VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

MAYERSNŮV VEKTOR V EKONOMII
MAYERS VALUE IN ECONOMICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Alexandr Karmazin

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Mgr. Jaroslav Hrdina, Ph.D.

BRNO 2016

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Karmazin Alexandr, Bc.

Matematické metody v ekonomice (6207R005)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Mayersnův vektor v ekonomii

v anglickém jazyce:

Mayers Value in Economics

Pokyny pro vypracování:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury

Seznam odborné literatury:

GILLES, Robert P. The Cooperative Game Theory of Networks and Hierarchies. Berlin, Springer Berlin Heidelberg, 2010. ISBN 978-3-642-05281-1.
OWEN, Guillermo. Game Theory. Emerald Group Publishing Limited, 2013. ISBN 978-1-781-90507-4.
PETERS, Hans. Game Theory-A Multi-Leveled Approach.
Springer-Verlag Berlin Heidelberg, 2008. ISBN 978-3-540-69291-1.
SCHOFIELD, Norman. Mathematical Methods in Economics and Social Choice. Springer Texts in Business and Economics, 2014. ISBN 978-3-540-00086-0.

Vedoucí bakalářské práce: doc. Mgr. Jaroslav Hrdina, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc.	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 29.2.2016

Abstrakt

Bakalářská práce je zaměřená na koaliční hry v teorií her. Na začátku práce jsou definované důležité pojmy, které se tyto hry popisuji. Další částí práce je taktéž aplikace těchto poznatku na reálné situaci, konkrétně se jedná o určení vyjednávací síly politických stran v ČR pomoci Myersnovy hodnoty. Součástí práce je taktéž vlastní aplikace vyvinuta v matematickém software Matlab pro výpočet této hodnoty.

Abstract

Bachelor thesis is focused on the coalition games in game theory. At the beginning, important terms that relate to these games are defined. Next part of the work is also the application of this knowledge to the real situation, namely the determination of the bargaining power of political parties in the Czech Republic using Myerson value. The work also includes custom application developed in mathematical software Matlab to calculate this value.

Klíčová slova

Teorie her, Shapleyho hodnota, Myersnova hodnota, Matlab, Poslanecká sněmovna

Key words

Game theory, Shapley value, Myerson value, Matlab, Chamber of Deputies

Bibliografická citace

KARMAZIN, A. *Mayersnův vektor v ekonomii.* Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 64 s. Vedoucí bakalářské práce doc. Mgr. Jaroslav Hrdina, Ph.D..

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu zákona č. 121/2000Sb., o právu autorském a o právech souvisejících s právem autorským.

V Brně …… Podpis……

Poděkování

Zde bych rád poděkoval doc. Mgr. Jaroslavu Hrdinovi, Ph.D. za poskytnutí odborného vedení, cenných rad a připomínek pro vypracování této práce.

OBSAH

ÚVOD

Teorie her je vědní obor, který řadíme jak do matematické ekonomie, tak do teorie rozhodování a operačního výzkumu. Zabývá se rozborem širokého spektra rozhodovacích situací s více účastníky. Může se jednat například o hlasování akcionářů na valné hromadě, nebo soupeření obchodníků na burze. [1]

V druhé polovině dvacátého století prošel tento obor značným rozvojem a vedle dosavadních časopisů se začaly objevovat i první knihy věnované teorii her, které shrnuly dosavadní výsledky, ale taky přinesly nová hlediska na tuto problematiku. Aktuálně je teorie her uplatňovaná v několika oblastech lidské činnosti, především v ekonomii. [1]

Hry jsou považované za modely reálných či imaginárních situací s prvky inteligence a soupeření. Teorie her se následně používá k analýze strategických vazeb mezi ekonomickými subjekty a hledá rovnovážné řešení dané situace, určuje vlivy, které tuto situaci z rovnovážné polohy vychylují, nebo co naopak brání danému systému být v rovnováze. [1]

Využiti teorie her nalezneme například při aukcích nebo teoriích vyjednávání. Ve své bakalářské práci se budu zaměřovat především na druhou zmiňovanou část. Znalosti z teoretické části budou aplikované na zjištění vyjednávacích sil politických stran v poslanecké sněmovně ČR. Výstupem bakalářské práce bude výpočet vyjednávacích sil politických stran a návrh vlastního software potřebného k výpočtu těchto hodnot.

