Implementace simplexové metody v oblasti multikriteriálního hodnocení variant metodou analýzy datových obalů - DEA

Simplex method implementation in multicriteria evaluation by the use of data envelopment analysis method - DEA

Bakalářská práce

Autor práce: Leoš Dobiáš

Vedoucí práce: Ing. Ludvík Friebel, Ph.D.

Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta

Katedra Informatiky

2008

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně, pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské/diplomové práce, a to v nezkrácené podobě, pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Budějovicích dne 24. 4. 2008

Anotace

Cílem této práce bylo vytvoření počítačového programu pro podporu výuky rozhodovacích modelů na Ekonomické Fakultě JU. Aplikace řeší výpočty modelů metody DEA a je naprogramována v jazyce MS C# 3.5 a XAML. Výpočetní jádro je koncipováno jako samostatná knihovna. Při implementaci patřičných modelů v budoucnu umožní i výpočty dalších metod lineárního programování.

Abstract

The aim of this dissertation has been to create a computer program to support the classes of decision-making models at the Faculty of Economy, University of South Bohemia. The application solves calculations of DEA method models and has been programmed in MS C# 3.5 and XAML languages. The computation core is outlined as an autonomous library. The implementation of appropriate models will also enable calculations of other linear programming methods in the future.

Poděkování

Rád bych tímto poděkoval následujícím osobám:

RNDr. Jaroslavu Ichovi, který u mne na počátku studia vzbudil zájem o objektové programování.

Ing. Ludvíku Friebelovi, Ph.D. a Ing. Janě Friebelové, Ph.D. za cenné konzultace při studiu teoretických východisek aplikace.

Spolupracovnicím, Jaroslavě Pudivítrové a Olze Dobré, za pomoc při typografické úpravě tohoto dokumentu.

Obsah

1. Úvod

S radostí jsem přivítal návrh Ing. Ludvíka Friebela, Ph.D., abych jako bakalářskou práci vytvořil aplikaci pro podporu výuky rozhodovacích modelů na Ekonomické fakultě Jihočeské univerzity.

Důvodem tohoto zadání je absence nekomerčního software pro výpočty modelů multikriteriálního rozhodování. Na Ekonomické fakultě je pouze jedna licence programu Banxia Frontier Analyst, na počítačových učebnách žádný software tohoto druhu není.

Mým cílem bylo naprogramování takové aplikace. Při její tvorbě jsem kladl důraz na další možnou rozšiřitelnost. Výsledný systém lze tedy rozšířit pro výpočet dalších optimalizačních metod naprogramováním jejich modelů. Systém lze rovněž rozšiřovat o další uživatelská rozhraní.

2. Teoretická východiska metody datových obalů DEA

(Použité zdroje [1], [2], [3])

Modely datových obalů slouží pro hodnocení technické efektivity produkčních jednotek systému na základě velikosti vstupů a výstupů. Protože vstupů a výstupů může být více druhů, řadí se DEA mezi metody vícekriteriálního rozhodování.

2.1. Optimalizační úlohy

Simplexová metoda je jednou z metod řešení optimalizačních úloh, které jsou definovány následujícím způsobem:

- Existuje univerzum U všech potenciálních řešení. To je dáno jako vektorový prostor s jednotlivými proměnnými jako souřadnicemi.
- Omezující podmínky vymezují podmnožinu všech přípustných řešení $P \subseteq U$
- Hodnota každého možného řešení je určena účelovou funkcí $\eta : U \to R$
- Hodnotu účelové funkce dle zadání buď maximalizujeme, nebo minimalizujeme

Cílem řešení optimalizační úlohy je nalezení takového

$$\vec{x} \in P : \eta(x) = \max/\min \eta(\vec{z}),\ \vec{z} \in P \qquad (1)$$

Grafické řešení lze použít pouze u nejjednodušších úloh se dvěma proměnnými, kde jednotlivé nerovnice určují poloroviny v Euklidovské rovině, jejichž průnikem je množina všech přípustných řešení. Účelová funkce je v tomto případě vyjádřena systémem rovnoběžek odpovídajících jednotlivým hodnotám přípustných řešení. Optimálním řešením je průnik množiny přípustných řešení s rovnoběžkou účelové funkce s co nejvyšší / nejnižší hodnotou.

V případě tří proměnných nerovnice určují poloprostory v trojrozměrném prostoru. Obecně pro úlohu s n proměnnými si lze množinu přípustných řešení vyjádřit jako n-dimensionální polyedr.

Pojem polyedru je třeba zavést pro pochopení dalšího postupu. Nebudu se zde do hloubky věnovat matematickému výkladu, který je mimo rámec mojí práce, spokojím se s jednoduchou definicí založenou na geometrické představě.

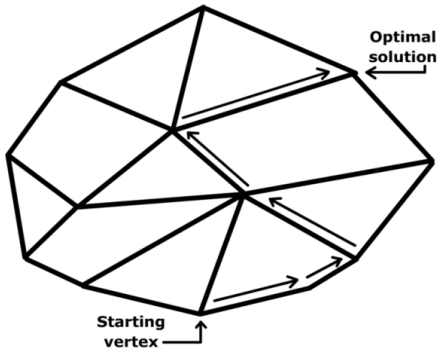
Polyedr je konvexní a uzavřená množina tvořená průnikem konečně mnoha poloprostorů.

V našem případě, kdy polyedr je tvořen průnikem poloprostorů tvořených nerovnicemi optimalizační úlohy, vytváří tento konvexní obal množiny všech přípustných řešení.

Pokud existuje řešení optimalizační úlohy, leží vždy na vrcholu, případně na celé hraně polyedru.

2.2. Simplexová metoda

Pro řešení úloh lineární optimalizace neboli lineárního programování (LP) se používá simplexová metoda. Z geometrického pohledu simplexová metoda přechází po hranách polyedru optimalizační úlohy, až najde optimální řešení.



Obrázek 1: Grafická interpretace simplexové metody (převzato z [4])

Praktická realizace představuje vyhledávání optimálního bazického řešení v matici simplexové tabulky.

2.2.1. Realizace výpočtu simplexové metody

Pro počítačově orientovaný zápis úlohy lineárního programování i průběhu výpočtu simplexové metody se používá simplexová tabulka.

Algoritmus 1 – výpočet, jednofázová metoda

1. Úlohu LP:

$$\max \vec{c} \cdot \vec{x}:$$
$$A \cdot \vec{x} = \vec{b} \qquad (2)$$
$$\vec{x} \geq 0$$

převedeme do kanonického tvaru:

$$\begin{aligned} &\min x_0 \\ x_0 + \vec{c} \cdot \vec{x} &= 0 \\ A \cdot \vec{x} &= \vec{b} \\ \vec{x} &\geq 0 \end{aligned} \qquad (3)$$

2. Soustava se potom převede do formy simplexové tabulky:

$$\begin{pmatrix} 1 & \vec{c} \\ \vec{0} & A \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ \vec{x} \end{pmatrix} = \begin{pmatrix} -b_0 \\ \vec{b} \end{pmatrix} \quad , \qquad (4)$$

která zapisuje koeficienty soustavy lineárních rovnic.

Horní (nultý) řádek tabulky nazýváme účelovým řádkem, jeho prvky jsou redukované ceny proměnných, sloupec b nazýváme pravou stranou a zbytek tabulky nazýváme levou stranou. V dalším textu již budeme tabulku zapisovat bez nultého účelového sloupce.

3. Získáme výchozí přípustné bazické řešení. K tomu potřebujeme tabulku v přípustném jednotkovém tvaru. Provedeme tedy následující úpravy:
   a. Všechny rovnice se zápornou pravou stranou vynásobíme −1. (Aby bylo výchozí bazické řešení přípustné.)
   b. Pro každý chybějící jednotkový vektor $\vec{e_i}$ v tabulce na levé straně přidáme umělou proměnnou $u_i \geq 0$ s cenou −ν, kde ν je velmi velká konstanta, formálně $\nu = \infty$.
   c. Zapíšeme vzniklou výchozí tabulku:

$$\left[\begin{array}{c|c} \vec{c} \ \ldots \ 0 \ldots -\nu \ldots & 0 \\ \hline \underbrace{\boldsymbol{A}' \mid \boldsymbol{I}}_{\boldsymbol{A}^u} & \vec{b}' \end{array}\right] \tag{5}$$

   d. Upravíme tabulku do jednotkového tvaru, aby i účelový řádek obsahoval redukované ceny 0 ve sloupcích jednotkové podmatice ***I***: Přičteme násobky řádků příslušných k jednotlivým jednotkovým vektorům ***I*** k účelovému řádku.
4. Jsou-li všechna čísla v účelovém řádku tabulky nekladná, máme optimální řešení. Nekladné redukované ceny všech proměnných znamenají, že zvětšováním žádné z nebazických proměnných již nelze zvýšit účelovou funkci. Pokud však stále ještě zůstává některá umělá

proměnná s nenulovou hodnotou, původní úloha nemá žádné přípustné řešení.

5. Je-li v tabulce sloupec ***s*** takový, že $c_s = a_{0s} > 0$ a $a_{is} \leq 0$ pro všechna $i > 0$, úloha nemá optimální řešení z důvodu neomezenosti.
6. Přejdeme k sousední bázi:

   a. Sloupcové pravidlo vybere sloupce s nejvyšší kladnou hodnotou redukované ceny v účelovém řádku, tj. ve směru nejvyššího přírůstku účelové funkce.

   b. Řádkové pravidlo vybere řádek $i > 0$, s nejmenší hodnotou podílu b_i / a_{is} pro $a_{is} > 0$, což je nutné pro zachování nezápornosti pravé strany.

   c. Přejdeme k sousední bázi aplikaci tzv. pivotu na prvku a_{is}: Pivotace prvku je vlastně jednou fází Gaussovy eliminace v matici, která řádkovými operacemi „vytvoří" jednotkový (sloupcový) vektor na této pozici.

