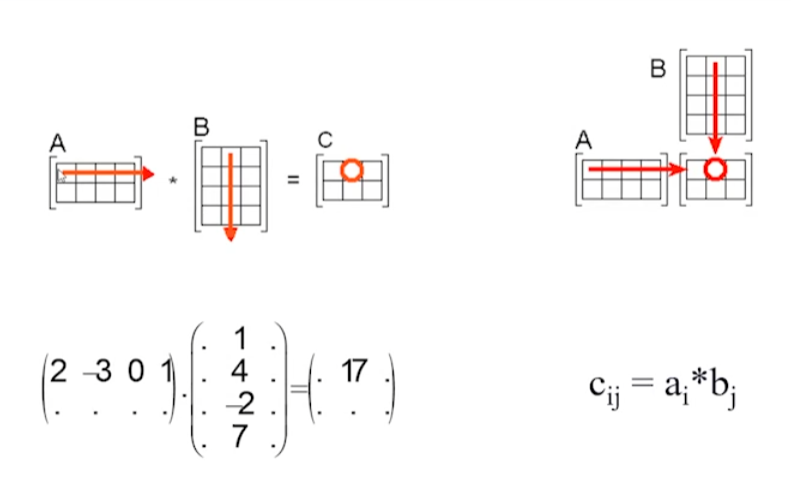
Lineární algebra

**Součin matic**

**Počet sloupců** první matice **se musí rovnat počtu řádků** druhé matice!

Násobím řádek u první matice se sloupcem v druhé matici

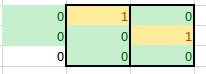
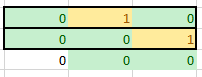
Výsledná matice má tyto rozměry:  
**Počet řádků první matice** X **Počet sloupců druhé matice**



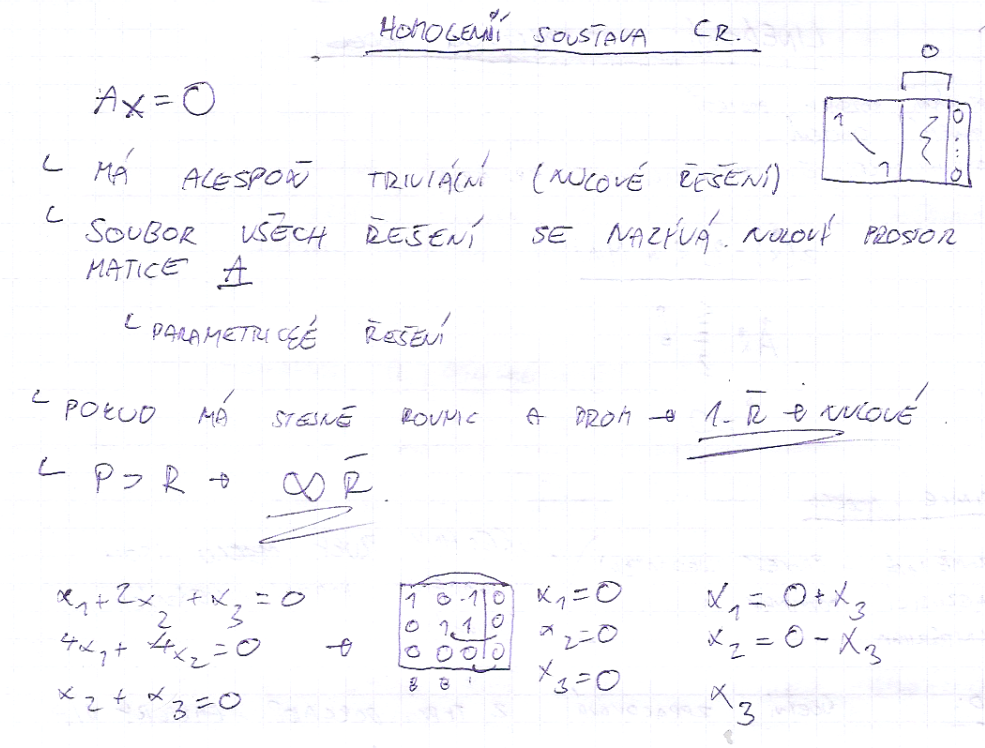
**Hodnost matice**

Provedu JEM a z té spočítám jednotkové vektory, například (1,0,0), (0,1,0)

například zde je hodnost 2



**Homogení soustava rovnic Ax=0**



Postup pro lineární model

Sestavení proměnných - x1,x2,x3

Typy proměnných:

* strukturní - hlavní rozhodovací proměnná
* nestrukturní - doplňková nebo pomocná proměnný

Omezující podmínky:

* Kapacitní - (Ax **<=** b) **Ax + Ed = b, d >= 0** Přidáváme doplňkové proměnné
* Požadavkové - (Ax **>=** b) **Ax - Ed + Ep = b, d >= 0** Přidáváme doplňkové proměnné, pomocné proměnné
* Doplnění nezáporných pomocných proměnných - kvůli konvergenci simplex. algoritmu
* Účelová fce
* určení, když je **=, +Ep**

**Doplňková proměnná** - překročení kapacity, cena je vždy 0

**Pomocná proměnná** - nedosažení kapacity, nesmí se dostat do báze! Tedy při MAX hodně záporná a při MIN hodně vysoká (alespoň o dvě cifry větší než ceny u UF)

| **Kanonický tvar** - tvar, kdy na levé straně je přítomna jednotková matice  **Samotná tabulka**   * báze - odejde proměnná z označeného řádku a přijde proměnná z označeného sloupce * zj-cj - **skalární součin vektoru aktuálního sloupce a bazického vektoru mínus cena** nad sloupcem * omega je jako **b děleno daný vybraný sloupec** * celkový výsledek - skalární součin vektoru b a báze   **Výběr kandidáta:**   * zj-cj: * pri **MAX** - vybírám nejvíce záporný prvek, cílem je mít všechny kladné (Hodnota MAX by se měla postupně zvětšovat) * pri **MIN**  - vybírám nejvíce kladný prvek, cílem je mít všechny záporné |
| --- |

* Při výběru podle omegy vyberu **nejmenší kladné číslo (ne NULU)**

**Alternativní řešení (struktura)** - pokud je v řádku **0 u nebazické proměnné** (Lze přidat do báze - nekonečně mnoho řešení)

**Hodnost matice** - počet lineárně nez. členů

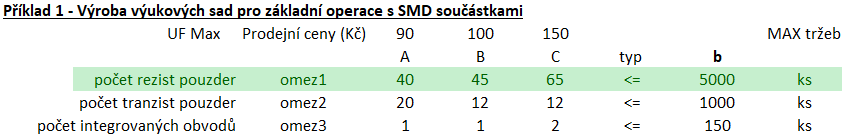
**Suboptimální řešení**

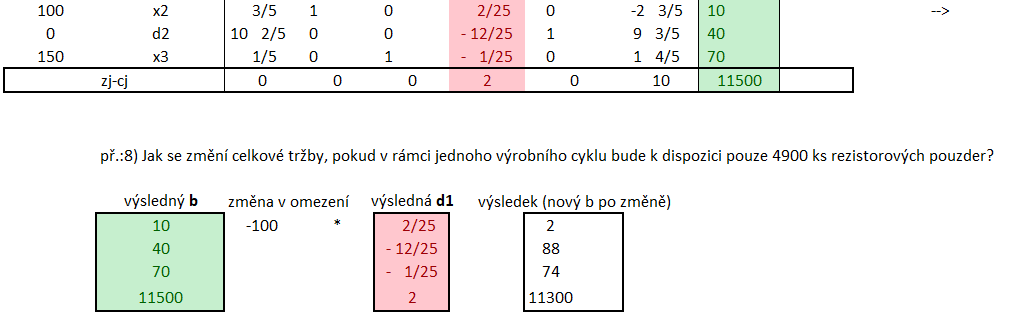
Příklad: pokud se změní počet omezení z 5000 na 4900 => tak se cena změní o 100\*(zj-cj) z posledního optimálního kroku

**“přípustné řešení”**, ve vektoru pravých stran(**b**) není záporné číslo a v bázi není **p**

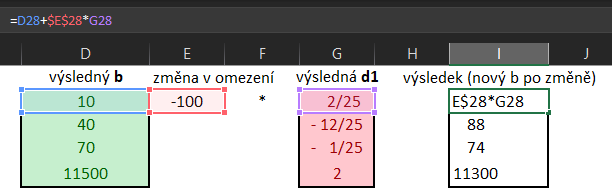
**Při změně nějaké hodnoty ve vektoru b:**

Vezmu výsledný **b,** změnu v omezení, a hodnoty doplňkové proměnné ve výsledném řešení, které se omezení týká (omezení1 = d1, omezení2 = d2 …)





Následně od výsledného b odečtu změnu \* výsledná d1



**Frobeinova věta**

**Nemá řešení** - hodnot matice A a rozšířené matice A|b si nejsou rovny

**Má řešení jedno** - počet proměnných stejný jako je hodnost matice A

**Nekonečně mnoho** - hodnosti jsou rovny, počet proměnných vyšší než je hodnost matice A

**Degenerované řešení** - pokud je ve vektoru pravých stran 0

**Vektor všech řešení** (více proměnných, než hodnost => nekonečně mnoho řešení):

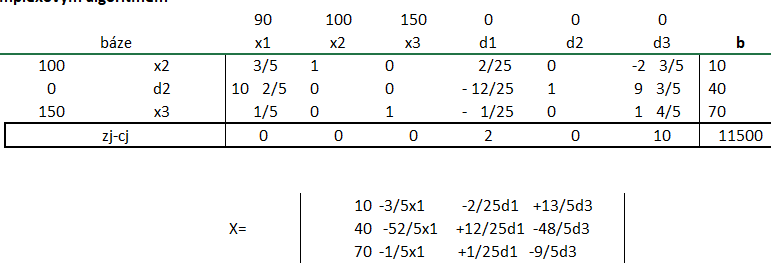
Vektor, kde pro nebazické proměnné píšeme x1,x2,.. a pro bazické napíšeme obecně tvar - pravá strana mínus hodnota proměnné x1, mínus hodnota bunky krát x2 atp

**Vektor bazických řešení** - vektor například:

vB -> (0,1,1,0,10,0) - proměnné x2=1, x3=1, d2=10

**Vektor parametrického (obecného) řešení**

Z bazického řešení si vezmu vektor pravých stran, hodím na začátek a pak na každém řádku odečítám nebazické proměnné

****

**Zjištění intervalu rezervy (interval stability složek pravých stran):**

Pro každý řádek si udělám podmínku ve tvaru:

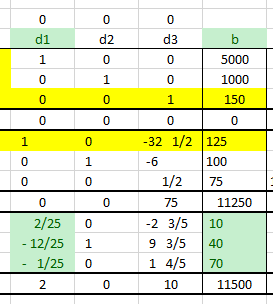
**MAX**: b + daná bunka proměnné \* q >=0

**MIN**: b + daná bunka proměnné \* q <=0

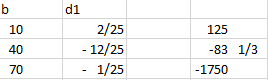
Interval pro změnu je to co vyjde.

Interval stability báze pro složku - interval po odečtení vlevo a po přičtení doprava a výsledek je interval

**zjednodušeně**:



Vezmu danou rezervu a spočítám **b/dx**

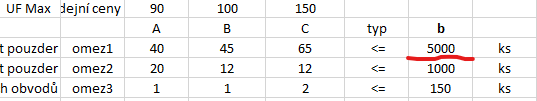
****

**vypočtu q:**

**pro MIN**: vybírám **nejmenší kladnou hodnotu**, následně **zapíšu jako zápornou**, když nenajdu -nekonečno

**pro MAX**: vybírám **nejméně zápornou hodnotu**, následně **zapíšu jako kladnou**, když nenajdu +nekonečno

**q** je elementem <-125;83,33'>

následně **b + q** (b z omezení)

<4875;5083.33’>

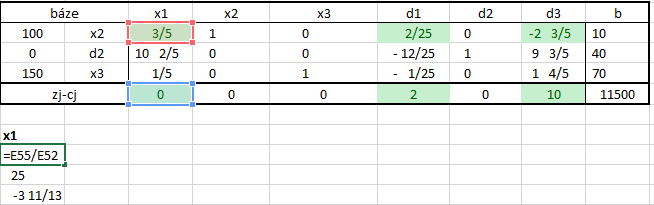
**Interval pro cenu**:

**pro bazické proměnné:**

Vezmu daný řádek pro bazickou proměnnou (ignoruju sloupec b) a postupně tvořím rovnice, vezmu **buňku kriteriálního řádku v daném sloupci + hodnota v daném sloupci/řádku \* proměnná v řádku.   
q >= 0 (Pro MIN je <=)** . Bazická jednička - ignoruju sloupeček.