CÍLE PRÁCE, METODY A POSTUPY ZPRACOVÁNÍ

Cíle práce

Hlavním cílem práce je osvojení si základů teorie her s důrazem na vybrané speciální partie kooperativní teorii her. Dalším cílem práce je aplikace těchto poznatků na řešených příkladech. Posledním dílčím cílem je návrh programu v matematickém software Matlab, který usnadní výpočet Myersnovy hodnoty hráčů v koaličních hrách.

Metody a postupy zpracování

Teoretická část práce popisuje koaliční hry a uvádí definice potřebné pro pochopení dané problematiky. Tato část taktéž popisuje typy koaličních her a rozebírá jejích vlastnosti. V závěru této části je definovaná Shapleyho hodnota a s ní související Myersnova hodnota. Dalším krokem je aplikování těchto poznatků na konkrétních situacích, které jsou popsané v analytické části. Návrhová část práce popisuje postup při vytváření vlastního programu pro výpočet Myersnovy hodnoty.

1 TEORETICKÁ VÝCHODÍSKA PRÁCE

Tato kapitola je rozdělena do tří částí. V první si nadefinujeme koaliční hry a rozebereme si některé jejich základní vlastnosti. Především se zaměříme na superaditivitu a konvexnost her. Dále také konstantní součet, monotónnost a symetrií her.

První třída her, které budeme vysvětlovat se jmenuje Market games (tržní hry) a představuje model výměnné ekonomiky s penězi. Dále bude následovat vysvětlování cost allocation games. Detailně se zaměříme na tři příklady: a water supply problem, airport games a minimum cost spinning games. Nakonec vyšetříme základní vlastnosti jednoduchých her. Tyto hry popisují typicky parlament, městskou radu (magistrát), ad hoc komise (pouze pro tento případ, jen k této věci) a tak dále. Vyskytují se také v mnoha aplikacích teorie her na politické vědy.

Poslední sekce je věnována detailnímu rozboru vlastností řešení koaličních her. Uvedeme hlavní axiomy pro řešení her a zvážíme jejich pravděpodobnost. Dále také ukážeme, že tyto axiomy jsou splněné v jádru, což je důležitým řešením pro kooperativní hry.

1.1 Koaliční hry

Nechť U je neprázdná množina hráčů. Množina U může být konečná nebo nekonečná. Koalice je neprázdná konečná podmnožina U. [2]

Definice 1.1.1

Koaliční *hra s přenosným užitkem* je dvojice (N, v), kde N je koalice a v je funkce, která každé podmnožině $S \subseteq N$ přiřadí reálné číslo v(s). Budeme předpokládat, že $v(\emptyset) = 0$. [2]

Poznámka 1.1.2.

Nechť G = (N, v) je koaliční hra. Množina N se nazývá množina hráčů z G a v je koaliční funkce. Nechť S je podkoalice (subcoalition) N. V takovém případě, pak jeho členové obdrží množství v(S) peněz. Číslo *v*(S) se nazývá hodnota koalice S. [2]

Poznámka 1.1.3.

Ve většině aplikací koaličních her jsou hráči osoby nebo skupiny osob, například odbory, města, národy, atd. Nicméně, v některých zajímavých modelech teorie her zabývajících se ekonomickými problémy, hráči nemusí být osobami. Mohou to být plány ekonomických projektů, faktory produkce nebo některé posuzovány ekonomické proměnné. [2]

Předpoklad 1.1.4.