      - Pro každé $j \neq i$, od *j*-tého řádku odečteme a_{ji} / a_{is}-násobek *i*-tého,
      - *i*-tý řádek vydělíme číslem a_{is}

      (Nyní v nové bázi sloupec ***s*** nahradí sloupec původního jednotkového vektoru $\vec{e}_i$.)

      Aplikací řádkových maticových operací na simplexové tabulce nezměníme zapsanou optimalizační úlohu a nová báze tabulky nemá horší účelovou hodnotu.
7. Vrátíme se v cyklu do bodu 3.

2.2.2. Dvoufázová simplexová metoda

Jedná se o druhý způsob odstranění umělých proměnných.

Algoritmus 2 – Implementace dvoufázové simplexové metody

Dvoufázová simplexová metoda (zapsaná simplexovou tabulkou) je založena na postupu Algoritmu 1 jednofázové metody s následujícími vylepšeními.

1. Vyjdeme z (už známého) kanonického tvaru úlohy v redukovaném zápisu:

$$\begin{aligned} &\min x_0 \\ x_0 + \vec{c} \cdot \vec{x} &= 0 \\ A \cdot \vec{x} &= \vec{b} \\ \vec{x} &\geq 0 \end{aligned} \tag{6}$$

2. Jako v Algoritmu 1, bod 3, po případném přidání umělých proměnných zapíšeme výchozí simplexovou tabulku v jednotkovém tvaru

$$\left[\begin{array}{c|c|c} \vec{c} \ \ldots \ 0 \ldots -\nu \ldots & & 0 \\ \hline \boldsymbol{A'} & \boldsymbol{I} & \vec{b'} \\ \underbrace{\qquad\qquad}_{\boldsymbol{A^u}} & & \end{array}\right], \tag{7}$$

která vyjadřuje vztahy

$$\begin{aligned} &x_0 + \vec{c} \cdot \vec{x} = 0 \\ &A^u \cdot \left(\vec{x}, \vec{u}\right)^T = \vec{b'} \\ &\vec{x}, \vec{u} \geq 0 \end{aligned} \tag{8}$$

Nyní však pro první fázi k ocenění umělých proměnných zavedeme novou účelovou funkci a pro $\overrightarrow{c^u} = \left(1, \vec{c}, \vec{0}\right)$ novou úlohu zapíšeme jako

$$\begin{aligned} &\max - u_1 - \ldots - u_k \\ &\overrightarrow{c^u} \cdot \left(x_0, \vec{x}, \vec{u}\right)^T = 0 \\ &A^u \cdot \left(\vec{x}, \vec{u}\right)^T = \vec{b'} \\ &\vec{x}, \vec{u} \geq 0 \end{aligned} \qquad (9)$$

V zápisu simplexovou tabulkou tato úloha zní následovně.

0…0	-1…-1	0
$\vec{c}$	0…0	0
A^u		$\vec{b'}$

(9)

Nakonec ještě musíme tabulku upravit tak, aby účelový řádek obsahoval 0 ve sloupcích umělých proměnných.

2. - 5. Postupujeme v těchto bodech podle Algoritmu 1, ale máme na paměti, že i původní účelový řádek je vyloučen z působnosti řádkového pravidla, třebaže jinak je pokládán za běžný řádek tabulky.

6. Pokud optimální řešení první fáze má kladnou hodnotu (tedy $\vec{u} \neq \vec{0}$), pak původní úloha nemá žádné přípustné řešení.
7. Jinak z výsledné tabulky první fáze vypustíme umělý účelový řádek i všechny sloupce uměle proměnných. Pokračujeme ve výpočtu druhé fáze metody už známým způsobem od bodu 2 Algoritmu 2.

2.3. Modely DEA

Cílem této metody je rozdělení zkoumaných objektů na efektivní a neefektivní podle velikosti spotřebovávaných zdrojů a množství vyráběné produkce nebo jiného typu výstupů. DEA porovnává jednotky vzhledem k nejlepším jednotkám. Jedná se o metodu odhadu produkční funkce založenou na teorii lineárního programování. Modely DEA vycházejí z Farrelova modelu pro měření efektivity jednotek s jedním vstupem a jedním výstupem, který rozšířili Charnes, Cooper a Rhodes (CCR) a Banker, Charnes a Cooper (BCC).

Vstupní údaje můžeme zapsat do tabulky, která má charakter kriteriální matice (sloupce vstupů odpovídají hodnocení podle minimalizačního kritéria a sloupce výstupů podle maximalizačního kritéria). Je akceptována kompenzace (vyšší výstupy potřebují více vstupů při zachování efektivity spotřeby).

Předpokládejme, že zkoumaný objekt zahrnuje p jednotek, jsou označeny S_1, S_2, …, S_p. Každá z nich spotřebovává m vstupů na produkci n výstupů. Potom x_{ik} je množství spotřebovávaného vstupu k-tou jednotkou a y_{jk} je množství výstupu produkovaného k-tou jednotkou. Vstupy a výstupy lze zapsat do přehledné tabulky:

	Vstupy				Výstupy			
	X_1	X_2	…	X_m	Y_1	Y_2	…	Y_n
S_1	x_{11}	x_{21}	…	x_{m1}	y_{11}	y_{21}	…	y_{n1}
S_2	x_{12}	x_{22}	…	x_{m2}	y_{12}	y_{22}	…	y_{n2}
…	…	…	…	…	…	…	…	…
S_p	x_{1p}	x_{2p}	…	x_{mp}	y_{1p}	y_{2p}	…	y_{np}

Tabulka 1: Vstupy a výstupy pro metodu DEA

Jednotka je efektivní, pokud spotřebovává malé množství vstupů na velké množství výstupů. Efektivita jednotlivých jednotek je dána vztahem:

$$efektivita = \frac{výstup}{vstup}. \tag{10}$$

Neefektivní jednotky by měly snížit množství vstupů nebo zvýšit množství výstupů. V případě více spotřebovávaných vstupů na produkci více výstupů se používá relativní míra efektivity:

$$efektivita = \frac{vážená\ suma\ výstupů}{vážená\ suma\ vstupů}, \tag{11}$$

což lze vyjádřit vztahem:

$$\Phi_k = \frac{\sum_{j=1}^{n} u_j y_{jk}}{\sum_{i=1}^{m} v_i x_{ik}}, k = 1,2,...,p, \tag{12}$$

kde u_i a v_j jsou jednotné váhy vstupů a výstupů pro všechny hodnocené jednotky.

Vzhledem k tomu, že každé středisko je jinak zaměřené, lze uvažovat váhy odděleně pro každé středisko. Tyto váhy nejsou odvozené od ceny, ale spíše od používané technologie v jednotlivých střediscích. Z tohoto důvodu se používá termín relativní technická efektivita, kterou vyjadřuje následující vztah:

$$\Phi_k = \frac{\sum_{j=1}^{n} u_{jk} y_{jk}}{\sum_{i=1}^{m} v_{ik} x_{ik}}, k = 1,2,...,p, \tag{13}$$

kde u_{ik} a v_{jk} jsou individuální váhy vstupů a výstupů pro jednotlivé jednotky.

Hypotetická (virtuální) jednotka je charakterizována jako vážený průměr efektivních jednotek (peer jednotek). Tato jednotka vyjadřuje spotřebu vstupů a produkci výstupů pro skutečnou neefektivní jednotku, která produkuje méně výstupů nebo spotřebovává více vstupů než její virtuální jednotka.

Modely DEA hledají individuální váhy pro jednotlivé hodnocené jednotky. Tyto váhy jsou hledány tak, aby byla maximalizována efektivita jednotek.

2.3.1. CCR vstupově orientovaný model

U modelů CCR předpokládejme konstantní výnos z rozsahu (stejná změna vstupů je provázena stejnou změnou výstupů). Pomocí tohoto modelu lze určit, jaké má být množství vstupů, aby se neefektivní jednotka stala efektivní. Koeficient technické efektivity je definován jako poměr vážené sumy výstupů a vážené sumy vstupů. Váhy musí být stanoveny tak, aby koeficient technické efektivity byl z intervalu $\langle 0;1 \rangle$. Jednotka s koeficientem technické efektivity rovným 1 je efektivní, koeficient menší než 1 ukazuje na neefektivní jednotku a určuje nutnost změny vstupů k zajištění efektivity jednotky.

Model CCR stanoví váhy vstupů a výstupů pro každou jednotku tak, aby jednotka maximalizovala svůj koeficient technické efektivity a byly splněny podmínky:

Váhy nesmí být záporné.

Při použití tohoto souboru vah nesmí žádný koeficient technické efektivity být větší než jedna.

Neznámými jsou v tomto modelu váhy přidělené vstupu *i* a váhy přidělené výstupu *j* jednotkou *k*. Váhy jsou určovány individuálně, proto je nutno vyřešit *p* modelů.