**zjednodušeně:**

**Vezmu řádek zj-cj dané b. proměnné a vydělím to hodnotou z řádku dané bazické proměnné. (tady například počítám x2 -> 0 / ⅗, 2 / 2/25, …)**

****

**vypočtu q:**

**začátek intervalu**: vybírám **nejmenší kladnou hodnotu**, následně **zapíšu jako zápornou**, když nenajdu -nekonečno

**konec intervalu**: vybírám **nejméně zápornou hodnotu**, následně **zapíšu jako kladnou**, když nenajdu +nekonečno

následně **cena + q**

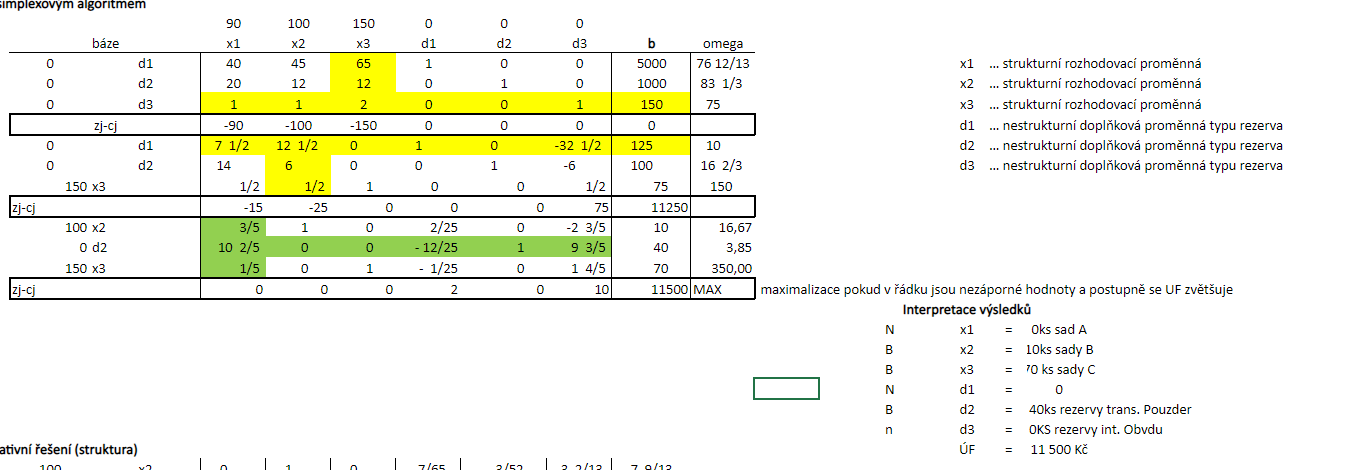
**Nebazická proměnná:**   
**Pro MINIMALIZACI**: interval <T; nek)  
**Pro MAXIMALIZACI:** (-nek; T>

Kde Nová cena **“T” je cena + nová kriteriální buňka ve sloupci dané proměnné**

Pokud je nutné ponechat nějakou rezervu volnou, tak se sníží ve vektoru b hodnota a cena se tedy zmení.

**Zjištění toho, co vyrábím**

Pokud se podívám do báze, tak tam je kolik čeho vyrábím a vpravo ve sloupci b je kolik ještě zbývá



**Vázaný extrém** - hl. abs. extrém - fce je omezena sadou omez. podmínek (Lagrangeova)

**Volný extrém** - hledáme lokální min/max

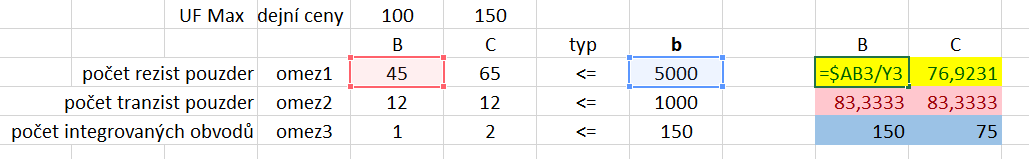
Grafický řešení Linéárního modelu

**Pro dvě proměnný:**

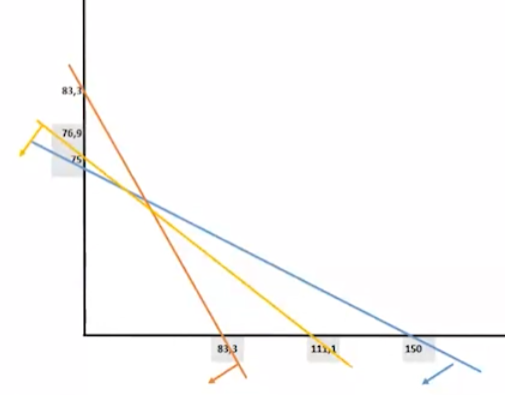
Dvě proměnný -> dvě osy

počet omezení -> počet přímek

Vydělit vektor b vektorem proměnné

****

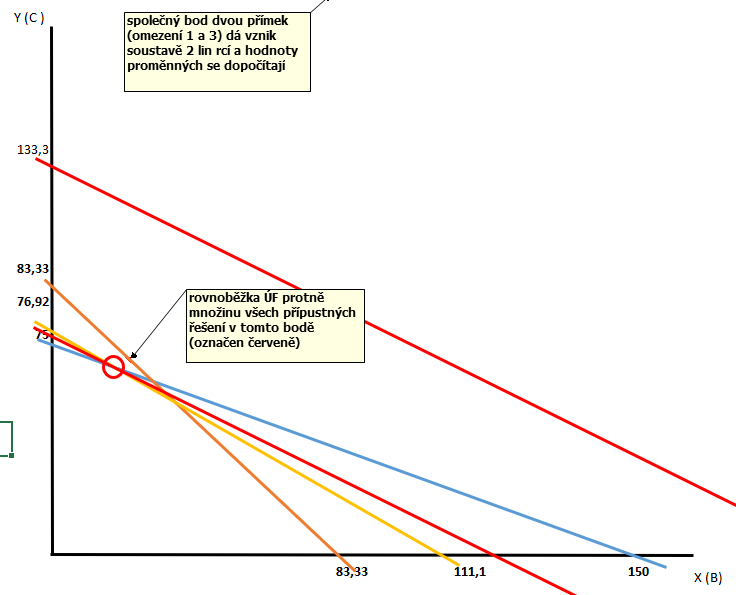
Následně zanesu na osu, tam, kde se přímky protínají, nachází se ideální řešení



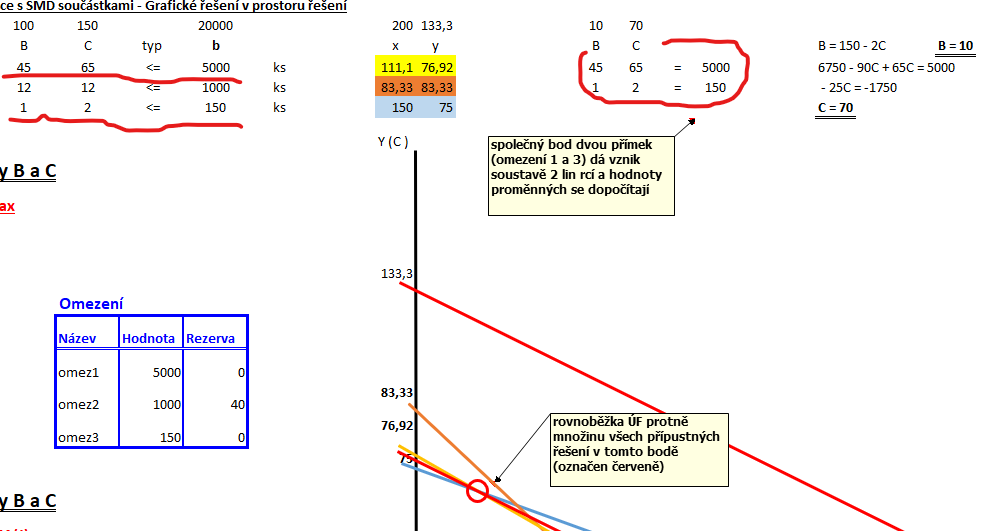
Následně si **dosadím za hodnotu UF co chci,** abych zjistil sklon přímky, která bude sloužit k vyobrazení plochy všech přípustných řešení, **vydělím hodnotu UF cenami obou proměnných**, dostanu průsečíky s osami

****

Opět zanesu do grafu a křivku posunu rovnoběžně do bodu, kde se protínají ty dané osy (červená)

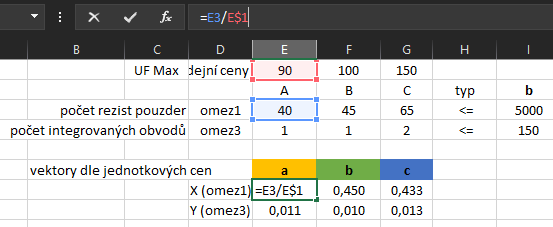


z přímek, který se protínají nám vznikne optimální bod, vypočteme ho tak, že vezmem rovnice (omezení) těch přímek, který se protínají a vyřešíme je jako soustavu rovnic (**lze i JEM)**



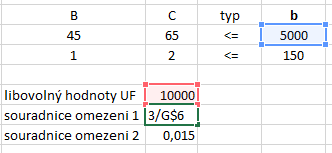
**Pro dvě omezení:**

Složky v daném omezení vydělím cenou proměnné, z toho mi následně vzniknou **vektory s počátkem v nule, co proměnná, to vektor**

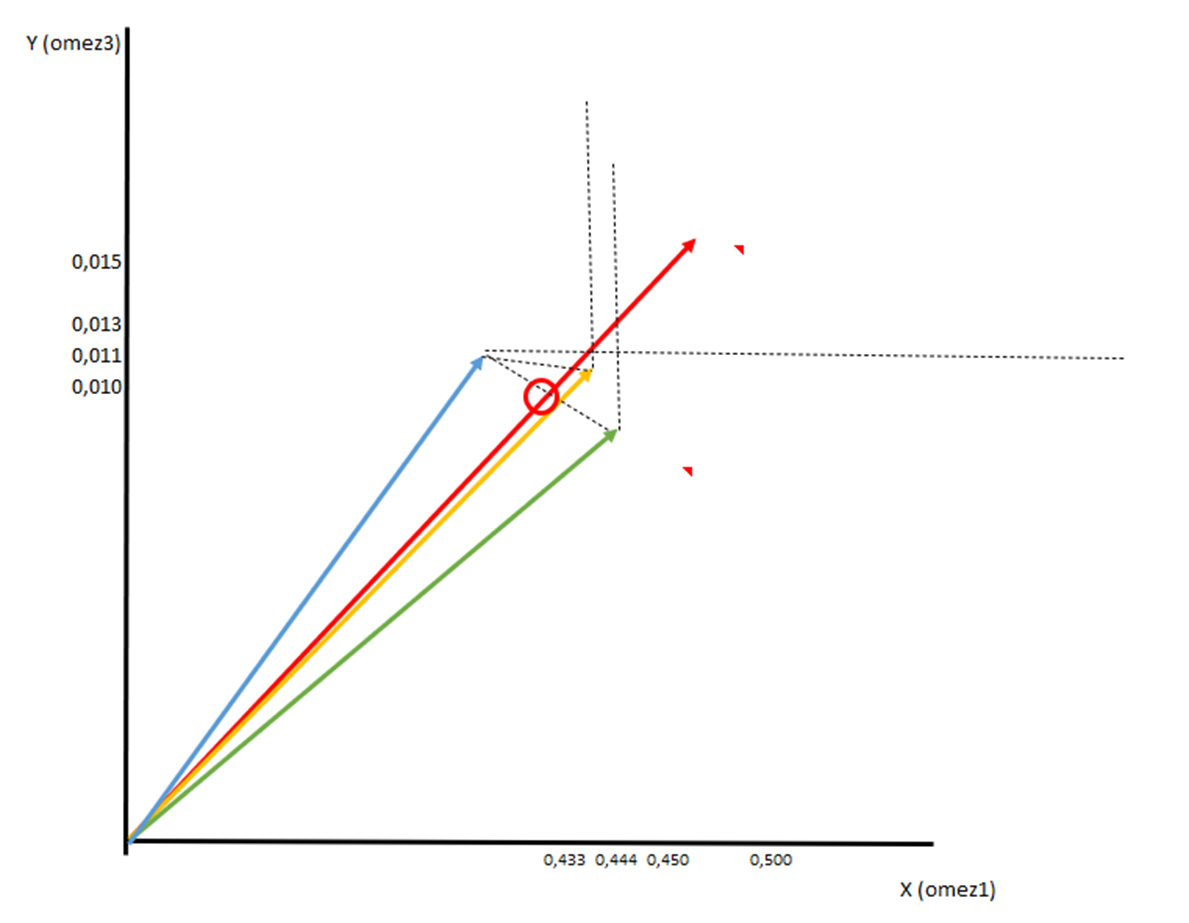


**Požadavkový vektor**

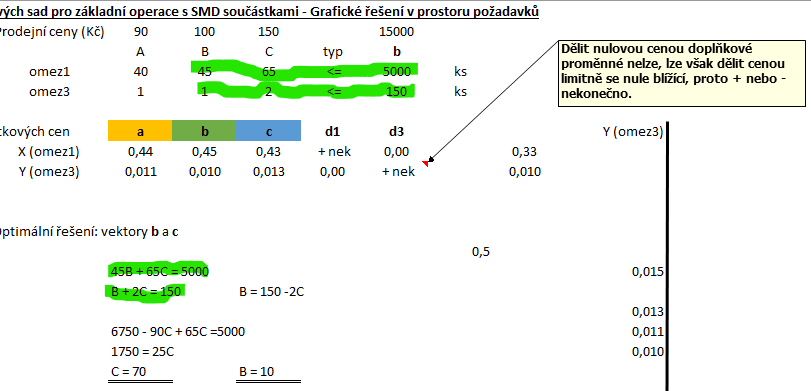
Zvolím **libovolnou hodnotu UF** a vypočtu **b / (libovolna hodnota UF)**