Budeme předpokládat, že užitkové funkce hráčů jsou lineární a rostoucí v závislosti na proměnné peníze. Můžeme dále předpokládat, že všechny tyto funkce mají stejný kladný sklon. Nyní, pokud se vytvoří koalice S, může rozdělit hodnotu koalice v(S) mezi své členy jakoukoli přípustnou cestou, to je taková, jejíž vedlejší platby jsou neomezené. Vzhledem k předchozímu předpokladu, existuje jednoduchá transformace z monetárních vedlejších plateb na odpovídající užitný výplatní vektor. Technicky vzato, můžeme vyjádřit všechny možné rozdělení *v*(S) jako rozdělení užitkové výplaty. V tomto smyslu jsou koaliční hry hrami s přenosným užitkem. Dále můžeme pracovat s koaličními hrami, kde výplata bude v jednotkách užitku. [3]

Definice 1.1.5

Hra (N, v) je *superaditivní*, pokud: [3]

$$(S, T \subseteq N \; a \; S \cap T = \emptyset) \Rightarrow v(S \cup T) \geq v(S) + v(T)$$

Podmínka superaditivity je splněná ve většině aplikací her s přenosným užitkem. Ve skutečnosti, se můžeme dohadovat, zda se S∪T vytvoří. Jestli se členové těchto koalic rozhodnou jednat odděleně nebo jako jedna společná koalice. Pokud se rozhodnou konat odděleně, získají v(S) + v(T), což implikuje (1.1.5.). V praxi bývá poměrně často superaditivita porušena z důvodů antimonopolních zákonů, což v důsledku snižuje zisk koalice S∪T, pokud se vytvoří. Rovněž velké koalice mohou být neefektivní, protože je pro ně mnohem obtížnější se dohodnout. [3]

Definice 1.1.6.

Hra je *slabě superaditivní*, pokud: [3]

$$v(S \cup \{i\}) \geq v(S) + v(\{i\}) \text{ pro všechny } S \subseteq N \; a \; i \notin S.$$

Definice 1.1.7

Hra (N, v) je *konvexní*, jestliže: [4]

$$v(S) + v(T) \leq v(S \cup T) + v(S \cap T) \text{ pro všechny } S, T \subseteq N.$$

Vidíme, že konvexní hra je zároveň superaditivní. Ekvivalentní definici je následující: hra je konvexní, právě tehdy když pro všechny i ∈ N platí: [4]

$$v(S \cup \{i\}) - v(S) \leq v(T \cup \{i\}) - v(T) \text{ pro všechny } S \subseteq T \subseteq N \backslash \{i\}$$

Tento zápis znamená, že hra (N, v) konvexní, právě když mezní příspěvek hráče koalice je monotónní a neklesající. Konvexní hry se vyskytují v některých důležitých aplikacích teorie her. [4]

Definice 1.1.8.

Hra (N, v) je *hra s konstantním součtem*, pokud: [3]

$$v(S) + v(N \backslash S) = v(N) \text{ pro všechny } S \subseteq N$$

Hry s konstantním součtem byly rozsáhle zkoumány v časných pracích teorie her. Velmi často jsou právě politické hry hrami s konstantním součtem.

Definice 1.1.9.

Hra (N, v) je *nepodstatná*, pokud je aditivní, to znamená: [3]

$$v(S) = \sum_{i \in S} v(\{i\}) \text{ pro všechny } S \subseteq N.$$

Nepodstatná hra je triviální z hlediska teorie her. Tedy, pokud každý hráč i ∈ N vyžaduje alespoň v({i}), pak rozdělení v (N) je jednoznačně určeno.

Poznámka 1.1.10

Nechť N je koalice a ℝ budeme značit množinu všech reálných čísel. Dále budeme značit $\mathbb{R}^N$ množinu všech funkcí z N do ℝ. Pokud $x \in \mathbb{R}^N$ a $S \subseteq N$, pak píšeme: [3]

$$x(S) = \sum_{i \in S} x^i \qquad \text{Zřejmě } x(\emptyset) = 0.$$

Poznámka 1.1.11

Nechť N je koalice a $x \in \mathbb{R}^N$. Když využijeme předchozí poznámku, umožní nám uvažovat x jako koaliční funkci. Tedy, (N, x) je koaliční hra dána předpisem: [3]