Matematický model pro jednotku H je následující:

$$\Phi_H = \frac{\sum_{j=1}^{n} u_{jH} y_{jH}}{\sum_{i=1}^{m} v_{iH} x_{iH}} \rightarrow \max, \tag{14}$$

za podmínek

$$\begin{aligned} &\frac{\sum_{j=1}^{n} u_{jH} y_{jk}}{\sum_{i=1}^{m} v_{iH} x_{ik}} \leq 1,\ k = 1,2,\ldots, p, \\ &u_{jH} \geq 0,\ v_{iH} \geq 0. \end{aligned} \tag{15}$$

Model lze upravit na lineární pomocí Charnes Cooperovy transformace:

$$\Phi_k = \sum_{j=1}^{n} u_{jH} y_{jH} \rightarrow \max, \tag{16}$$

za podmínek

$$\begin{aligned} &\sum_{i=1}^{m} v_{iH} x_{iH} = 1, \\ &-\sum_{i=1}^{m} v_{iH} x_{ik} + \sum_{j=1}^{n} u_{jH} y_{jk} \leq 0, \\ &u_{jH} \geq 0,\ j = 1,2,\ldots,n, \\ &v_{iH} \geq 0,\ i = 1,2,\ldots,m. \end{aligned} \tag{17}$$

Když sestavíme k tomuto modelu duální model, zjistíme, které jednotky tvoří množinu peer jednotek neefektivní jednotky H a zároveň získáme koeficienty λ_{kH} kombinace peer jednotek, které tvoří virtuální efektivní jednotku k jednotce H.

Duální model má tvar:

$z_H \rightarrow \min$

za podmínek

$$\begin{aligned} & x_{iH} z_H - \sum_{k=1}^{p} \lambda_{kH} x_{ik} \geq 0,\ i = 1,2,...,m, \\ & \sum_{k=1}^{p} \lambda_{kH} y_{jk} \geq y_{jH},\ j = 1,2,...,n, \\ & \lambda_{kH} \geq 0,\ k = 1,2,...,p. \end{aligned} \tag{18}$$

Velikost vstupů a výstupů virtuální jednotky lze spočítat jako kombinaci vstupů a výstupů peer jednotek

$$\begin{aligned} & x'_{iH} = \sum_{k=1}^{p} \lambda_{kH} x_{ik},\ i = 1,2,...,m, \\ & y'_{jH} = \sum_{k=1}^{p} \lambda_{kH} y_{jk},\ j = 1,2,...,n. \end{aligned} \tag{19}$$

Příklad 1 – hodnocení produkčních jednotek metodou DEA - vstupově orientovaný model (převzato se svolením autora z [1])

Zadání

Obchodní řetězec má sedm poboček, přičemž každá z těchto poboček je charakterizována počtem zaměstnanců, režijními náklady v tisících Kč, počtem obsloužených zákazníků za hodinu a denními tržbami v 10 tisících Kč. Potřebné údaje jsou v následující tabulce.

Pobočka	A	B	C	D	E	F	G
Zaměstnanci	4	7	8	4	2	5	6
Režijní náklady	3	3	1	2	4	2	4
Zákazníci	1	2	3	4	4	5	6
Tržby	5	7	4	3	6	5	2

Tabulka 2: Vstupy a výstupy produkčních jednotek

Řešení

Vzhledem k tomu, že budeme hodnotit sedm poboček, je nutné sestavit sedm modelů. Primární model pro první pobočku – pobočku A (1):

$$
\begin{aligned}
e_1 = u_{11} + 5u_{21} &\to \max \\
4v_{11} + 3v_{21} &= 1 \\
-4v_{11} - 3v_{21} + u_{11} + 5u_{21} &\le 0 \\
-7v_{11} - 3v_{21} + 2u_{11} + 7u_{21} &\le 0 \\
-8v_{11} - v_{21} + 3u_{11} + 4u_{21} &\le 0 \\
-4v_{11} - 2v_{21} + 4u_{11} + 3u_{21} &\le 0 \\
-2v_{11} - 4v_{21} + 4u_{11} + 6u_{21} &\le 0 \\
-5v_{11} - 2v_{21} + 5u_{11} + 5u_{21} &\le 0 \\
-6v_{11} - 4v_{21} + 6u_{11} + 2u_{21} &\le 0 \\
u_{j1} &\ge 0,\ j = 1, 2, \\
v_{i1} &\ge 0, i = 1, 2.
\end{aligned}
\tag{20}
$$

Řešení této soustavy nerovnic spadá do oblasti úloh lineárního programování řešitelných pomocí simplexové metody popisované v předchozí kapitole.

Výsledky

Proměnná	Hodnota
e_1	0,869565189
v_{11}	0,086956523
v_{21}	0,217391297
u_{11}	0
u_{21}	0,173913047

Tabulka 3: Výsledky pro primární model

Z výsledku je patrné, že jednotka A (první) není efektivní, neboť hodnota e_1 je menší než jedna. Proto přistoupíme k sestavení duálně sdruženého modelu, jehož vyřešením získáme vhodnou kombinaci vzorových jednotek pro dosažení efektivity neefektivní jednotky A. Duální model pro první jednotku:

$$
\begin{aligned}
& z_1 \rightarrow \min \\
4z_1 - 4\lambda_{11} - 7\lambda_{21} - 8\lambda_{31} - 4\lambda_{41} - 2\lambda_{51} - 5\lambda_{61} - 6\lambda_{71} &\geq 0 \\
3z_1 - 3\lambda_{11} - 3\lambda_{21} - \lambda_{31} - 2\lambda_{41} - 4\lambda_{51} - 2\lambda_{61} - 4\lambda_{71} &\geq 0 \\
\lambda_{11} + 2\lambda_{21} + 3\lambda_{31} + 4\lambda_{41} + 4\lambda_{51} + 5\lambda_{61} + 6\lambda_{71} &\geq 1 \\
5\lambda_{11} + 7\lambda_{21} + 4\lambda_{31} + 3\lambda_{41} + 6\lambda_{51} + 5\lambda_{61} + 2\lambda_{71} &\geq 5 \\
& \lambda_{k1} \geq 0,\ k = 1, 2, ..., 7
\end{aligned}
\qquad (20)
$$

Pro řešení duálního modelu je opět použita simplexová metoda. Výsledky pro jednotku A jsou zobrazeny v tabulce 4.

Proměnná	Hodnota
z_1	0,869565189
z_1	0,869565189
λ_{11}	0
λ_{21}	0
λ_{31}	0
λ_{41}	0
λ_{51}	0,380434781
λ_{61}	0,543478251
λ_{71}	0

Tabulka 4: Výsledky duálního modelu

Z výsledku vyplývá, že vzorovými (peer) jednotkami pro jednotku A jsou jednotky pátá (E) a šestá (F), neboť hodnoty proměnných λ_{51} a λ_{61} jsou nenulové. Proměnná λ_{51} se vztahuje k páté jednotce (E) – první index je 5, proměnná λ_{61} se vztahuje k šesté jednotce (F) – první index je 6. Známe tedy koeficienty kombinace. Nyní musíme vstupy efektivních jednotek E a F pronásobit koeficienty a pro každý vstup zvlášť tyto výsledky sečíst, abychom získali optimální velikost vstupů pro jednotku A.

Konkrétně pro první vstup:

$$x'_{11} = \lambda_{51}x_{15} + \lambda_{61}x_{16} = 0{,}380434781 \cdot 2 + 0{,}543478251 \cdot 5 = 3{,}478260815$$

Pro druhý vstup:

$$x'_{21} = \lambda_{52}x_{25} + \lambda_{62}x_{26} = 0{,}380434781 \cdot 4 + 0{,}543478251 \cdot 2 = 2{,}608695626$$

Jednotka A by měla snížit svůj první vstup – počet zaměstnanců z původních 4 na 3,5 (jednomu pracovníkovi snížit úvazek zhruba na polovinu). Druhý vstup – režijní náklady by měla snížit z původních 3.000 Kč na zhruba 2.600 Kč. Potom bude jednotka A efektivní.

Stejně jako pro jednotku A je nutné sestavit modely pro všechny ostatní jednotky a vyřešit je stejným způsobem. V tabulce 4.2 jsou uvedeny výsledky pro všechny jednotky. Neuvádíme hodnotu účelové funkce duálního modelu, neboť je shodná s hodnotou účelové funkce primárního modelu.

	Primární model					Duální model						
	e_k	v_{1i}	v_{2i}	u_{1j}	u_{2j}	λ_{1k}	λ_{2k}	λ_{3k}	λ_{4k}	λ_{5k}	λ_{6k}	λ_{7k}
A	0,870	0,087	0,217	0	0,174	0	0	0	0	0,380	0,544	0
B	0,966	0,069	0,172	0	0,138	0	0	0	0	0,060	1,328	0
C	1	0,068	0,455	0	0,25	0	0	1	0	0	0	0
D	0,941	0,177	0,147	0,235	0	0	0	0	0	0,118	0,706	0
E	1	0,083	0,208	0	0,167	0	0	0	0	1	0	0
F	1	0,15	0,125	0,2	0	0	0	0	0	0	1	0
G	0,857	0,107	0,089	0,143	0	0	0	0	0	0,429	0,857	0

Tabulka 5: Výsledky primárních a duálních modelů

2.3.2. CCR výstupově orientovaný model

Vychází ze stejných předpokladů jako vstupově orientovaný model. Určuje takové množství výstupů, aby se neefektivní jednotka stala efektivní. Zde je koeficient technické efektivity určen jako poměr vážené sumy vstupů a vážené sumy výstupů. Váhy musí být stanoveny tak, aby hodnota tohoto koeficientu byla větší nebo rovna 1.

Výstupově orientovaný model CCR stanoví pro každou jednotku individuální váhy vstupů a výstupů tak, aby jednotka minimalizovala svůj koeficient technické efektivity a přitom byly splněny podmínky:

Při použití souboru vah pro všechny jednotky nesmí žádný koeficient technické efektivity být menší než 1.