****

**Zanesu do grafu**

****

Vypočítám optimální bod tak, že **si vezmu ty dva vektory, které “půlí” požadavkový vekto**r a opět vypočítám u obou proměnných jejich omezení jako soustavu rovnic a zjistím optimální bod



Dopravní úlohy

**Model je řešitelný**, když je vyvážený = stejná suma kapacit dodavatelů a požadavků odběratelů. Pokud nesedí suma, přidám buď VDF nebo SF (F-fiktivní), tak aby seděla suma

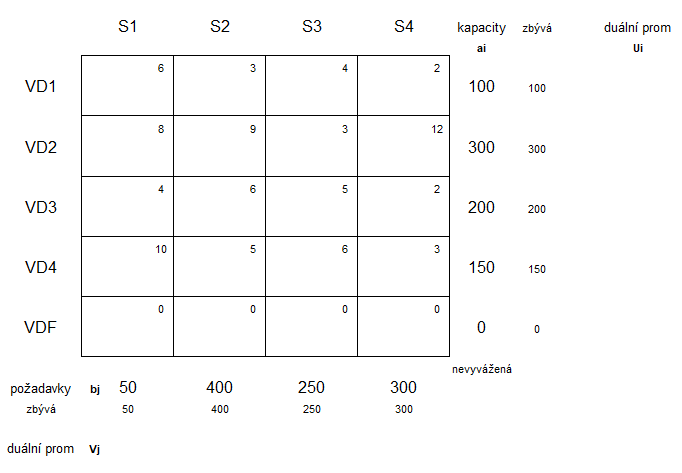
**Počet omez. podmínek** - součet odběratelů + dodavatelů

**Počet proměnných** - počet dodavatelů \* počet odběratelů (fiktivní se počítá také)

**Počet baz. proměnných** - počet dodavatelů + počet odběratelů - 1 (kvůli lineární závislosti)

**1.úrovnová úloha** = rovno od dodavatelů ke spotřebitelům

**Indexová metoda**



**Před výpočtem:**

1. Kontrola, zda nemáme záporné proměnné
2. Vyvážení dopravní úlohy

Celková zásoba == celkové požadavky. Pokud má obchod více, zbyde mu zboží a musíme vytvořit fiktivního zákazníka (zboží zbyde ve skladě). Naopak pokud zákazník chce více, musíme vytvořit fiktivního dodavatele.

**Postup:**

1. Vyberu nejmenší index dopravy (**vpravo nahoře v buňce**) (**nikoliv u fiktivních, ty nechám na konec**). Pod něj napíšu kapacitu, kterou můžu maximálně převést s ohledem na zbývá ve sloupci a řádku.
2. Upravím položku zbývá v sloupci a řádku. Pokud již vyčerpám sloupec nebo řádek, proškrtnu ho.
3. Následně hledáme další nejmenší prvek, krom těch, kterých jsou proškrtlý, dokud nevyčerpáme všechny kapacity
4. Zjišťuje degeneraci a pokračuju v MODI
5. UF se zjistí **sečtením násobků bazických proměnných a indexů**

**Vogelova metoda**

**Postup**:

1. Vyvážení dopravní úlohy jako u indexové
2. Vypočtu si diference pro každý řádek a sloupec. Vezmu rozdíl dvou nejmenších čísel v řadě. Pro dvě stejné v řadě je diference 0, platí i pro poslední zbývající číslo.
3. Vyberu největší diferenci a v tom vyberu nejmenší index dopravy (klidně i fiktivní). Když mám na na výběr z více, vyberu si tu, kde je menší index. Pokud se vyčerpá kapacita, vyškrtnu řádek/sloupec.
4. Následně opakuju vypočtením diference a hledáním největší diference a nejmenšího indexu.
5. Zjišťuje degeneraci a pokračuju v MODI
6. UF se zjistí sečtením násobků b. proměnných a indexů

**MODI optimalizační metoda**

**1. Zjištění degenerace**

* Pokud splníme najednou dvě podmínky kapacity při výpočtu, vzniká degenerované řešení (při poslední kroku to neplatí)
* nebo pokud je se počet bazických cest (proměnné ve finálním řešení) **!=** p.zákazníku + p.dodavatelů - 1
* Počet degenerací je rozdíl mezi těmito čísly z předchozího bodu

=> Pokud se tak stane, vyškrtnu daný sloupec. Nutno přidat další bazickou trasu k trase samotné v řádku i sloupci – první řádek nebo poslední sloupec. **Epsilon se napíše tam, kde se k němu nedostanu Danzigovym obvodem**

**2.** Musí platit počet rovnic ( baz. proměnné xij bez fiktivních + eps) == počet baz proměnných (nezapomenout na eps z předchozího kroku - stále čeká na doplnění)

**2. Dantzigův obvod - 1-4 TEST OPTIMALITY**

**1**. Do tabulky si doplním sloupec U a řádek V. Do **řádku/sloupce**, **kde je nejvíce baz. proměnných** si za duální proměnnou doplním **0** (když jsou dva, vezmu jeden, je to jedno). Následně mi musí vždy sečtení U + V dát index, který je v dané buňce - jen u bazických proměnných. Vyplním takto všechny U a V.

**2.** Do každého levého dolního rohu buňky se napíše daný součet z = u+v :  
Vzorec do excelu: =Sloupec$Řádek+$SloupecŘádek (V + U)

**3.** Do levé buňky nahoru (perspektiva/hodnota testu optimality) se všude doplní z - c. Tedy bunka vlevo dole mínus bunka vpravo nahoře

**4.** U **MIN** by mělo u **nebázických proměnných** z - c (perspektiva) vyjít všude záporné (jako u simplexu krit. řádek). Pokud to neplatí, tak hledám tedy to nejvíce kladné číslo vlevo nahoře v bunce

U **Max** zase všude kladné z-c. Pokud to neplatí, tak hledám tedy to nejvíce záporné z-c

**Pokud hodnoty vyhovují, tak je úloha hotova.**

**Pokud naleznu nevyhovující hodnoty**:

1. tak si v dané nevyhovující největší bunce napíšu plus a snažím se udělat obvod (ideálně co nejlehčí.

2. Znaménka +-m se střídají (Vždycky po projítí bazic. proměnné se musím otočit o 90 stupnu nebo pres ni jet dal)

3. Vyberu **nejmenší číslo v bazické hodnotě označené “-m”** a zapíšu si to m=hodnota té vybrané buňky a doplním ji do nevyhovující buňky a tu starou buňku vyřadím z báze a nastavím na 0

4. Poté postupně přepočítávám obvod. Bud přičtu hodnotu m nebo ji od dané buňky odečtu (podle znaménka u m).

**TIP**: Pokud se stane, že se vyskytují dvě stejná čísla u dvou stejných známínek u m, tak se musí u toho nevybraného místo 0 napsat eps (aby nevznikla degenerace nebo nevypadla z báze).

Opakuju od dantzig. kroku 1. včetně

**Výsledek optimalizace:**

**Alternativní řešení** **existuje** - perspektivita je 0 u **nebazických proměnných** (hodnota vlevo nahoře v bunce je nula). Ostatní musí být <=0

**Při změně cenového indexu/koeficientu:**

Nebázická proměnná - pouze dojde ke změně perspektivy, nemusí změnit optimalitu (pokud je perspektiva stále záporná)

Bázická proměnná - dojde ke změně celého řešení pravděpodobně

**Interval cenového indexu** - <levý spodní index; inf)

**Propustnost tras** - Nejmenší index (pravá horní buňka) z nebazických buněk (prázdné trasy)

|  | |  | |  | |  | |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

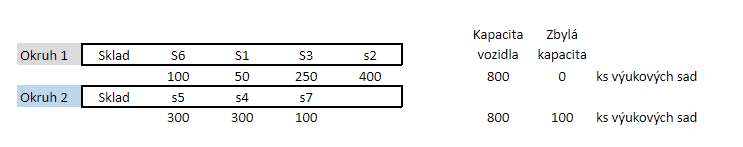
# 

# Okružní úlohy

Poznám to obvykle podle toho, že se píše kapacita přepravy, tedy bude nejspíše nutné nějak rozdělit na podokruhy v porovnání s klasickou dop. ulohou.

**Mayerova metoda (rozdělení do skupin na jednookružní okruhy)**

1. vybereme misto nejdále od výchozího místa (obvykle sklad)
2. Daný sloupeček vyškrtnu
3. Od něj pak místo nejbližší (v rámci řádků) a vyškrtnu sloupec
4. Dále pak nejbližší od všech již vybraných
5. Pokračujeme, dokud součet kapacit není takový, že přidáním dalšího místa překročíme kapacitu
6. Jsou-li více než dva okruhy, pak vybraná místa prozatím vyškrtáme a pokračujeme stejným postupem ve výběru dalšího okruhu
7. Metoda rozdělí do skupin, následně využíváme metody níže na každý okruh (**Výchozí bod/místo se ponechá v obou tabulkách** - chci tam dojet žejo). Počet okruhů bych měl zjistit z celkového součtu / kapacita 1 rozvozu.
8. Zapíšu tabulkou

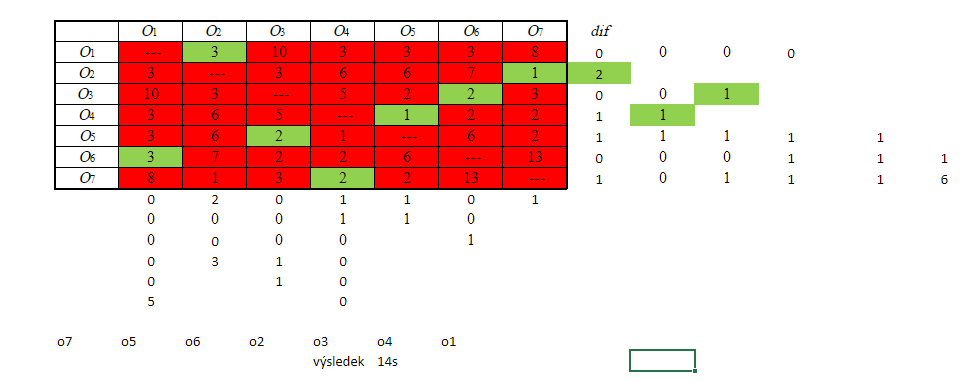


Metody pro 1. okružní problémy:

Nakopíruju si tabulku se sloupci v rámci skupiny v případě převodu z víceokružních.