$$x(S) = \sum_{i \in S} x^i \text{ pro všechny } S \subseteq N.$$

Definice 1.1.12

Řekněme, že dvě hry (N, v) a (N, w) jsou *strategicky ekvivalentní*, pokud existuje $\alpha > 0$ a $\beta \in \mathbb{R}^N$ takové, že: [3]

$$w(S) = \alpha v(S) + \beta(S) \text{ pro všechny a } S \subseteq N$$

Definice 1.1.13

Hra (N, v) je *0-normalizovaná*, pokud: [3]

$$v(\{i\}) = 0 \text{ pro všechny } i \in N.$$

Můžeme říct, že každá hra je strategicky ekvivalentní 0-normalizované hře.

Definice 1.1.14

Hra (N, v) je *monotónní*, pokud: [3]

$$S \subseteq T \subseteq N \Rightarrow v(S) \leq v(T)$$

Definice 1.1.15

Nechť G = (N, v) je hra a nechť π je permutace N. Pak π je symetrie G, pokud $v(\pi(S)) = v(S)$ pro všechny S ⊆ N. Grupa všech symetrií je značená $\mathcal{SYM}(G)$. Hra G je *symetrická*, pokud $\mathcal{SYM}(G)$ je grupa $\mathcal{SYM}_N$ všech permutací N. [3]

Poznámka 1.1.16

Pokud A je konečná množina, potom označíme | A | počet členů této množiny. [3]

1.2 Druhy koaličních her

V této části budou představené některé důležité druhy koaličních her.

1.2.1 Market Games (Tržní hry)

Nechť U je množina hráčů. Trh je čtveřice $(N, \mathbb{R}_+^m, A, W)$. Kde N je koalice (množina obchodníků); $\mathbb{R}_+^m$ je nezáporný m-dimenzionální Euklidovský prostor (prostor komodit); $A = (a^i)_{i \in N}$ je indexovaný soubor bodů z $\mathbb{R}_+^m$ (počáteční vklady); a $W = (w^i)_{i \in N}$ je indexovaný soubor spojitých konkávních funkcí v $\mathbb{R}_+^m$ (užitkové funkce). [3]

Budeme předpokládat, že naše trhy mají přenosný užitek, to znamená, že existuje dodatečná komodita - peníze a každý obchodník měří svůj užitek ze statků těmito penězi. Formálně, užitek obchodníka i ∈ N ze statku $x \in \mathbb{R}_+^m$ a množství peněz $\xi \in \mathbb{R}$ je $W^i(x, \xi) = w^i(x) + \xi$. Množství peněz ξ v předchozí rovnici může být i záporné. Také můžeme předpokládat, že na začátku nemá žádný obchodník peníze. Samozřejmě, pokud je W^i užitková funkce obchodníka i, pak i $W^i + b$, kde $b \in \mathbb{R}$. [3]

Nechť $(N, \mathbb{R}_+^m, A, W)$ je trh, a ať $\emptyset \neq S \subseteq N$. Obchod mezi členy S ve výsledku končí indexovaným souborem $(x^i, \xi^i)_{i \in S}$ kde $x^i \in \mathbb{R}_+^m$ pro všechny $i \in S$, $\sum_{i \in S} x^i = \sum_{i \in S} a^i$ a $\sum_{i \in S} \xi^i = 0$. Celkový užitek koalice S, jako výsledek předchozí transakce, je: [3]

$$\sum_{i \in S} W^i(x^i, \xi^i) = \sum_{i \in S} w^i(x^i) + \sum_{i \in S} \xi^i = \sum_{i \in S} w^i(x^i)$$

To nás vede k následující definici. Přípustná S-alokace je indexována soubor $x_S = (x^i)_{i \in S}$ takový, že $x^i \in \mathbb{R}_+^m$ pro všechny $i \in S$ a $\sum_{i \in S} x^i = \sum_{i \in S} a^i$. Označíme X^S množinu všech přípustných S-alokací. [3]

Definice 1.2.1.