Výstupově orientovaný model pro jednotku H

$$\Phi_H = \frac{\sum_{i=1}^{m} v_{iH} x_{iH}}{\sum_{j=1}^{n} u_{jH} y_{jH}} \rightarrow \min \tag{21}$$

za podmínek

$$\begin{aligned} &\frac{\sum_{i=1}^{m} v_{iH} x_{iH}}{\sum_{j=1}^{n} u_{jH} y_{jH}} \geq 1, k = 1,2,..., p, \\ &u_{jH} \geq 0,\ j = 1,2,..., n, \\ &v_{iH} \geq 0, i = 1,2,..., m. \end{aligned} \tag{22}$$

Stejně jako u vstupově orientovaného modelu lze zafixovat jmenovatele na hodnotu 1:

$$\Phi_H = \sum_{i=1}^{m} v_{iH} x_{iH} \tag{23}$$

za podmínek

$$\begin{aligned} &\sum_{j=1}^{n} u_{jH} y_{jH} = 1 \\ &-\sum_{i=1}^{m} v_{iH\,ik} + \sum_{j=1}^{n} u_{jH} y_{jk} \geq 0,\ k = 1,2,..., p, \\ &u_{jH} \geq 0,\ j = 1,2,..., n, \\ &v_{iH} \geq 0, i = 1,2,..., m. \end{aligned} \tag{24}$$

Duální model má tvar

$z_1 \to \max$

za podmínek

$$\begin{aligned}
&\sum_{k=1}^{p} \lambda_{kH} x_{ik} \le x_{iH},\ i = 1,2,...,m, \\
&z_H y_{jH} - \sum_{k=1}^{p} \lambda_{kH} y_{jk} \le 0,\ j = 1,2,...,n \\
&y_{kH} \ge 0,\ k = 1,2,...,p
\end{aligned} \tag{25}$$

Příklad 2 – hodnocení metodou DEA – výstupově orientovaný model

(převzato se svolením autora z [1])

Zadání

Zadání je totožné s příkladem uvedeným v předchozí kapitole.

Řešení

Primární model pro první pobočku – pobočku A:

$$\begin{aligned}
\Phi_1 &= 4v_{11} + 3v_{21} \to \min \\
u_{11} + 5u_{21} &= 1 \\
4v_{11} + 3v_{21} - u_{11} - 5u_{21} &\ge 0 \\
7v_{11} + 3v_{21} - 2u_{11} - 7u_{21} &\ge 0 \\
8v_{11} + v_{21} - 3u_{11} - 4u_{21} &\ge 0 \\
4v_{11} + 2v_{21} - 4u_{11} - 3u_{21} &\ge 0 \\
2v_{11} + 4v_{21} - 4u_{11} - 6u_{21} &\ge 0 \\
5v_{11} + 2v_{21} - 5u_{11} - 5u_{21} &\ge 0 \\
6v_{11} + 4v_{21} - 6u_{11} - 2u_{21} &\ge 0 \\
u_{j1} &\ge 0,\ j = 1,\ 2, \\
v_{i1} &\ge 0,\ i = 1,\ 2.
\end{aligned} \tag{26}$$

Proměnná	Hodnota
Φ_1	1,15
v_{11}	0,1
v_{21}	0,25
u_{11}	0
u_{21}	0,2

Tabulka 6: Váhy pro jednotku A

Z výsledku je patrné, že jednotka A (první) není efektivní, neboť hodnota e_1 je větší než jedna. Proto přistoupíme k sestavení duálně sdruženého modelu, jehož vyřešením získáme vhodnou kombinaci vzorových jednotek pro dosažení efektivity neefektivní jednotky A. Duální model pro první jednotku – jednotku A:

$$\begin{aligned} z_1 &\rightarrow \max \\ 4\lambda_{11} + 7\lambda_{21} + 8\lambda_{31} + 4\lambda_{41} + 2\lambda_{51} + 5\lambda_{61} + 6\lambda_{71} &\leq 4 \\ \lambda_{11} + 3\lambda_{21} + 3\lambda_{31} + 2\lambda_{41} + 4\lambda_{51} + 2\lambda_{61} + 4\lambda_{71} &\leq 3 \\ \lambda_{11} - \lambda_{11} - 2\lambda_{21} - 3\lambda_{31} - 4\lambda_{41} - 4\lambda_{51} - 5\lambda_{61} - 6\lambda_{71} &\leq 0 \\ 5\lambda_{11} - 5\lambda_{11} - 7\lambda_{21} - 4\lambda_{31} - 3\lambda_{41} - 6\lambda_{51} - 5\lambda_{61} - 2\lambda_{71} &\leq 0 \\ \lambda_{k1} \geq 0,\ k &= 1, 2, ..., 7 \end{aligned} \tag{27}$$

Proměnná	Hodnota
z_1	1,15
z_1	1,15
λ_{11}	0
λ_{21}	0
λ_{31}	0
λ_{41}	0
λ_{51}	0,4375
λ_{61}	0,625
λ_{71}	0

Tabulka 7: Výsledky duálního modelu

Z výsledku vyplývá, že vzorovými (peer) jednotkami pro jednotku A jsou jednotky pátá (E) a šestá (F), neboť hodnoty proměnných λ_{51} a λ_{61} jsou nenulové. Proměnná λ_{51} se vztahuje k páté jednotce (E) – první index je 5, proměnná λ_{61} se vztahuje k šesté jednotce (F) – první index je 6. Známe tedy koeficienty kombinace. Nyní musíme výstupy efektivních jednotek E a F pronásobit koeficienty a pro každý vstup zvlášť tyto výsledky sečíst, abychom získali optimální velikost výstupů pro jednotku A.

Konkrétně pro první výstup:

$$y'_{11} = \lambda_{51} y_{15} + \lambda_{61} y_{16} = 0{,}4375 \cdot 4 + 0{,}625 \cdot 5 = 4{,}875$$

Pro druhý výstup:

$$y'_{21} = \lambda_{52} y_{25} + \lambda_{62} y_{26} = 0{,}4375 \cdot 6 + 0{,}625 \cdot 5 = 5{,}75$$

Jednotka A by měla zvýšit svůj první výstup – počet obsloužených zákazníků - z původního 1 na zhruba 5. Druhý výstup – tržby - by měla zvýšit z původních 50.000 Kč na zhruba 57.500 Kč. Potom bude jednotka A efektivní.

Stejně jako pro jednotku A je nutné sestavit modely pro všechny ostatní jednotky a vyřešit je stejným způsobem. V tabulce 8 jsou uvedeny výsledky pro všechny jednotky.

	Primární model					Duální model						
	e_k	v_{1i}	v_{2i}	u_{1j}	u_{2j}	λ_{1k}	λ_{2k}	λ_{3k}	λ_{4k}	λ_{5k}	λ_{6k}	λ_{7k}
A	1,15	0,1	0,25	0	0,2	0	0	0	0	0,438	0,625	0
B	1,036	0,072	0,179	0	0,143	0	0	0	0	0,063	1,375	0
C	1	0,068	0,455	0	0,25	0	0	1	0	0	0	0
D	1,063	0,188	0,156	0,25	0	0	0	0	0	0,125	0,75	0
E	1	0,083	0,208	0	0,167	0	0	0	0	1	0	0
F	1	0,15	0,125	0,2	0	0	0	0	0	0	1	0
G	1,167	0,125	0,104	0,167	0	0	0	0	0	0,5	1	0

Tabulka 8: Výsledky modelového příkladu

3. Možnosti současných vývojových prostředí

(Použité zdroje [5], [6], [7])

Jako vývojové prostředí při tvorbě programu jsem použil Microsoft Visual Studio 2008. Toto prostředí umožňuje tvorbu programů v mnoha programovacích jazycích. Tuto aplikaci jsem programoval v jazyce C# a XAML. Visual Studio a jazyk C# využívá nových možností frameworku Microsoft .NET 3.5.

3.1. Novinky v .NET 3.5

```
/// <summary>
/// rozsireni tridy,
/// vraci seznam s opacnymi hodnotami
/// </summary>
/// <param name="toOpposite">vstupni seznam</param>
/// <returns>seznam s opacnych hodnot</returns>
public static List<decimal> OppositeValues(this List<decimal> toOpposite)
{
    //'tradicni' zapis
    List<decimal> neg = new List<decimal>(toOpposite.Count);
    foreach (decimal d in toOpposite)
        neg.Add(-d);
    return neg;

    //zapis pomoci Language Integrated Querry:
    return (from d in toOpposite
            select -1 * d).ToList();

    //zapis pomoci Lambda Expression
    return toOpposite.Select(d => -1 * d).ToList();
}
```


Ukázka 1: Nové možnosti .NET 3.5

3.1.1. Linq (Language INtegrated Querry)

LINQ je dotazovací jazyk integrovaný přímo v jazycích .NET. Poskytuje metody pro práci s daty (výběr, filtrování, třídění, grupování, . . .)

- LINQ - dotazování v objektech (kolekcích, polích apod.)
- LINQ to XML - dotazování do XML
- LINQ to SQL - dotazování do databáze

Společně s jazykem LINQ (a částečně právě díky němu) přicházejí i další užitečná rozšíření

- Extension methods - umožňují rozšířit již existující typ o nové instanční metody bez toho, abychom jakýmkoli způsobem do daného typu zasáhli, nebo abychom od tohoto typu dědili (což v případě sealed classes ani nelze).
- Lambda expressions - pomocí lambda výrazů je možné tvořit anonymní metody, které obsahují jeden výraz nebo několik příkazů, a použít je kdekoli, kde je očekávána instance delegáta.
- Anonymous types - umožňují definovat nové bezejmenné typy za běhu a rovnou vytvořit jejich instanci
- Partial methods - klíčové slovo partial je možné aplikovat nejen na typy, ale také na jednotlivé metody, čímž se dosáhne možnosti oddělit deklaraci metody od její implementace v jednotlivých souborech
- Automatic properties – jednoduché property, které pouze nastavují hodnotu jedné privátní proměnné, lze nyní napsat mnohem jednodušším zápisem díky automatickým vlastnostem. Privátní proměnnou již není třeba deklarovat a těla částí get a set zůstanou prázdná. Překladač si vše potřebné doplní sám.