**Vogelova aproximační metoda**

1. Napíšu si diference sloupců a řádků. Vždy beru rozdíl dvou nejmenších čísel v daném řádku/sloupci. Pokud je tam například 3 a 3, tak je diference 0.  
   vzorec na diferenci **SMALL((...;...;...;...);2)-SMALL((...;...;...;...);1)**
2. Vyberu nejvyšší sazbu z diferencí.
3. Vyberu pro daný řádek/sloupec nejmenší číslo a proškrtnu celý řádek a sloupec a cestu zpět (nelze se vracet) a cestu do destinace, ze které se začíná.
4. **Dále musíme proškrtnout místo, tak abychom nemohli vytvořit předčasně kruh,** proškrtáváme místa, které už jsme navštívili (zapisujte si pořadí měst). Také smyčku u skupinek.
5. Dopíšu do pořadí vybraný sloupec (pořadí průchodu). Pokud se skládají skupinky,
6. Opakuju pořád dokola, než se vyškrtne vše. Co zbyde (obvykle 1 prvek dopíšu také do řady). Až už nemůžu vytvořit dvojici, zkusím se na to podívat a pospojovat, co zbyde logicky - aby to dávalo smysl
7. Následně vybraná pole napíšu za sebe v pořadí a sečtu jejich váhy - to je nejkratší průchod



**Metoda nejbližšího souseda**

1. Vytvořím si tabulku, kde v každé tabulce **začínám postupně každým začínajícím místem**, Sklad, S1, pak začínám S2 v další tabulce atp)
2. Od daného místa v tabulce volím místo nejbližší na řádku a proškrtnu sloupec. **Kontroluju cestu, kterou bych se zase mohl vrátit tam, kde jsem již byl, KONTROLUJI AKTUÁLNÍ SLOUPEC A VŠECHNA ZAZNAMENANÁ MÍSTA**
3. Od toho místa zase vybírám nejbližší místo, proškrtnu sloupec
4. opakuju pro každou tabulku, výsledek je cca toto:

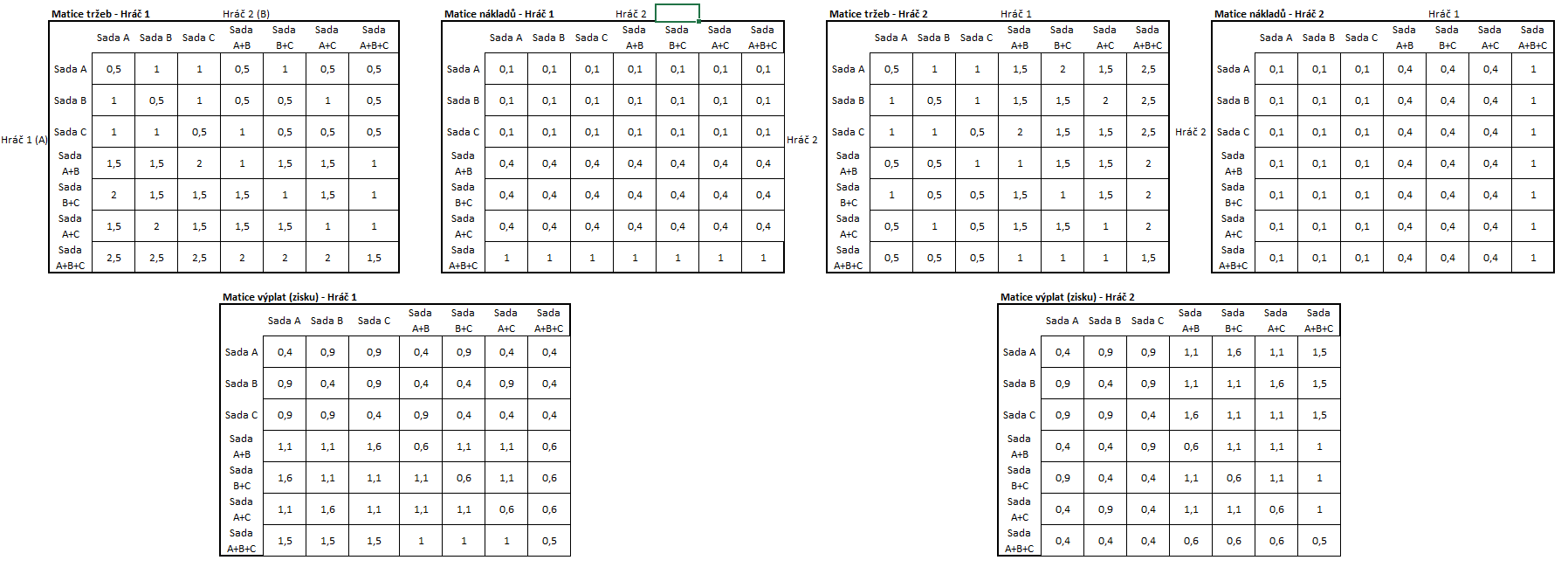
|  |  |  |  |  | suma |
| --- | --- | --- | --- | --- | --- |
| Sklad | S7 | S5 | S4 | Sklad | **13** |
| S4 | S5 | S7 | sklad | S4 | **13** |
| S5 | S4 | S7 | sklad | S4 | 16 |
| S7 | S5 | S4 | sklad | S7 | **13** |

1. Následně si pro všechny varianty pořadí vyberu místo, které má nejmenší sumu a ta je ta nejlepší

**!!** Při třech místech/ 4 cestách stačí jen jedna tabulka - vždy vyjde stejně

nejvyšší propustnost nebázických(ty který jsme nevybrali) je nejmenší NEIMAGINÁRNÍ číslo v pravém horním rohu.

Teorie her



**Určení typu hry:**

**Kooperativní x Nekooperativní** - zda se hráči můžou domluvit

**S konstatním/nekonstantním součtem** - zda součet výplat obou hráčů je v každé kombinaci strategii stejný - konstantní.

**s konstatním/nulovým součtem** - výhra jednoho je ve stejné výši prohra druhého

**Hra s nekonstantním součtem**

Dva hráči, například konkurenční hra, šachy atp., nekooperativní. Náklady a tržby vezmu ze zadání.

V každé matici pro jednotlivé hráče (levý obrázek) se v řádcích uvádí daný hráč, a ve sloupcích hráč opačný

Z toho vyplývá:

**Pokud oba hráči mají stejnou strategii**,

jedná se o **čtvercové matice**, pak můžeme **matice druhého hráče určit transpozicí matic** hráče prvního a naopak. =TRANSPOZICE(druhá matice) (potvrdím ctrl + shift + enter)

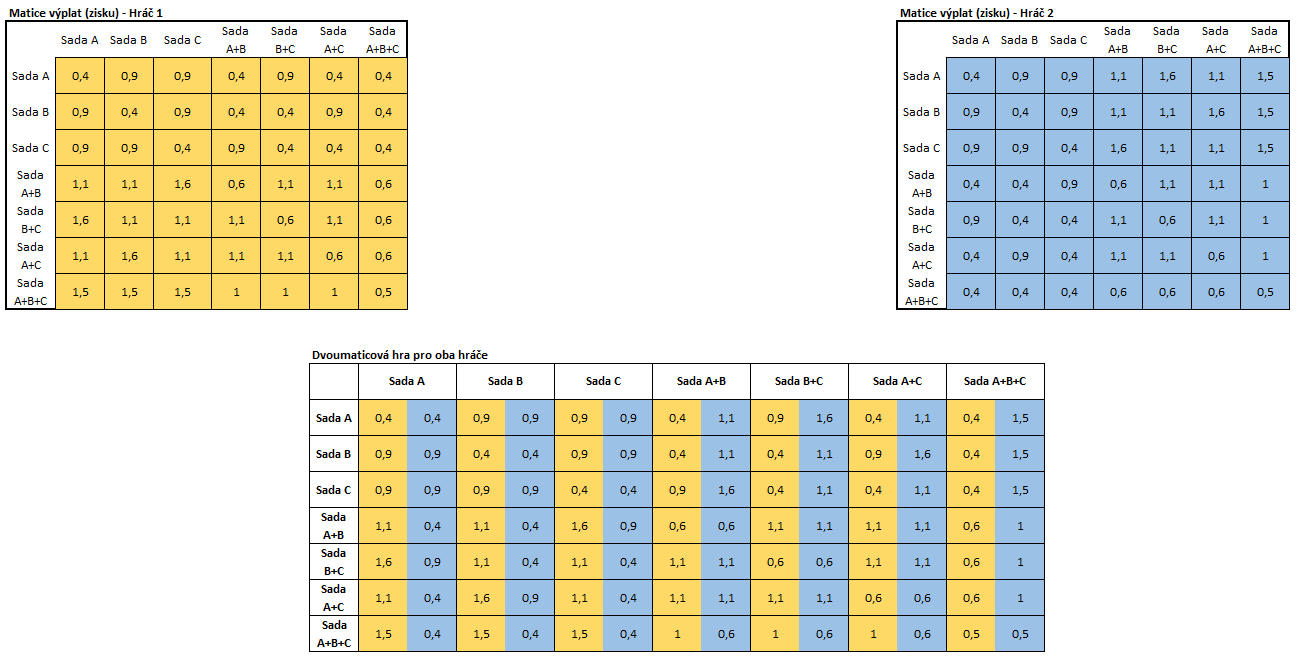
U obou sestavím tři matice, tržby, náklady a **výplatní matici,** ve výplatní matici se uvadí, kolik hráč “získá” v dané strategii, ze zadaní je určeno jestli jde o zisk, tržby, či cokoliv jiného.

**Čistá strategie** - jednoznačně určená strategie hráče => má sedlový bod

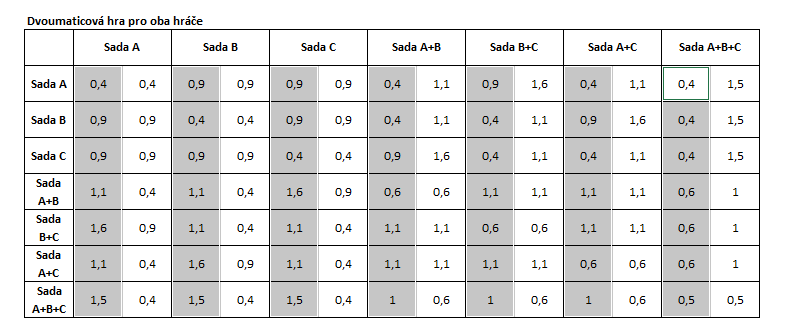
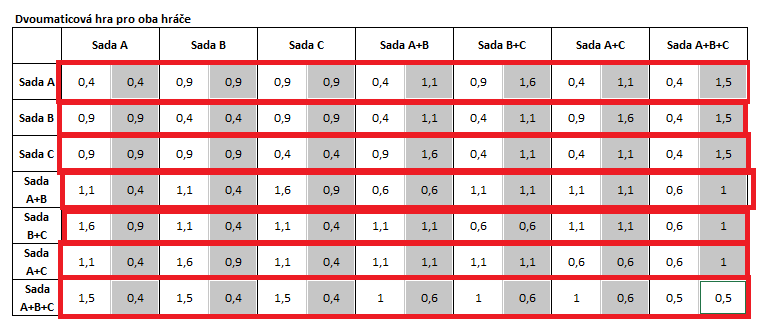
**Smíšená strategie** - pro každou strategii je dána pravděpodobnost jejího použití - četnost použití při opakování hry - při mnoha partiích

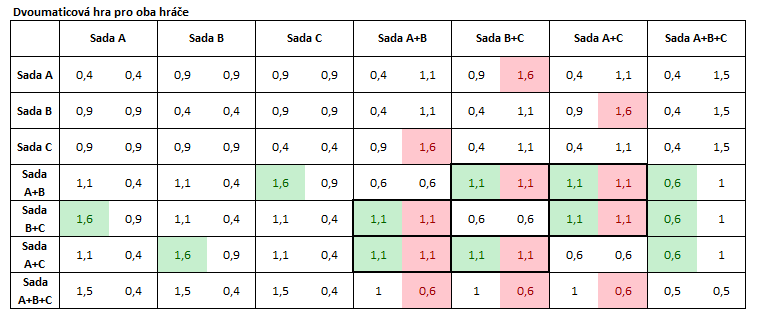
**Sestavení dvoumaticové hry pro oba hráče**

Vložím do této tabulky hodnoty z **výplatních matic** obou hráčů, levý sloupec hráč 1, pravý hráč 2



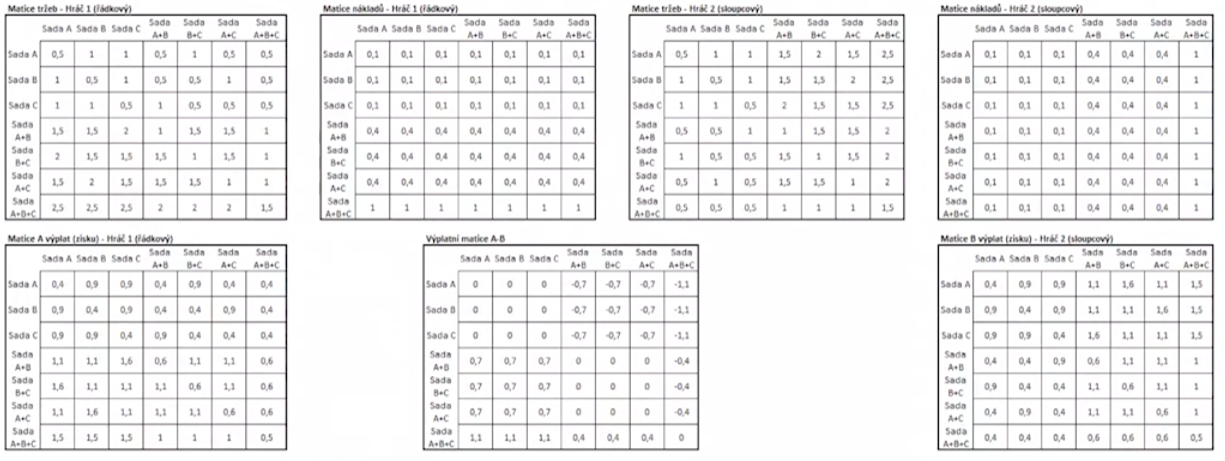
Určím **Nashovy rovnovážné body -** doporučená **optimální strategie pro oba hráče,** když ji budou **oba dodržovat,** pokud ne, **ten, kdo ji nebude dodržovat, si může pouze pohoršit.**

1. Ze zvýrazněných buněk vyberu pro **každý sloupec** nejlepší varianty (Maximální hodnotu/y), **označím podmíněným formátováním,** vyberu zvýrazění prvního hráče
2. Ze zvýrazněných buněk zvýrazním pro **každý řádek** maximalní hodnotu/y (pouze ALE hráče B ne A)  
   
3. Nashovi rovnovážné body zvolím tam, kde mám **zvýrazněné buňky obou dvou hráčů**



**Hra s nekonstantním součtem** - výplaty v bunkách se ruzni

**Hra s konstantním nulovým součtem**



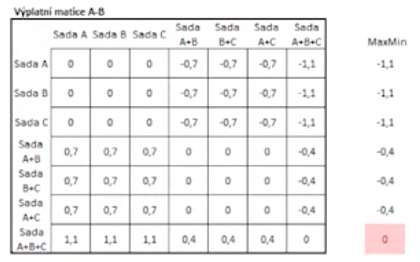
Opět vypočtu matici výplat obou hráčů.

V tomto případě řesím, o kolik hráč 1 dostane více zisku než hráč 2, v tom případě **odečtu matici výplat hráče 1 od matice výplat hráče 2.**

**Horní a dolní cena hry.**

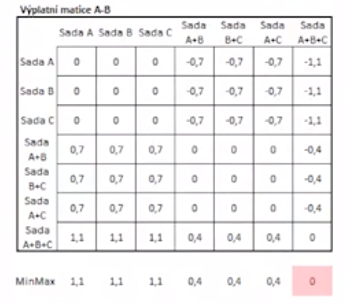
**MAXIMIN (spodní cena)** - Hraju na jistotu, z **MINIMA řádků** vyberu **MAXIMUM**

funkce **min()**, následně podmíněné formátování



**MINIMAX (horní cena)** - z **MAXIMA sloupců** vyberu **MINIMUM**

funkce max(), následně podmíněné formátování

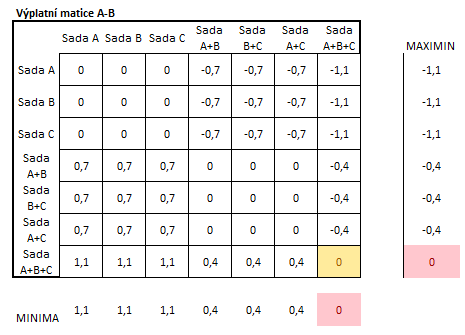


**Sedlový bod hry**

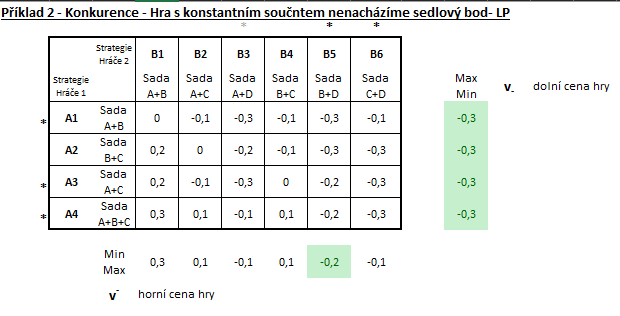
Sedlový bod **existuje, pokud horní a spodní cena hry se rovnají**.

Sedlových bodů může být i více.

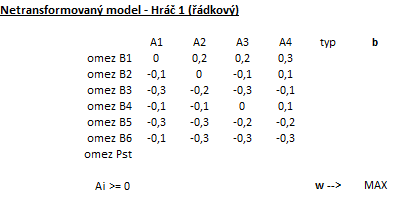
V tomto případě 0 = 0



**Řešení hry bez sedlového bodu => převod na LP (není sedlový bod)**

****

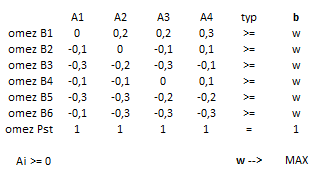
Strategie druhého hráče je omezení hráče prvního, proto **vytvoříme netransformovaný model** pro hráče tak, že **vezmem strategie druhého jako omezení prvního hráče**(v tomto případě musím udělat transpozici)



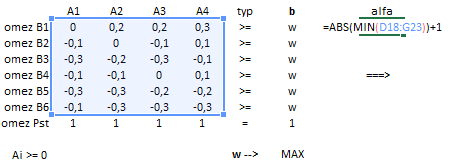
Jedná se o **MAX, tudíž typ je požadavkový**, **pokud by se jednalo o sloupcového hráče, jednalo by se o MIN a typ by byl kapacitní**

Omezení jsou také pravděpodobnosti co si zvolíme ve smíšené strategii, tudíž jejich součet musí být 1.

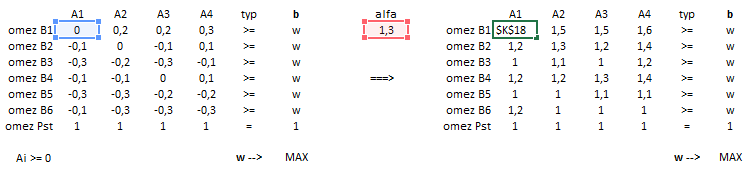
Za vektor pravých stran dosadíme neznámou w.



Spočítám alfu tak, že vyberu z celé matice **MINIMUM v ABSOLUTNÍ HODNOTĚ a PŘIDÁM 1,** aby mi nevycházela následně někde nula.

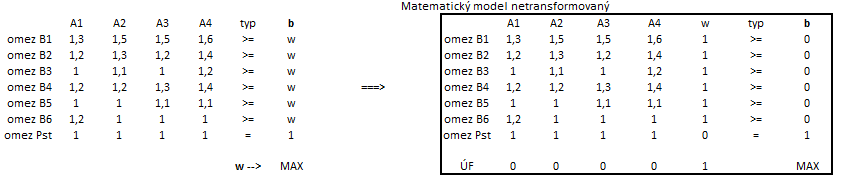


Následně **přičtu** alfu



Převedu **w** na levou stranu, proto mi v **b** zbyde 1 na konci a v w zase 0

V ÚF se snažím maximalizovat w, proto jen jedna 1 na konci

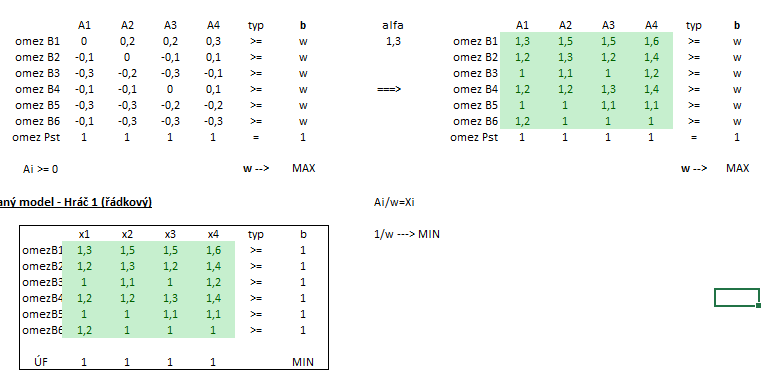


**Transformovaný model**

Ai / w = Xi

1/w → ÚF musím mít opak netransformovaného modelu, jelikož dělím w, tudíž čím menší w, tím větší Xi

ÚF - v ÚF mám místo w Xi, tudíž samé jedničky

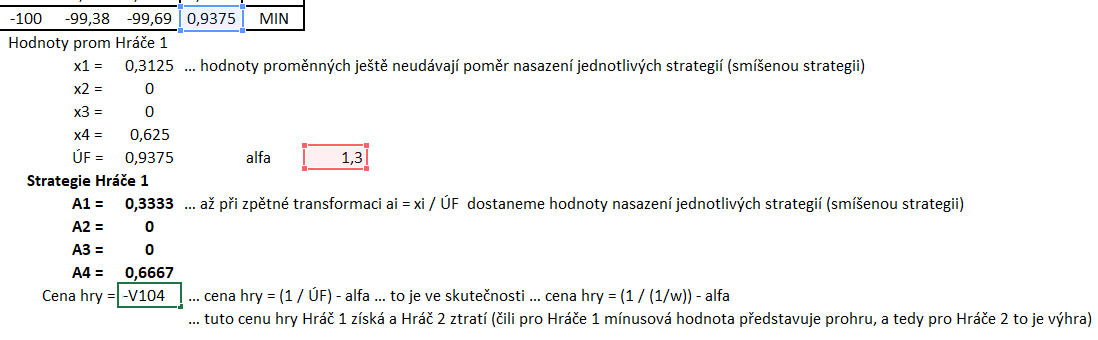


Řešení:

Po vyřešení simplexem si vypočítám cenu hry

cena hry = (1 / ÚF ze simplexu) - alfa

pokud je záporná, hráč tratí, pokud kladná, získává, druhý hráč to má naopak

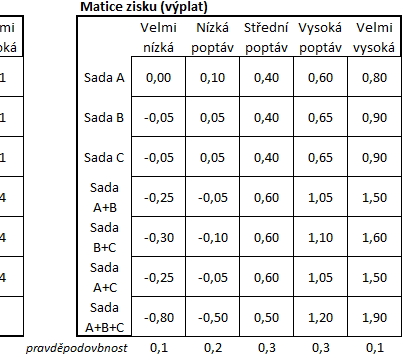


Teorie rozhodování

Neboli hra proti přírodě, rozhodovací model, hra proti neinteligentnímu hráči

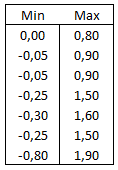
K matici zisku přidám i pravděpodobnost stavu okolností.

Pořadí Varianta - stav okolností - výplata

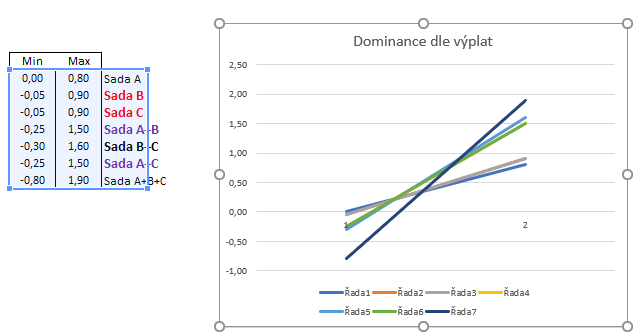


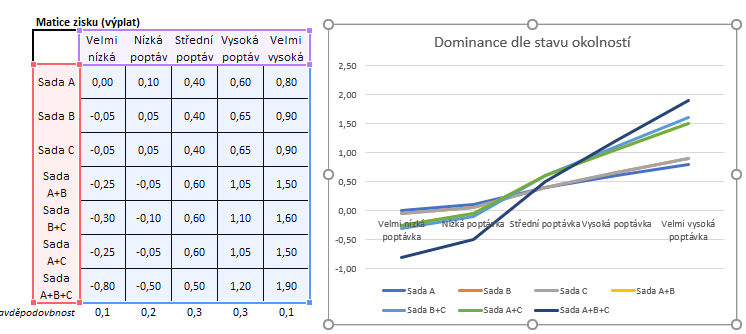
**Dominance dle výplat a stavu okolností**

Vezmu minimum a maximum z každého řádku Výp. Matice



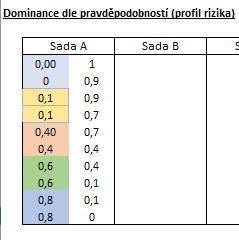
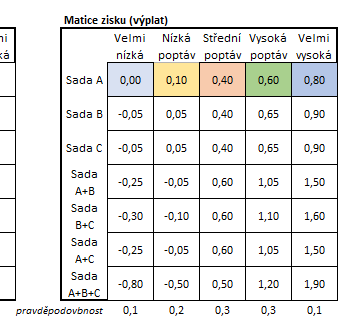
Z toho vytvořím grafy pokaždé je osa x název sady a osa y minimum až maximum