3.1.2. Windows Presentation Foundation (WPF)

Grafický subsystém primárně určený pro Windows Vista po instalaci .NET frameworku 3.0 nebo vyšším funguje i na XP.

Kód kontrolky nebo okna je rozdělen do dvou souborů. V jednom je pomocí jazyka XAML definován vzhled a vázání s daty. V druhém souboru je pak tzv. kód na pozadí, psaný v libovolném .NET jazyce (v mém případě C#). Svázání těchto souborů dohromady je umožněno technologií částečných (partial) tříd.

3.1.3. Jazyk XAML

XAML (eXtensible Application Markup Language) je značkovací, deklarativní jazyk založený na XML. Je určen k definování vzhledu aplikace, definování interakce s uživatelem a vázání grafického rozhraní s daty. WPF je založeno na vektorové grafice, proto je možné provádět plynulé animace, přechody barev, rotace, změny průhlednosti, vykreslovat stíny a odlesky atd.

4. Požadavky na vstupy a výstupy programu

Předmětem zadání bylo vytvoření aplikace, která by umožnila provádět optimalizační výpočty za použití modelů metody DEA.

4.1. Vstupy

Vstupní data jsou definována jako tabulka produkčních jednotek a hodnot jejich vstupů a výstupů. Příklad je uveden v následující tabulce, která obsahuje několik produkčních jednotek (Pobočky) a hodnot jejich vstupů (zaměstnanci a režijní náklady) a výstupů (zákazníci a tržby).

Formát vstupních dat měla představovat *tabulka 9*

	Zaměstnanci	Režijní náklady	Zákazníci	Tržby
Pobočka A	4	3	1	5
Pobočka B	7	3	2	7
Pobočka C	8	1	3	4
Pobočka D	4	2	4	3
Pobočka E	2	4	4	6
Pobočka F	5	2	5	5
Pobočka G	6	4	6	2

Tabulka 9: Ukázková vstupní data

4.2. Výstupy

Požadovaný výstup lze shrnout do následujících bodů:

1. požadovaný výstup představují hodnoty efektivity pro jednotlivé jednotky
2. dalším výstupem budou doporučení pro neefektivní jednotky
3. vzorové jednotky pro neefektivní jednotky
4. grafické znázornění

5. Implementace

Při návrhu programu jsem dodržoval principy a doporučení platná pro objektové programování. Mým cílem bylo vytvořit výpočetní jádro oddělené od grafického uživatelského rozhraní. Aplikaci jsem rozdělil do dvou nezávislých projektů: výpočetního jádra a uživatelského grafického rozhraní.

Jádro aplikace tvoří DLL (Dinamic Linked Library) knihovna, která má na starost provádění výpočtů. Pro komunikaci s grafickým uživatelským prostředím (GUI) slouží přesně definovaný interface.

Proces zadání, výpočtu a zobrazení výsledků probíhá následovně:

- GUI odešle vstupní data v definovaném formátu výpočetnímu jádru
- výpočetní jádro provede výpočet požadovaného modelu
- výpočetní jádro z výsledků výpočtu extrahuje výstupní data, opět v jasně definovaném formátu
- výstupní data jsou odeslána GUI
- GUI zobrazí výsledky, tedy optimalitu jednotek a případná doporučení pro jejich optimalizaci. Výsledky jsou zobrazeny v ‘user friendly’ podobě.

Výpočetní jádro je distribuované ve formě DLL souboru. Dokumentace k projektu je umístěna na webových stránkách [8]. Díky tomuto řešení je možno postupně, v případě potřeby, dopisovat pro aplikaci další rozhraní, (programy s jiným způsobem zadávání vstupů a prezentace výstupů, webová rozhraní, webservices)

Následuje popis výpočetního jádra a GUI.

5.1. Výpočetní jádro

5.1.1. Princip funkce

Princip funkce je následovný: vstupní data se předají do konstruktoru třídy DeaTaskSolvable, která obsahuje kolekci instancí třídy ProductionUnit, jenž představuje produkční jednotku. ProductionUnit obsahuje jméno jednotky a seznam hodnot jejích vstupů a výstupů. DeaTaskSolvable pak při řešení úlohy předává své produkční jednotky vybranému modelu, který implementuje interface IDeaModel. Rozhraní IdeaModel je zavedeno pro budoucí rozšiřitelnost aplikace, případné další modely metody DEA budou implementací tohoto rozhraní.

Třída implementující rozhraní IdeaModel obsahuje slovník produkčních jednotek a k nim příslušejících simplexových tabulek. Simplexové tabulky pro jednotlivé modely jsou vytvářeny, a po zpracování vyhodnocovány, pro různé modely různým způsobem.

Simplexové tabulky jsou instancemi třídy STSolvable, která obsahuje metody pro řešení simplexové tabulky.

Po přijetí zprávy InvokeModelSendResult instance třídy DeaTaskSolvable spustí výpočet simplexových tabulek pro všechny své produkční jednotky. Po ukončení výpočtů provede extrakci výsledků a získaná data odešle v parametru události OnModelHasResult

5.1.2. Podrobný popis

Vytvoření a řešení simplexové tabulky

Třídy potřebné pro řešení simplexových tabulek se nacházejí ve jmenném prostoru DeaSolverProject.SimplexTables.

Účelová funkce a seznam nerovnic příslušejících k určité produkční jednotce je předán instanci třídy STSolvable, tím je spuštěn řetězec událostí vedoucí ve výsledku k vytvoření simplexové tabulky.

Třída STSolvable je, přes několik abstraktních tříd, oděděna od abstraktní třídy STB, jak ukazuje *obrázek 2.*

V konstruktoru třídy STSolvable je volán konstruktor její nadtřídy STconstrucableFinall a takto postupuje tok programu dále až k výchozí třídě STB, která je sama oděděna od třídy System.Data.DataTable. V konstruktoru jsou přitom předávány reference na účelovou funkci a příslušnou soustavu nerovnic. V okamžiku dosažení třídy STB tato vytváří instanci třídy System.Data.DataTable. Potom se tok programu obrací, při průchodu zpět každou ze tříd je simplexová tabulka částečně upravena. K rozdělení kódu pro tvorbu simplexové tabulky do několika tříd mne vedla snaha o přehlednost, udržitelnost a silnou kohezi programu. Každá ze tříd této hierarchie přidává jasně definovanou část funkčnosti, za kterou nese také zodpovědnost.

STB: rozšiřuje funkcionalitu třídy System.Data.DataTable především o property potřebné pro práci s daty simplexové tabulky.

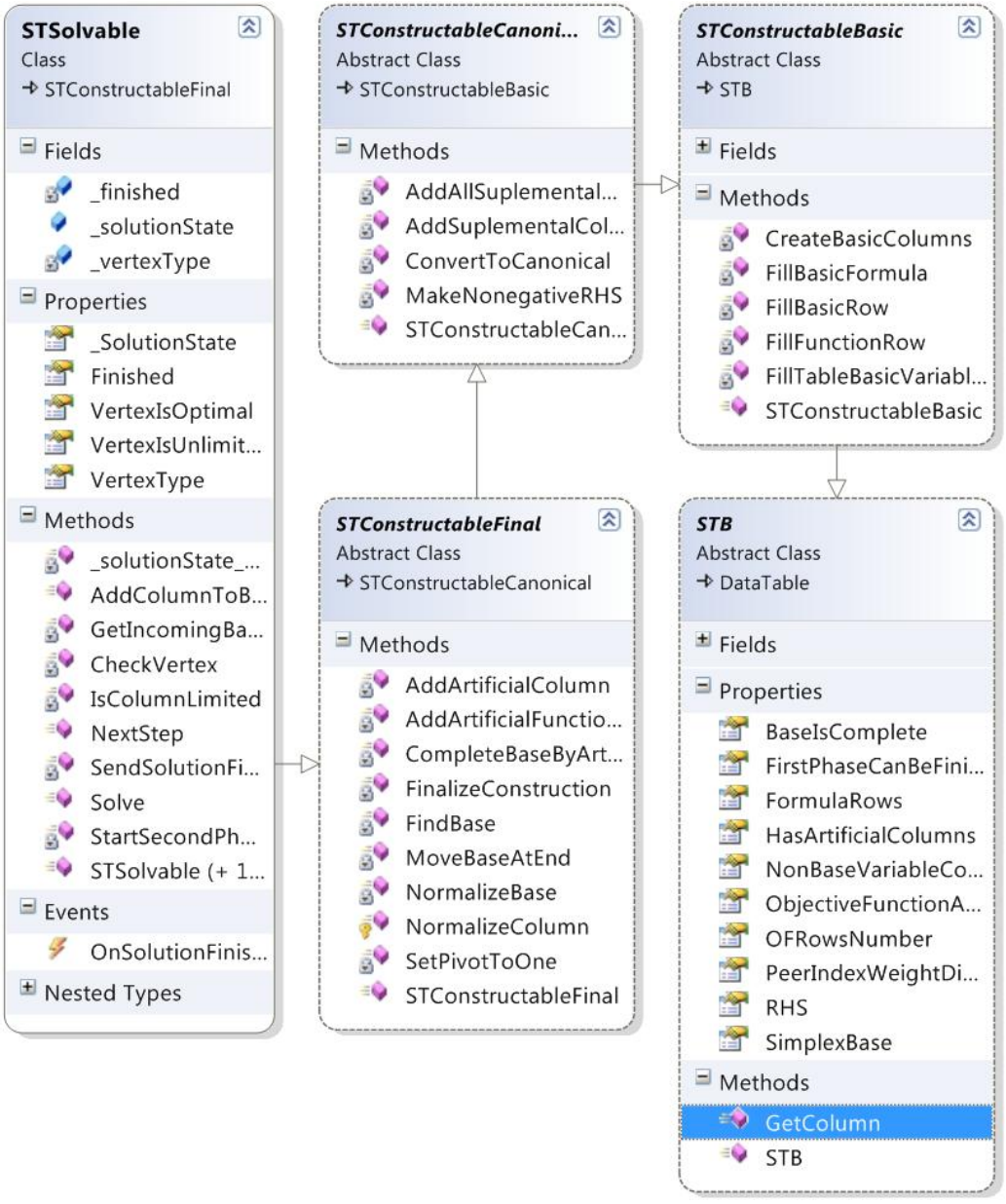
STConstuctableBasic: obsahuje metody potřebné k vytvoření simplexové tabulky v základním tvaru.