**Dominance dle pravděpodobnosti (profil rizika)**

**Levý sloupec** - vezmu nejhorší (minimální jistou) pravděpodobnost, opíšu do dalšího řádku, takhle až do konce sloupce

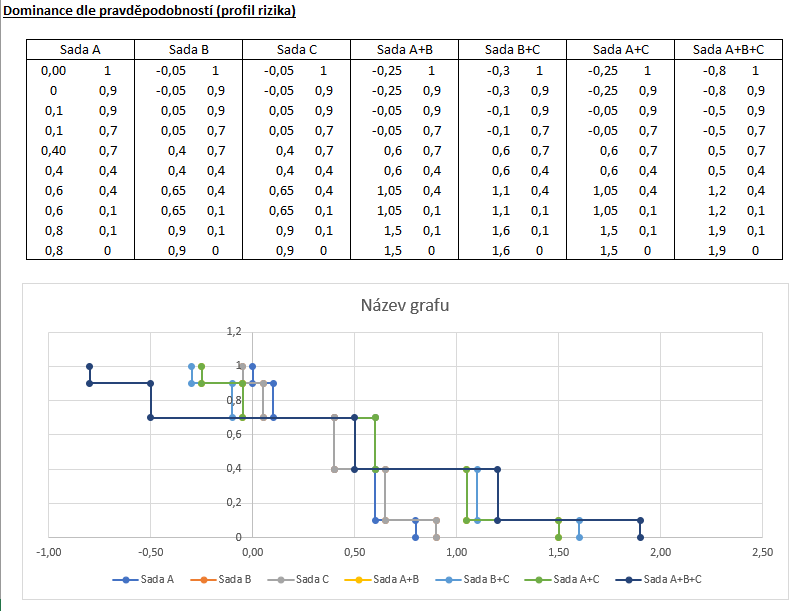
**Pravý sloupec -** Jdu od pravěpodobnosti 1 až k 0, napíšu 1, pak odečtu první pravděpodobnost, následně přepíšu a odečtu další … až do konce sloupce  


Učiním tak pro každou sadu

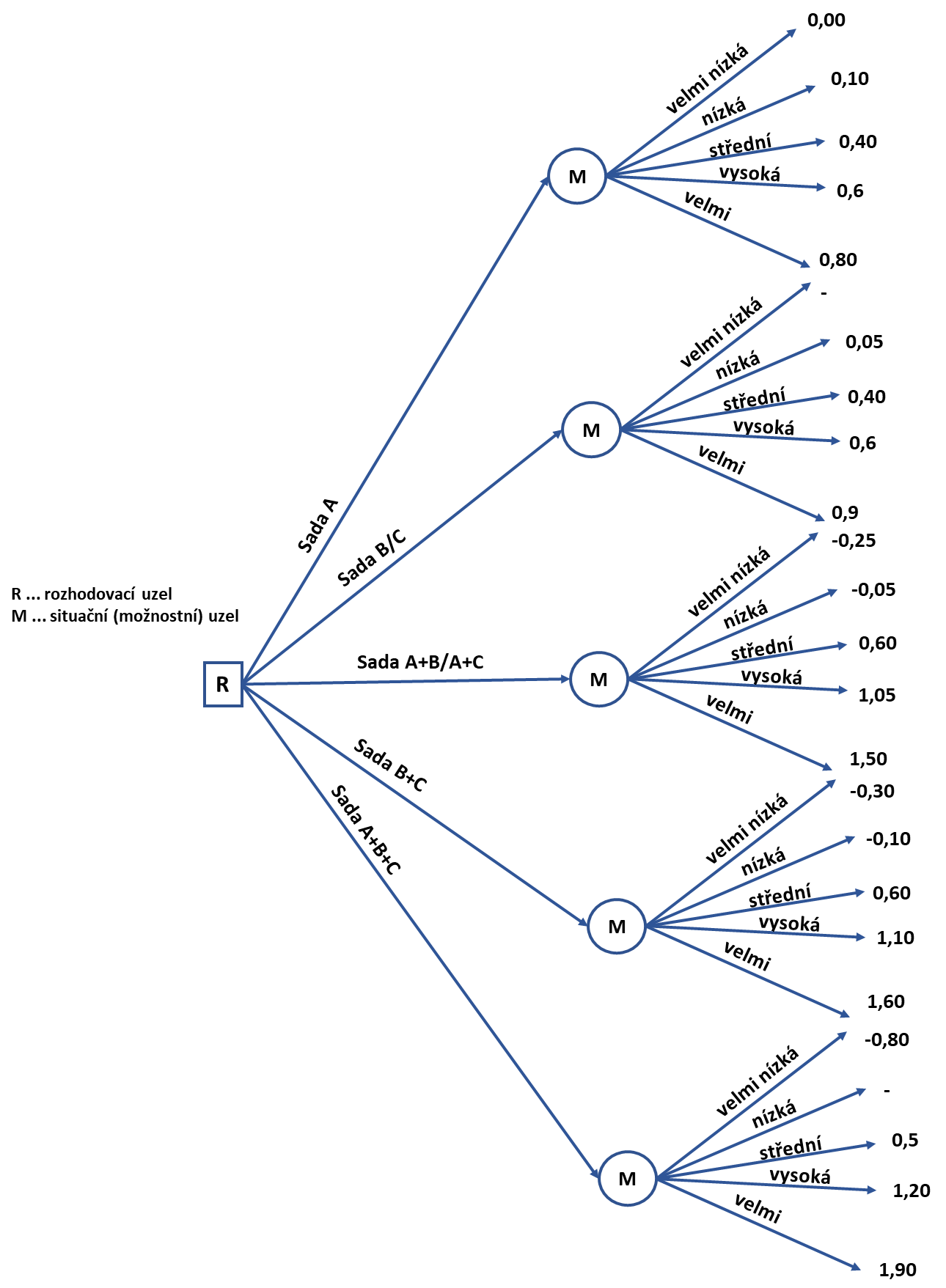
Hint: **Pravý sloupeček se nemění, můžu překopírovat na celou tabulku**

Vložím graf, pokud je **jedna sada jasně nad druhou, neprotínají se, dominuje jí**

**Graf:** Bodový s rovnými spojnicemi a značkami

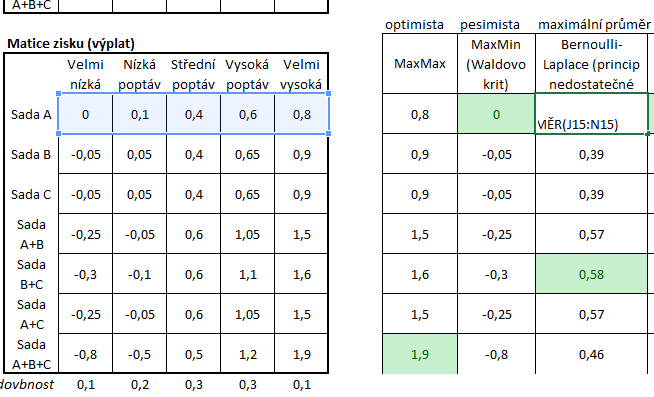


**Rozhodovací strom**

****

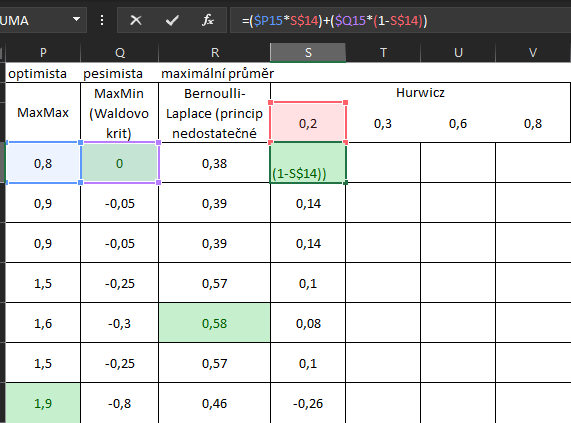
**Bernoulli-Laplacovo pravidlo**

vezmu to nejlepší z průměru

****

**Hurwiczovo pravidlo**

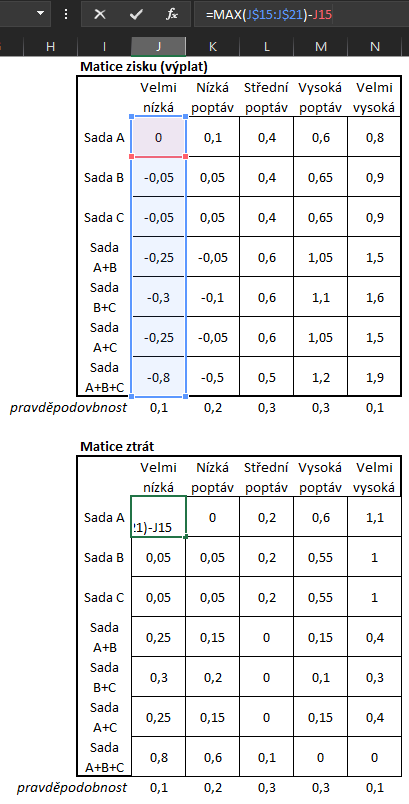
Do červeného řádku vložím pravděpodobnost, sčítám **MAX** a **MIN** s ohledem na pravděpodobnost, z toho vybírám to nejlepší

****

dále musím funkci **kopírovat sloupec po sloupci**, aby fungovalo podmíněné formátování

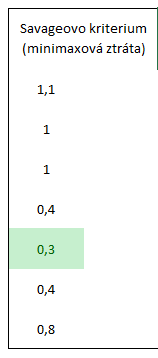
**Matice** **ztrát**

Kolik tratím v každé alternativě oproti MAX zisku v dané okolnosti



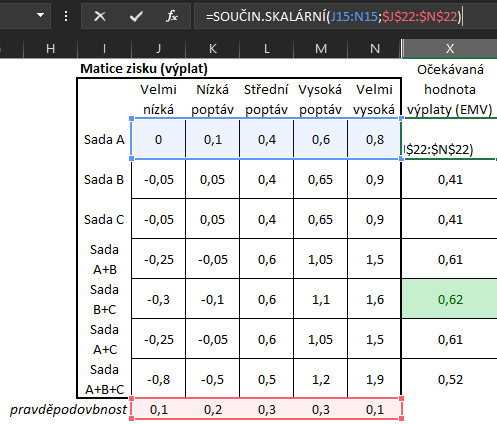
**Savageovo pravidlo**

Jedná se o **MINIMAX** z **MATICE ZTRÁT  
=MAX(řádek) -> z těchto hodnot tu nejmenší**

****

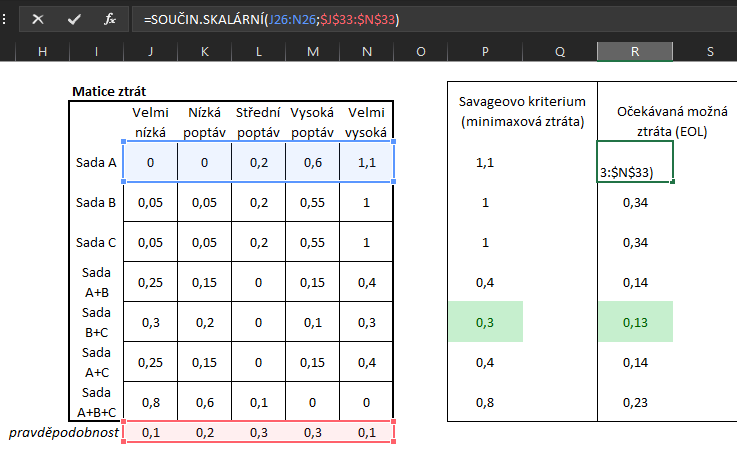
**Očekávaná hodnota výplaty (EMV - Expected Monetary Value)**

Jedná **SKALÁRNÍ SOUČIN ALTERNATIVY A PRAVĚPODOBNOSTI v MATICI ZISKU**

****

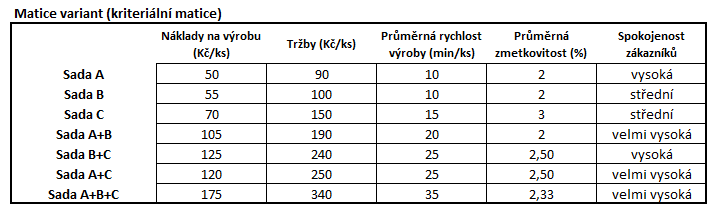
**Očekávaná možná ztráta (EOL - Expected oppurtunity loss)**

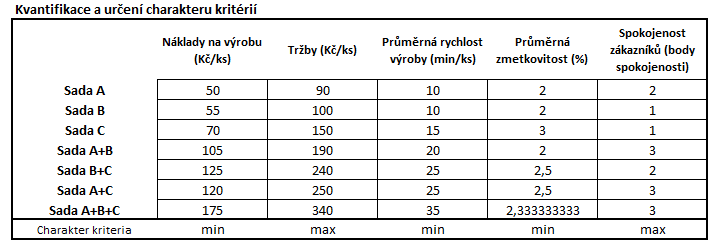
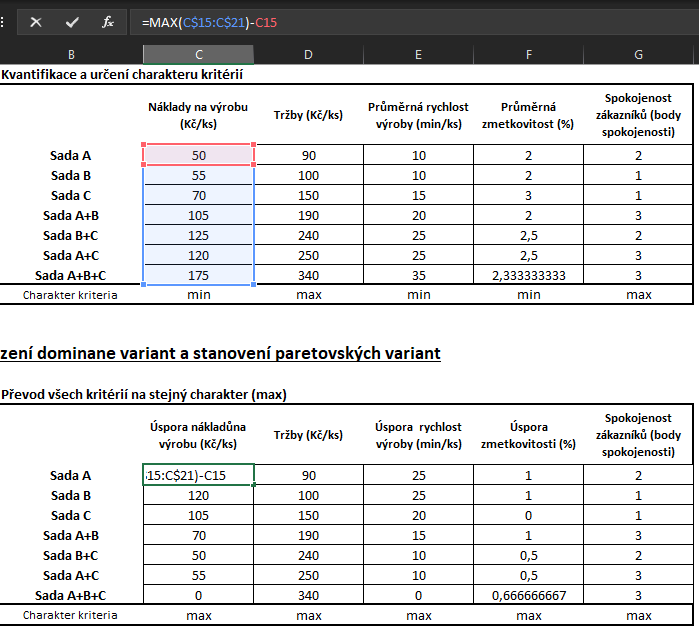
Jedná **SKALÁRNÍ SOUČIN ALTERNATIVY A PRAVĚPODOBNOSTI v MATICI ZTRÁT**

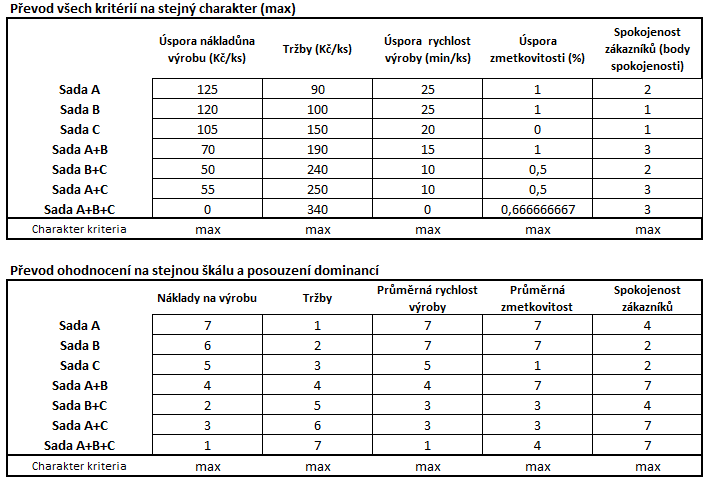
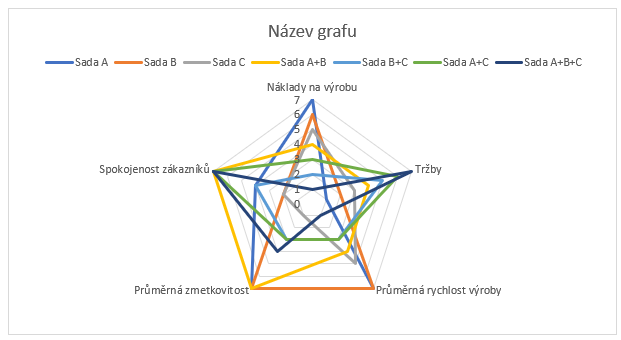
****

Vícekriteriální rozhodování

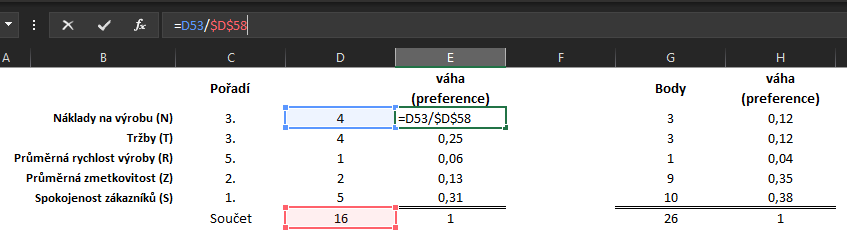
V zadání **Matice variant** aka (**Kriteriální Matice**)



1. **Kvantifikace a určení charakteru kritérií:**  
   Slovní ohodnocení si převedu na škálu čísel  
   Charakter kritéria min/max, například náklady budou min, jelikož je chci mít co nejmenší  
   
2. **Posouzení dominance variant a stanovení paretovských variant**Nejdříve převedu vše na maximalizační kritéria: U min vezmu **max(oblast sloupce)-daný řádek**  
   

Převedu na stejnou škálu, takže **jdu od počtu variant k 1**Pokud je více stejně hodnotných variant, ohodnotím stejně, ale následně přeskočím takový počet 1 v stupnici dolů  
například: (3,3,3,1) 4 čísla, trojky ohodnotím 4, přeskočím 3 a 2, jelikož tam jsou 3 trojky, 3-1 je 2, takže přeskočím dvě čísla (3 a 2) až na jedničku (4,4,4,1))  
  
vytvořím graf, ze kterého je jasné, jaká sada dominuje jakou (když se neprotínají, tak ta, která je uvnitř té druhé je dominovaná, druhá dominující) **Zde zelená sada dominuje modrou, tudíž modrou vyškrtnu**  


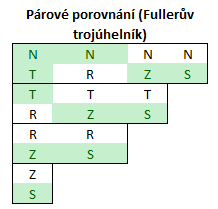
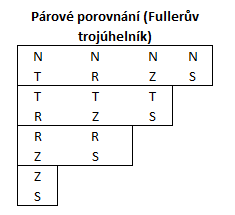
**Stanovení vah kritérií (Metoda pořadí a bodovací)**

Mám dáno ze zadání pořadí (popřípadě si vymyslím), převedu na stejnou škálu jako u předchozího kroku a následně určím preference tak, že vydělím sloupec před preferencí celkovým součtem viz. obrázek. Body mám taky dané ze zadání (nebo si domyslím, ale **musí být zachované pořadí**) a provedu to stejné.  


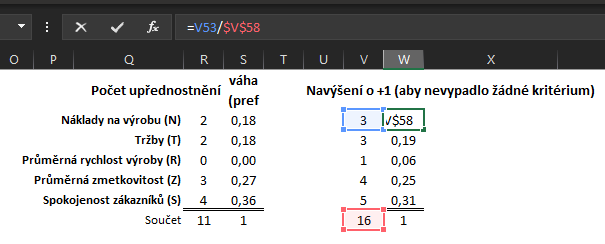
**Fullerův trojúhelník**

Porovnávám první atribut se všema zbývajícíma a tak dále, vždycky ten porovnanej už můžu vynechat, proto vznikne tento trojúhelník

Následně zvýrazním ty důležitější z toho páru (nebo oba najednou, jsou-li stejně důležité)



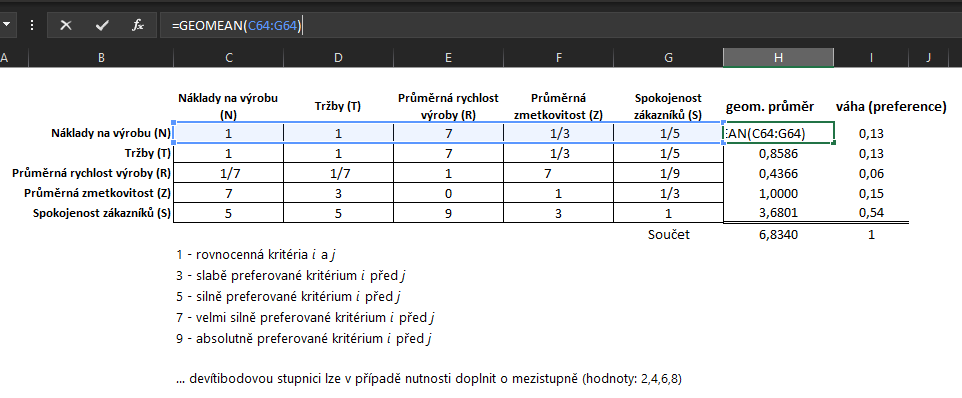
Vypočtu počet upřednostnění a přidám jedničku, váhy spočítám opět stejně (úplně pravý sloupec)



**Saatyho metoda**

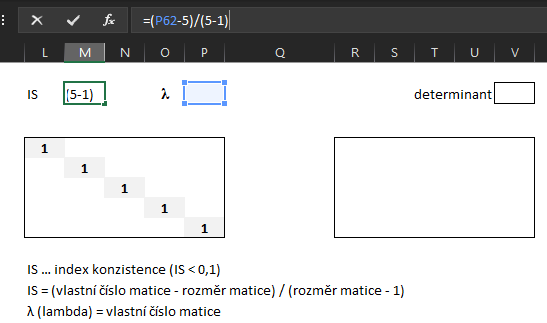
Vytvořím saatyho matici, porovnávám **i - řádek** a **j - sloupec** na stupnici **1/9 - 9 (z hlavy, odhad z kriteriální matice)**

Následně udělám **geometrický průměr (GEOMEAN V EXCELU)** a **vypočtu váhu** stejně jako u předchozích případů (**konec strany 39)**

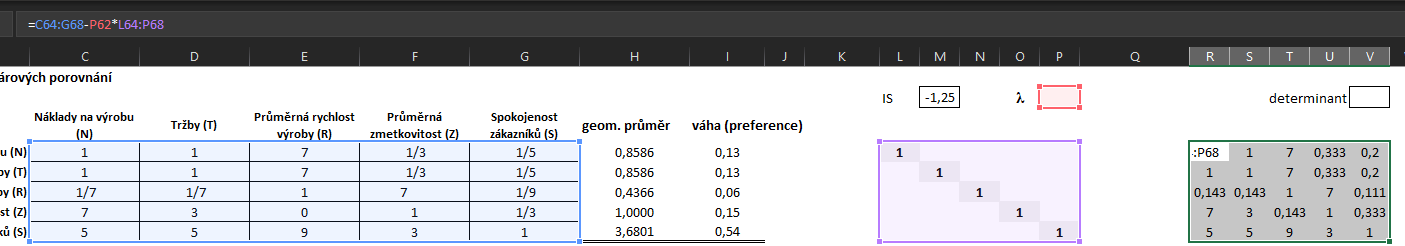


(zde je [4,1] a [4,3] špatně )

Vytvořím si tyto věci dle posaných vzorečků



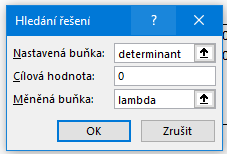
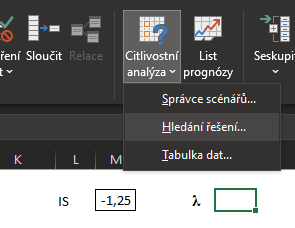
**Od saatyho matice odečtu lambda krát jednotková matice** (viz. obrázek)



do políčka determinantu dám fci. DETERMINANT(obsah matice)

Lambdu nastavím na nějaké náhodné kladné číslo, třeba 20

Kliknu do políčka lamby, zvolím v excelu záložku data a následně citlivostní analýzu a provedu dle obrázků



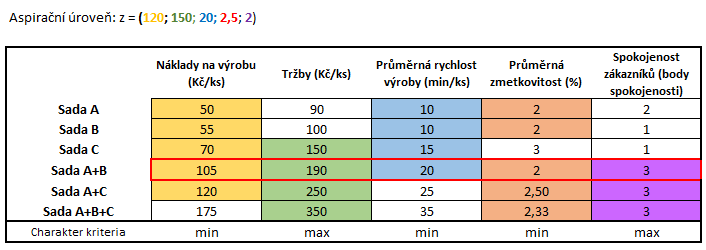
Pokud mi vyjde IS menší jak **0.1** - mám to správně -> jedná se o **konzistentní rozhodování**

**Stanovení kompromisní varianty (kompromisního řešení**)

**Metoda aspirační úrovně (konjuktivně)**

Je dán vektor aspirační úrovně a já ponechám jen to řešení, co splňují všechny složky viz. obrázek  
pokud nám vyjde míň nebo víc řešení, tak buď zpřísním nebo zjemním aspirační úroveň

(data máme z kriteriální matice)