STConstructableCanonical: převádí tabulku ze základního tvaru na kanonický tvar.

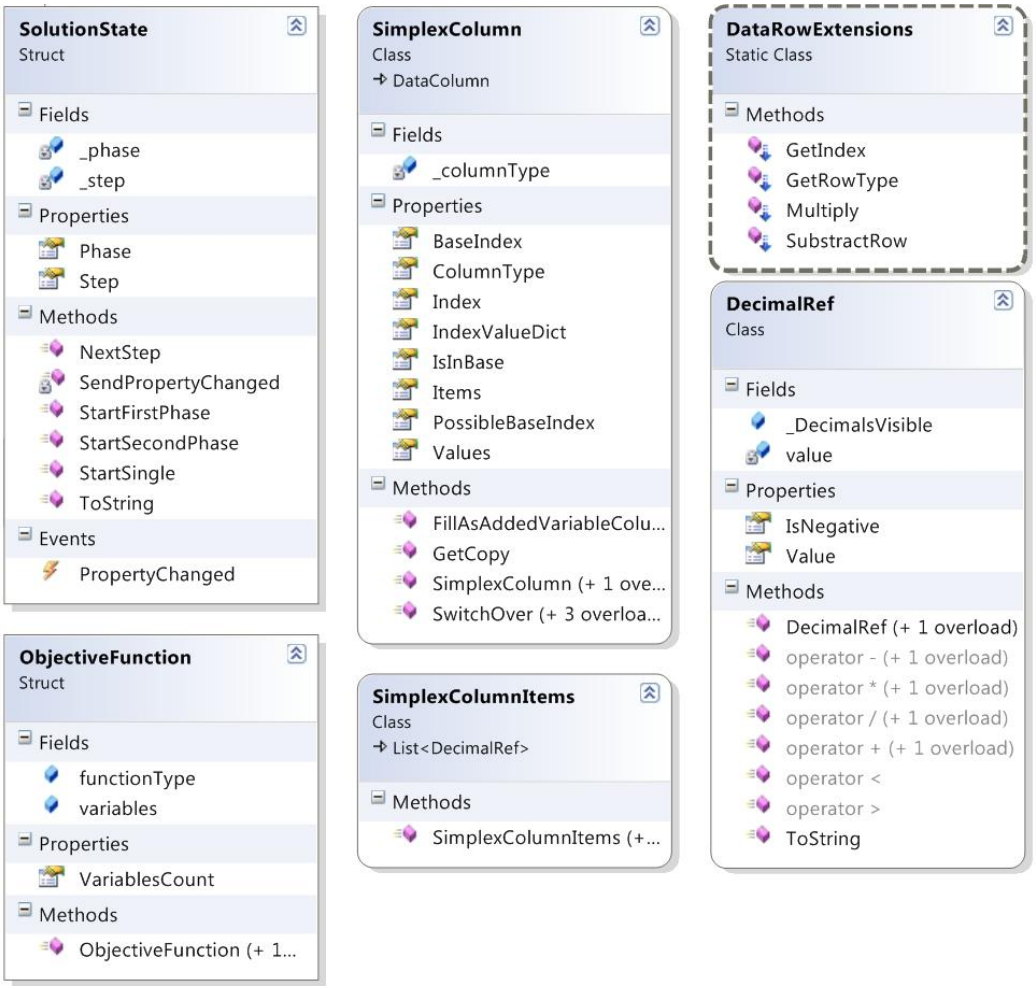
STConstructableFinall: dokončuje konstrukci simplexové tabulky přidáním řádku účelové funkce umělých proměnných.

STSolvable: po obdržení zprávy Solve provede řešení simplexové tabulky.

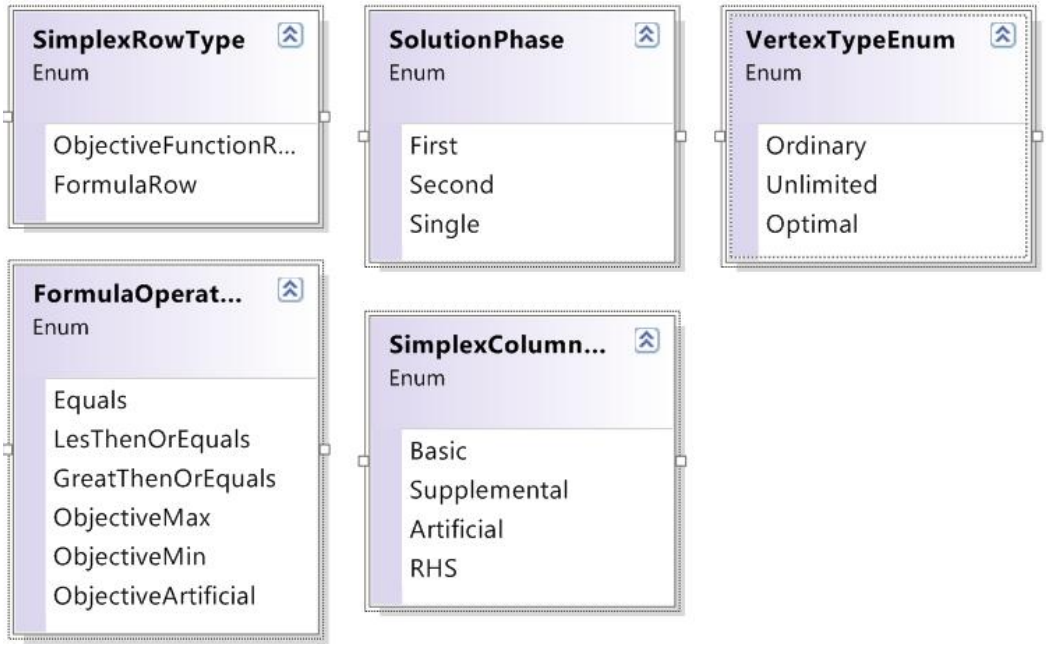
Ve jmenném prostoru DeaSolverProject.SimplexTables je dále vnořen jmenný prostor DeaSolverProject.SimplexTables.Parts, ve kterém se nalézají další třídy, struktury a výčtové typy potřebné pro konstrukci a řešení simplexové úlohy. Viz. *obrázek 3* a *obrázek 4*



Obrázek 2: NameSpace DeaSolverProject.SimplexTables



Obrázek 3: NameSpace DeaSolverProject.SimplexTables.Parts , třídy a struktury

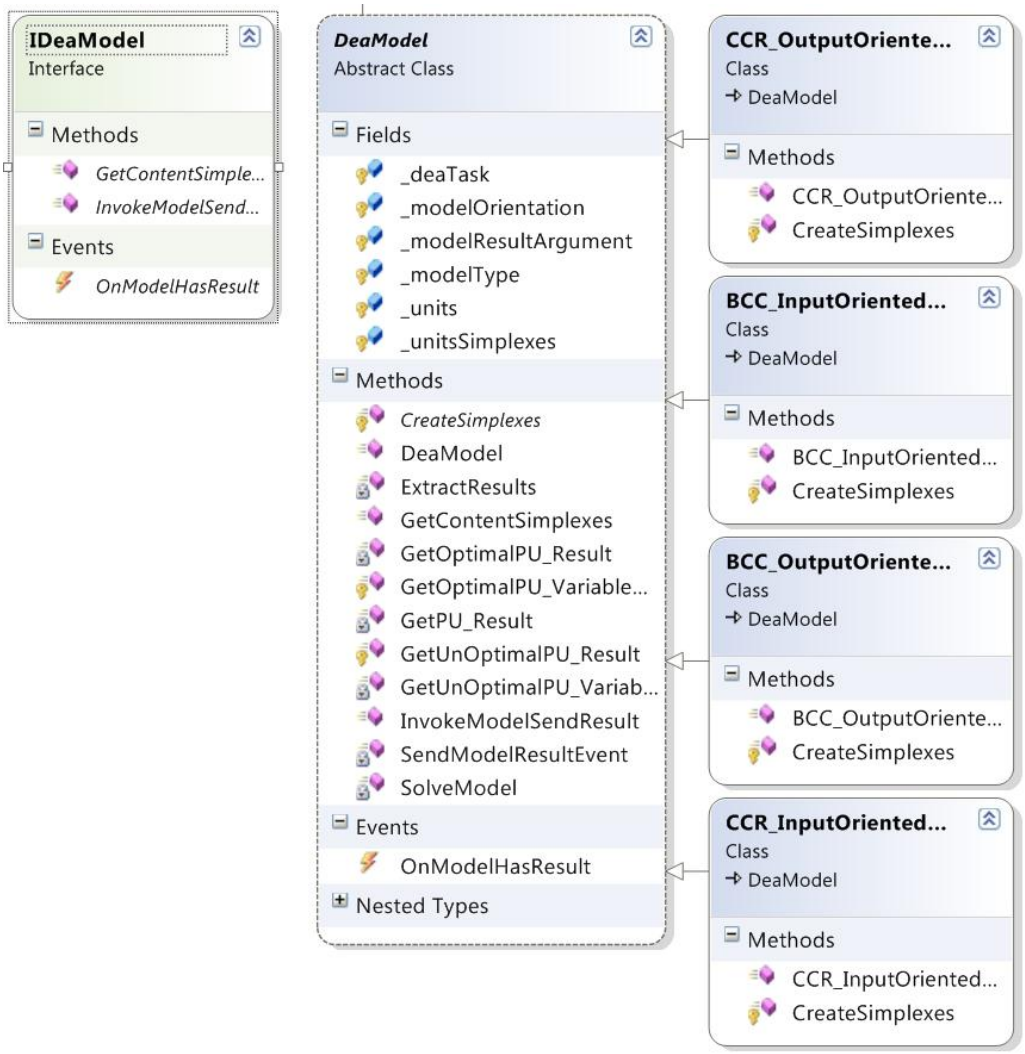


Obrázek 4: NameSpace DeaSolverProject.SimplexTables.Parts , enumerace

DEA

Dalším důležitým jmenným prostorem je DeaSolverProject.DEA. Zde se nachází třídy potřebné pro konstrukci úloh DEA.

Ve jmenném prostoru DeaSolverProject.DEA.Models se nachází jednotlivé modely řešení úloh DEA, viz *obrázek 5*.



Obrázek 5: NameSpace DeaSolverProject.DEA.Models

5.2. GUI

5.2.1. Princip funkce

Hlavní okno aplikace je tvořeno tabulkou (DataGridView) očekávající vstupní data. Ta je možno zadat ručně nebo načíst z XML nebo Excelovského CSV souboru. Zadaná data je možno do XML nebo CSV souboru také uložit. Popis okna viz *obrázek 8.*

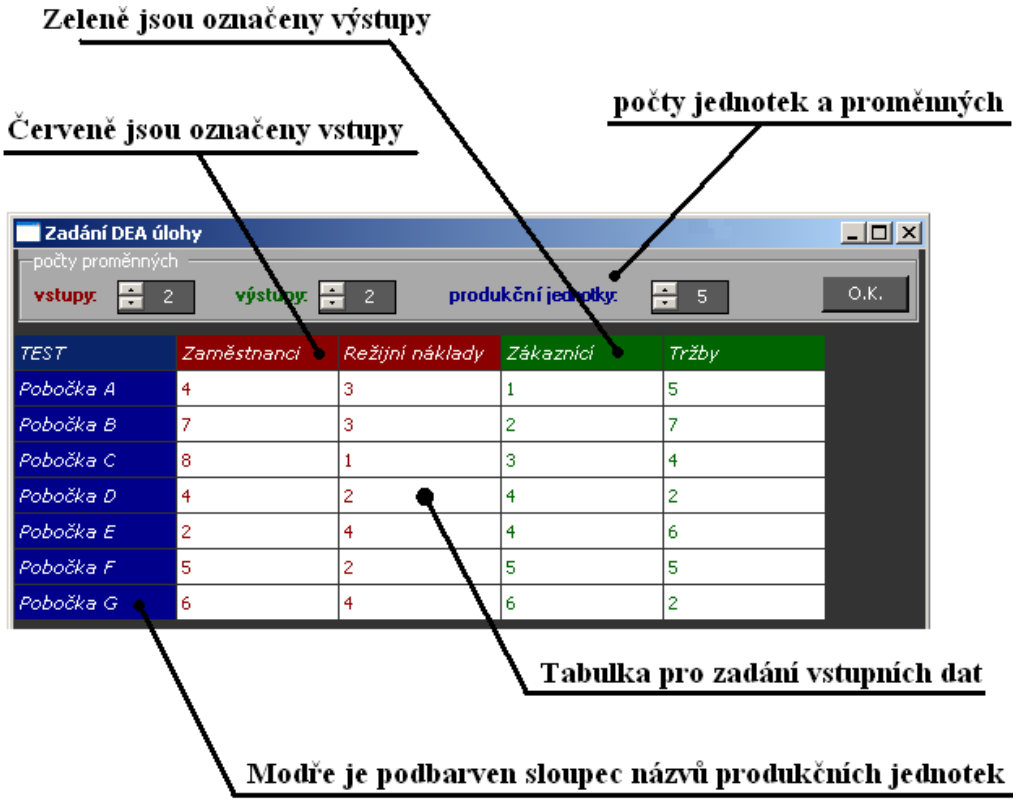
Po ukončení zadání, resp. načtení vstupních dat, jsou tato odeslána výpočetnímu jádru. Kontrolka pro vstup dat tedy provádí komunikaci se třídou DeaTaskSolvable.

Další kontrolka na hlavním okně obsahuje seznam dostupných modelů (viz *obrázek 7*). Výběrem modelu je odeslána výpočetnímu jádru zpráva se žádostí o výsledky požadovaného modelu pro odeslaná vstupní data.

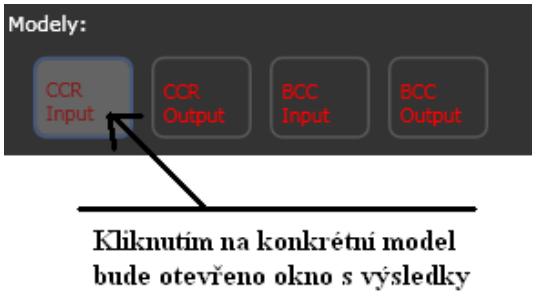
Jakmile GUI obdrží od jádra událost ukončení výpočtu OnModelHasResult, zobrazí výsledky výpočtu obsažené v parametru této události.

Výsledky jsou zobrazeny v novém okně ve formě sloupcových grafů s hodnotami optimalit jednotlivých jednotek (viz *obrázek 8*).

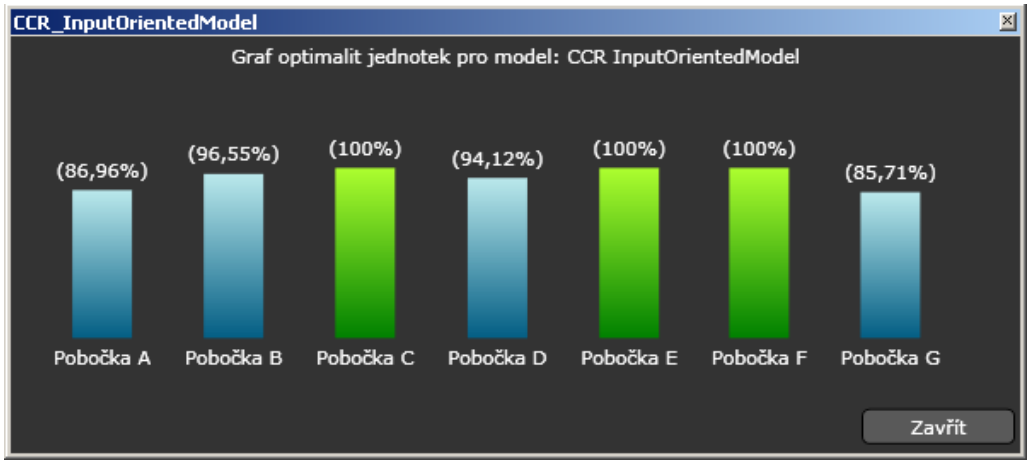
Detaily konkrétní jednotky se zobrazí jako popup okno po najetí myší na sloupec optimality (viz *obrázek 9*). Formulář s detailem jednotky zobrazuje jednotlivé proměnné, tedy vstupy a výstupy, produkční jednotky opět ve formě sloupcových grafů. Velikost sloupce je dána hodnotou optimality proměnné, u neoptimálních proměnných se zobrazí doporučení pro optimalizaci spolu se vzorcem výpočtu.



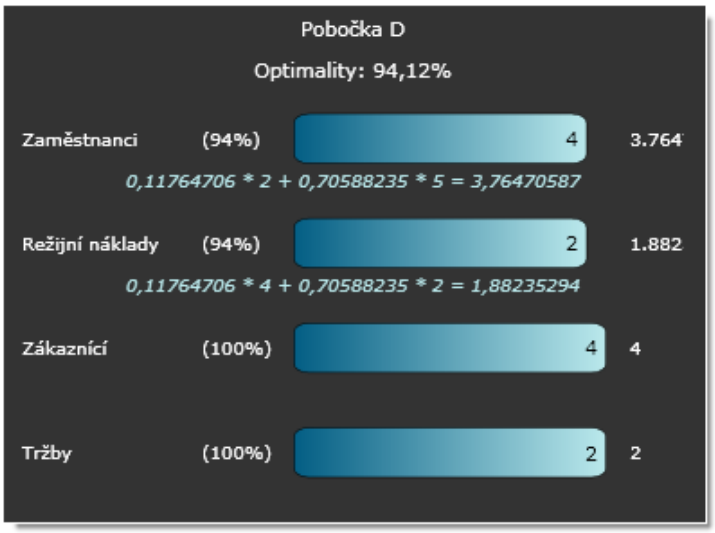
Obrázek 6: Tabulka pro zadání vstupních dat



Obrázek 7: Výběr modelů



Obrázek 8: Znázornění výsledků - graf optimalit



Obrázek 9: Reprezentace detailu produkční jednotky

Použitá technologie (WPF)

Windows Presentation Foundation (dále jen WPF) je grafický subsystém .NET frameworku 3.0 a novějších. Více informací o WPF viz kapitola o vývojových prostředích.