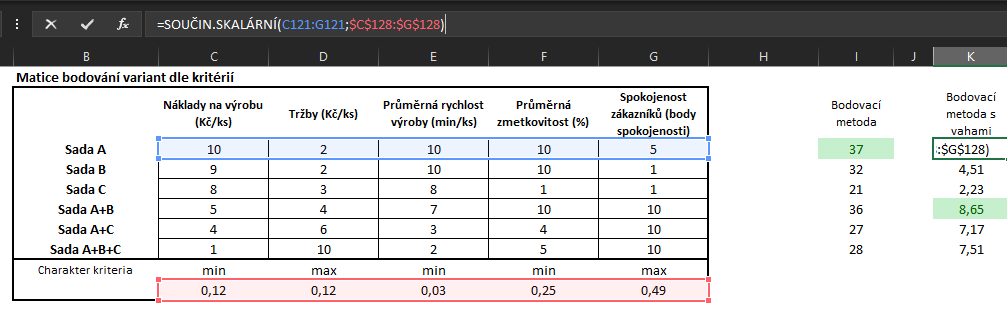
**Bodovací metoda a Bodovací metoda s vahami**

Z kriteriální matice převedu **dle sebe +-** na škálu od 1-10

Váhy vezmu se **Saatyho matice**

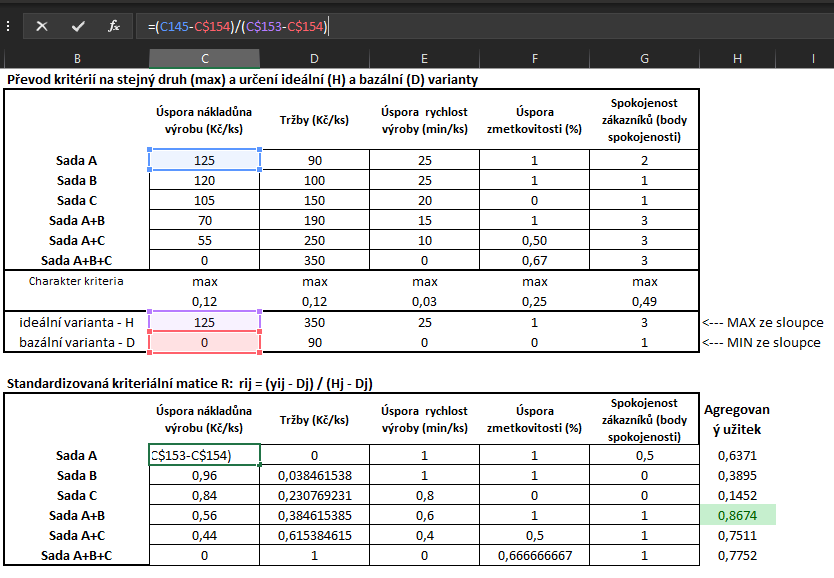
**Bodovací metoda**, kde **sčítám body každé varianty**  (SUMA(vybraný řádek))

**BM s vahami**, **skalární součin řádku a vah, dle něj rozhoduji o optimální variantě**



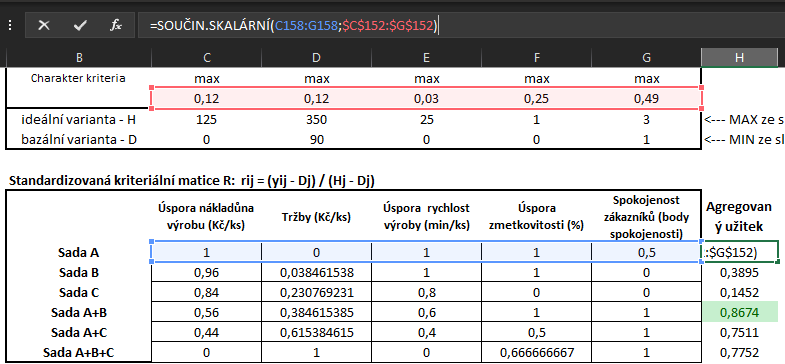
**Metoda váženého součtu**

Převedu z **kriterialni matice** opět všechen charakter na maximalizacni a idealni a bazalni variantu vezmu tak, že vezmu MAX a MIN z daného sloupce



Standardizovanou matici spočítám dle vzorečku

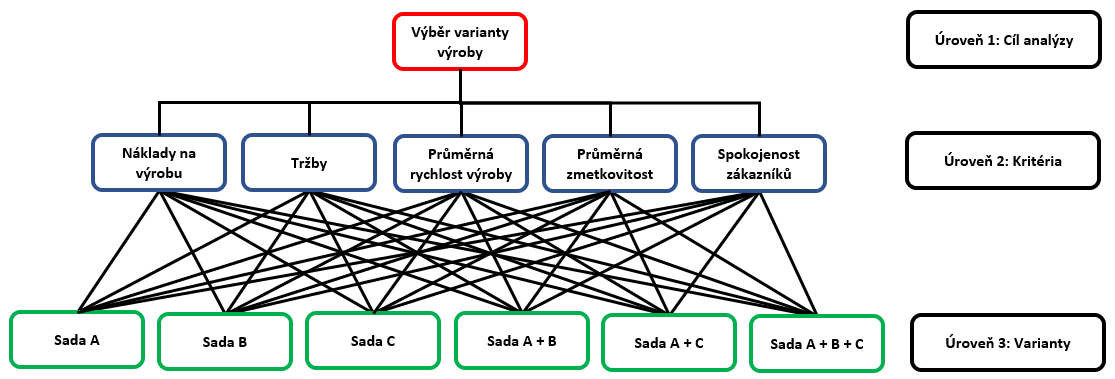
**Agregovaný užitek** je skalární součin řádku a **vah ze saatyho matice**

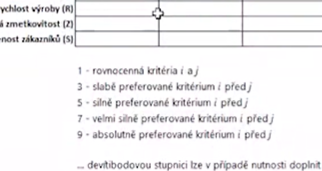
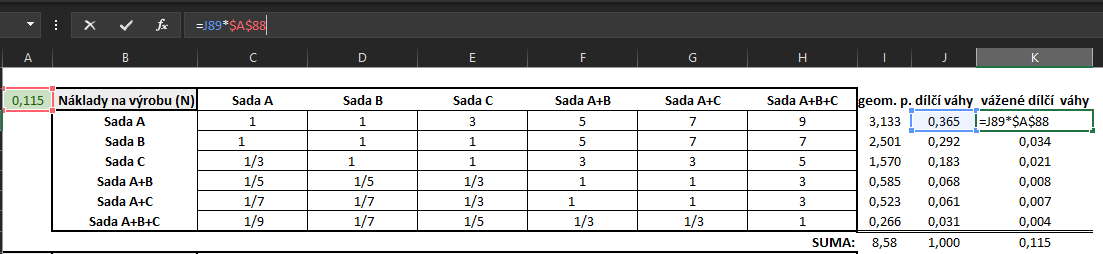
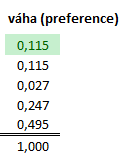
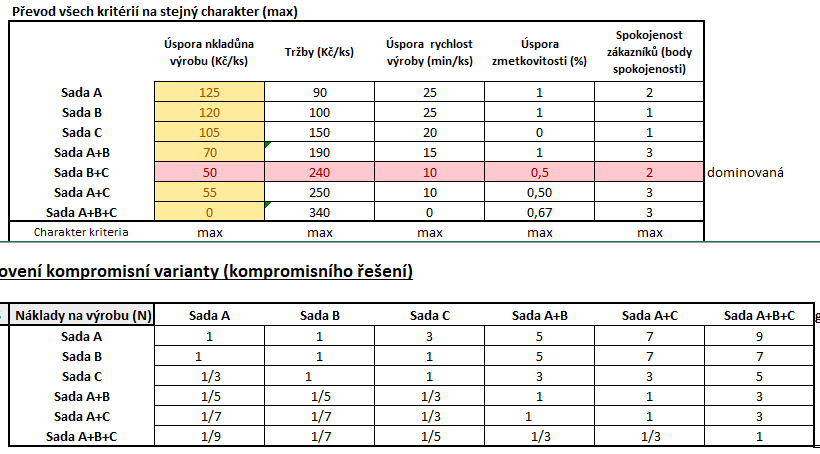


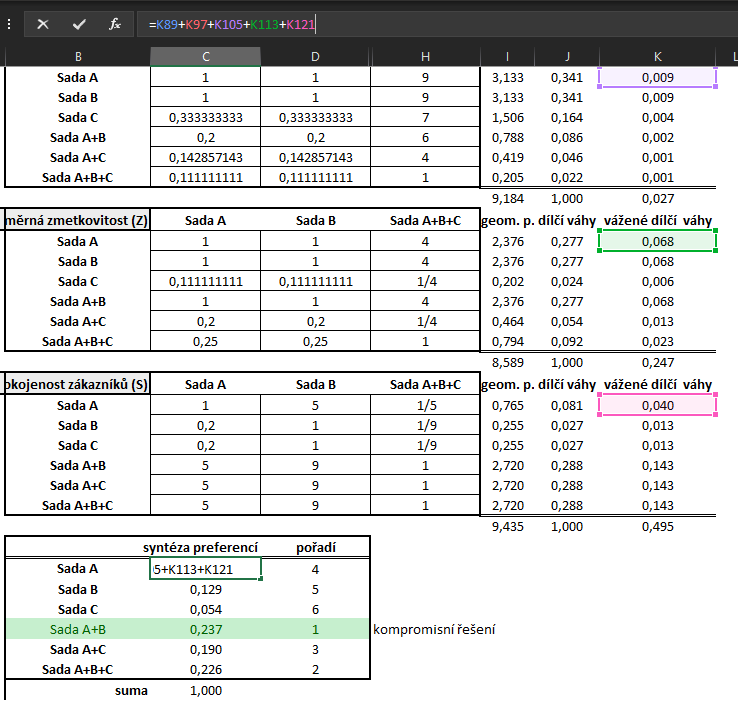
Analytický Hiearchický Proces

Mnoho kroků stejných jako v Teorii rozhodování, AHP slouží ke stanovení kompromisní varianty **krok 1-2 není součástí AHP** ale musí se před AHP provést

**Hiearchická struktura procesu**

****

1. Kvantifikace a určení charakteru kritérií **viz. strana 37**
2. Posouzení dominance variant a stanovení paretovských variant **viz strana 38**
3. Nakreslit hiearchickou strukturu AHP
4. Stanovení vah kritérií (jen saatyho matice) **str. 40 - 42**
5. Preference variant dle kritérií  
   Pro každé **kritérium** provedu Saatyho matici viz. str 41-42, **nepočítám s dominující sadou**  
     
   K Saatyho matici přidám ještě **sloupec vážených dílčích vah** a **buňku s váhou**z první Saatyho matice vezmu **váhu kritéria  
     
     
   <- Váha ze Saatyho matice pro dané kritérium**Zde určuji dle jednotlivého **kritéria**

Syntéza preferencí  
Ze všech **Saatyho matic kritérií** vezmu **váženou dílčí váhu pro danou sadu** a následně funkci rozkopíruju - pro kontrolu **součet by měl být 1**  
Z těchto možností určím pořadí a **kompromisní řešení je první  
**

**Teorie pojmy:**

**Simplex:**

* doplňková proměnná, pomocná proměnná, účelová fce, kanonický tvar
* alternativní, optimální, suboptimální řešení
* omez. fce (kapacitní, požadavková, určení)
* Frobeinova věta

**Dopravní úloha:**

* indexová metoda, vogelova metoda, MODI metoda
* dantzigův obvod

**Okružní problém:**

* Mayerova metoda, vogelova metoda, nejbližšího souseda

**Teorie her:**

* kooperativní/nekooperativní, s konstatním /nekonstantním součtem
* nekonstantní součet - dvoumaticová hra (nashovy rovnovážné body)
* konstantní součet - horní/dolní cena hry (sedlový bod)
* transformovaný/netransformovaný model
* MinMax (horní cena hry), MaxMin (spodní cena hry)

**Teorie rozhodování**

* matice zisku/výplat/ztrát
* dominance dle pravděpodobnosti, rozhodovací strom,Bernoulli-Laplacovo pravidlo Max průměr, hurwiczovo, savageovo MinMax, EMV, EOL
* Optimista MaxMax, Waldovo kritérium - pesimista Maxmin

**Vícekriteriální optimalizační model:**

* matice variant
* posouzení dominance/paretovské varianty, metoda pořadí/bodovací/bodovací s vahami/váženého součtu, fullerův trojůhelník, saatyho metoda, metoda aspirační úrovně
* agregovaný užitek
* bazální/ideální varianta
* **AHP**

**-** preference dle kritérií, kompromisní (nejlepší) řešení, syntéza preferencí

**Vektorovy prostor**

vektorový prostor - množina vektorů, množina skalárů, Operace mezi vektory, Dimenzí, počet nezávislých vektorů = počet dimenzí

**Definujte generátory vektorového prostoru** a bázi vektorového prostoru.

* dimenze, báze, bazický vektor

**Nominální** proměnná je taková, o jejíž dvou hodnotách můžeme pouze říci, zda jsou stejné či různé

**Ordinální** (**pořadová**), u jejíž dvou hodnot můžeme navíc určit pořadí

**Kvantitativní/numerické** - zahrnují jak nominální, tak ordinální a lze určit i o kolik je rozdíl