Zde uvádím několik konkrétních případů použití technologie WPF v tomto projektu.

- Pokročilé vázání dat a datová šablona

```
<!--Datatemplate pro listbox-->
<DataTemplate x:Key="chartColumnTemplate">
  <Border Margin="5">
    <StackPanel>
      <Label
        HorizontalAlignment="Center"
        Content="{Binding Path=UnitOptimality,
        Converter={StaticResource converter}}"
        />
      <Rectangle
        Height="{Binding Path=UnitOptimality}"
        Style="{StaticResource chartColumnRectangle}"
        />
      <Label
        Content="{Binding Path=ProductionUnitName}"
        />
    </StackPanel>
  </Border>
</DataTemplate>
```

Ukázka 2: Příklad datové šablony

DataTemplate struktura říká, jak má kontrolka (v tomto případě ListBox) zobrazovat data v ní obsažená.

- Styly

Styl je struktura definující především vzhled kontrolky a případnou interakci s okolím (pomocí tzv. Triggerů). Doporučuje se tyto styly definovat odděleně v slovníku zdrojů ResourceDictionary, které je možno následně přepínat a tím měnit vzhled aplikace. Styl upravující vzhled tlačítka (viz *Ukázka* 3: Příklad stylu).

```
<Style TargetType="{x:Type Button}" x:Key="{x:Type Button}">
 <Setter Property="Foreground" Value="White" />
 <Setter Property="Background" Value="#595959" />
 <Setter Property="Margin" Value="3" />
 <Setter Property="MinHeight" Value="21"/>
 <Setter Property="MaxHeight" Value="21"/>
 <Setter Property="MinWidth" Value="90"/>
 <Setter Property="FontSize" Value="11" />
 <Setter Property="FontFamily" Value="Verdana" />
 <Setter Property="Template">
  <Setter.Value>
   <ControlTemplate
     TargetType="Button">
     <Border x:Name="Border"
       CornerRadius="4"
       BorderThickness="1"
       Background="{TemplateBinding Background}"
       BorderBrush="Black">
       <ContentPresenter
        x:Name="Content"
        Margin="20,2,20,2"
        HorizontalAlignment="Center"
        VerticalAlignment="Center"
        RecognizesAccessKey="True"/>
     </Border>
     <ControlTemplate.Triggers>
      <Trigger Property="IsMouseOver"
        Value="true">
        <Setter Property="Background" Value="#AEAEAE" />
        <Setter Property="Foreground" Value="Black" />
      </Trigger>
      <Trigger Property="IsEnabled" Value="False">
        <Setter Property="Background" Value="#666666" />
        <Setter Property="Foreground" Value="#808080" />
      </Trigger>
    </ControlTemplate.Triggers>
   </ControlTemplate>
  </Setter.Value>
 </Setter>
</Style>
```

Ukázka 3: Příklad stylu

6. Zhodnocení a závěr

6.1. Splnění cílů

6.1.1. Vstupy

Aplikace umožňuje vkládání dat dvěma způsoby:

1. přímým zadáním do vstupní tabulky
2. načtením ze souboru XML nabo Excelovského formátu CSV

Vložená data je možno rovněž exportovat do souboru, opět ve formátu XML nebo CSV. Požadavky na vstupní část jsou tedy splněny.

6.1.2. Výpočet výsledků

Výpočetní jádro implementuje dva modely DEA, jsou to CCR vstupově orientovaný a CCR výstupově orientovaný model. Pro tyto modely jsem provedl ověření výsledků výpočtu. K porovnání jsem měl k dispozici několik množin vstupních dat a jim odpovídající výsledky řešené v profesionálním programu Banxia Frontier Analyst. Testovací data jsou z praxe zemědělské výroby, vstupní hodnoty jsou v *tabulce č. 10.* Výsledky programu Banxia Frontier Analyst jsou v *tabulce č. 11*. Po zadání stejných vstupních dat do mého programu jsem obdržel výsledky shodné s profesionálním řešením, viz *obrázek č. 10*

Implicitní požadavek na správnost výpočtu je tedy rovněž splněn.

6.1.3. Výstupy

Všechny požadavky na výstupy, definované v části Zadání, byly splněny, jak vyplývá z *obrázků č 8*, *9 a 10*.

	vstupy					výstupy		
jedn	krmivo	spotř	nákl./s	amort.	am. pr	narozeno	odchov.	hmot
1	3,44	64,42	206	42	71	20,77	19,49	7
2	3,65	64,39	98	46	162	21,5	20,69	7,08
3	3,25	60,68	173	43	160	21,69	19,55	6,91
4	3,55	68,34	52	7	28	21,15	18,96	6,83
5	3,38	68,73	72	8	93	19,61	17,95	6
6	3,15	58,96	223	38	98	21,72	19,5	6,2
7	2,76	53,93	90	6	116	20,75	18,68	5,2
8	3,28	56,77	125	56	83	24,14	21,09	6,6
9	2,51	52,08	117	50	83	19,2	17,59	8,13
10	4,08	78,42	78	38	49	21,19	18,99	5,25
11	3,42	68,55	170	35	76	21,03	18,21	7,93
12	3,26	61,68	78	29	44	20,4	19,29	6
13	3,13	57,47	101	34	55	23,61	19,88	6,82
14	3,22	56,78	122	11	134	22,02	20,7	7
15	3,79	66,60	97	133	74	22,61	20,77	6,2
16	3,15	59,63	77	51	92	21,01	19,28	6,73
17	2,66	50,49	105	52	57	21,11	19,23	6,74
18	3,51	56,61	158	112	110	25,45	22,63	7,07
19	3,23	58,83	176	19	87	22,62	20,04	6,48
20	3,25	61,69	112	61	72	21,69	19,23	7,28

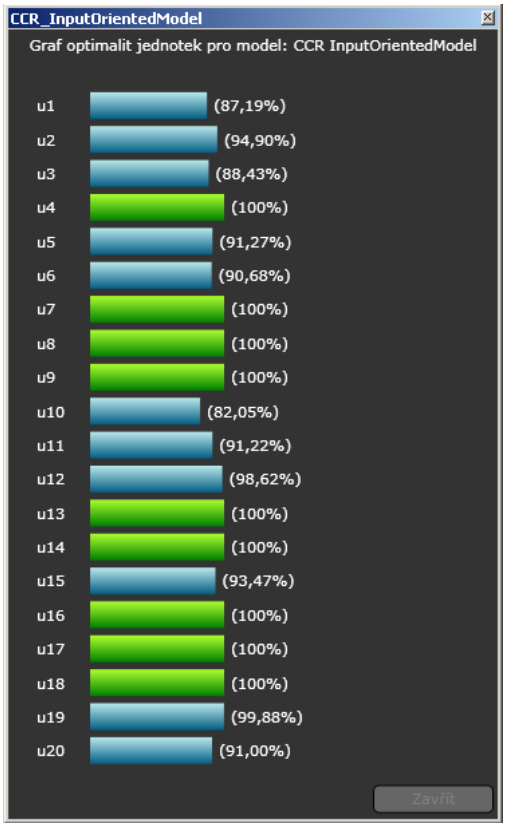
Tabulka 10: Vstupní data pro porovnání správnosti výpočtu

Vysledky CCR		
vstupy	Unit	Score
	1	87,19
	2	94,9
	3	88,43
	4	100
	5	91,27
	6	90,68
	7	100
	8	100
	9	100
	10	82,05
	11	91,22
	12	98,62
	13	100
	14	100
	15	93,47
	16	100
	17	100
	18	100
	19	99,88
	20	91

Tabulka 11:

Výsledky programu

Banxia Frontier Analyst



Obrázek 10: Výsledky mého programu

6.2. Přínos projektu

Věřím, že aplikace, jejíž vytvoření bylo cílem této bakalářské práce, bude sloužit účelu, k němuž byla vytvořena, tedy k podpoře výuky rozhodovacích modelů na Ekonomické Fakultě JU.

Vzhledem k nezávislosti výpočetního jádra aplikace a možnosti doplňování dalších modelů (jako implementací rozhraní `IDeaModel`) je možno systém dále rozšiřovat (rozšíření systému je cílem bakalářské práce mého kolegy Jiřího Adámka pro akademický rok 2008/2009).

Kromě splnění zadání práce na tomto projektu přispěla k rozvoji mých programátorských znalostí. Také jsem pochopil, že základní principy a poučky objektového programování je nutno při vytváření projektu tohoto rozsahu (cca 5000 řádků kódu) důsledně dodržovat.

7. Reference

[1] **Friebelová, Jana, Klicnarová, Jana.** Rozhodovací modely pro ekonomy. EF JU v Č. Budějovicích, 2007

[2] **Janáček, Jaroslav.** Matematické programování. EDIS, 2003

[3] **Hliněný, Petr.** Optimalizační úlohy, výukový text. FI MU, 2007 [Online] [Citace: 15.1.2008] http://www.fi.muni.cz/~hlineny/Teaching/OU/OU-text07.pdf

[4] http://www.nationmaster.com/encyclopedia/Simplex-algorithm [Online] [Citace: 24.4.2008]

[5] Microsoft Developer Network [Online] [Citace: 14.4.2008] http://msdn2.microsoft.com/

[6] **Klein, Scott.** Professional LINQ. Wrox Press USA, 2008.

[7] webové stránky Scotta Guthrie, vývojáře jazyka LINQ [Online] [Citace: 14.4.2008] http://weblogs.asp.net/scottgu/

[8] dokumentace k aplikaci [Online] [Citace:24.4.2008] http://lama.zf.jcu.cz/dea/documentation/Index.aspx