Hra (N, v) je *tržní hra*, pokud existuje trh $(N, \mathbb{R}_+^m, A, W)$ takový, že: [3]

$$v(S) = \max\{\sum_{i\in S} w^i(x^i) \,|x_S \in X^S\}$$

Pro každé S⊆N.

Příklad 1.2.2.

Nechť $N = N_1 \cup N_2$, kde $N_1 \cap N_2 = \emptyset$ a $|N_j| \geq 1$ pro $j = 1{,}2$ a nechť $m = 2$. Pro $i \in N_1$ nechť $a^i = (1{,}0)$ a pro $i \in N_2$ nechť $a^i = (0{,}1)$. Nakonec, nechť $w^i(x_1, x_2) = min\{x_1, x_2\}$ pro všechny $i \in N$. Potom $(N, \mathbb{R}_+^m, A, W)$je trh. Koaliční funkce v odpovídající tržní hře je dána: [3]

$$v(S) = \min\{|S \cap N_1|, |S \cap N_2|\} \text{ pro všechny } S \subseteq N.$$

1.2.2 Cost allocation Games

Nechť U je množina hráčů. Cost allocation problém je hra (N, c) kde N je koalice a koaliční funkce c je nákladová funkce problému. Intuitivní, N reprezentuje množinu potencionálních zákazníků jisté veřejné služby nebo veřejného prostředku. Každý zákazník bude obsloužen na nějaké předem dohodnuté úrovni, nebo nebude obsloužen vůbec. Ať S⊆N. Pak c (S) reprezentuje nejmenší náklad na co nejefektivnější obsloužení členů množiny S. Hra (N, c) se jmenuje cost game (nákladová hra). [3]

Přestože je nákladová hra (N, c) formálně hrou, z jistého hlediska, je to spíš aplikace, protože nákladová funkce není interpretována jako obyčejná koaliční funkce. Je možné spojovat nákladovou hru (N, c) s obyčejnou hrou (N, v) nazývanou savings game, která je dána funkcí $v(S) = \sum_{i\in S} c(\{i\}) - c(S)$ pro všechny $S \subseteq N$. [3]

Nechť (N, c) je nákladová hra a (N, v) je odpovídající savings game. Pak (N, c) je subaditívní, kde platí: [3]

$$(S, T \subseteq N \; a \; S \cap T = \emptyset) \Rightarrow c(S) + c(T) \geq c(S \cup T)$$

Právě tehdy když (N, v) je superaditívní a (N, c) je konkávní. Dále platí: [3]

$$c(S) + c(T) \geq c(S \cup T) + c(S \cap T) \text{ pro všechny } S, T \subseteq N,$$

Právě tehdy když (N, v) je konvexní. V aplikacích nákladových her je obvykle subaditívní.

Příklad 1.2.3. Cost-sharing problem

Skupina míst N zvažuje možnost postavení společné čističky vody. Každý magistrát požaduje minimálně takovou dodávku vody, kterou by si dokázal zajistit i vlastním distribučních systémem nebo systémem, o který by se dělil s některým jiným magistrátem, nebo i všemi ostatními magistráty. Alternativní náklad c (S) koalice S ⊆ N je minimální náklad dodávky členem koalice S nejefektivnějším možným způsobem. Když vezmeme v úvahu fakt, že množina S⊆N může dodávky vody vyřešit několika oddělenými podsystémy, získáváme subaditívní nákladovou hru.[3]

Příklad 1.2.4 Airport games

Vezměme v úvahu letiště s jednou dráhou. Předpokládejme, že máme *m* různých typů letadel, a že c_k, $1 \leq k \leq m$ je cena vybudování dráhy vyhovující letadlu typu *k*. Nechť N_k je množina přistání letadla typu *k* v daném časovém intervalu a řekněme, že $N = \bigcup_{k=1}^{m} N_k$. Tedy, "hráči" (členy množiny N) jsou přistání letadel. Nákladová funkce odpovídající dané airport game je dána: [3]

$$c(S) = max\{c_k | S \cap N_k \neq \emptyset\} \; a \; c(\emptyset) = 0$$

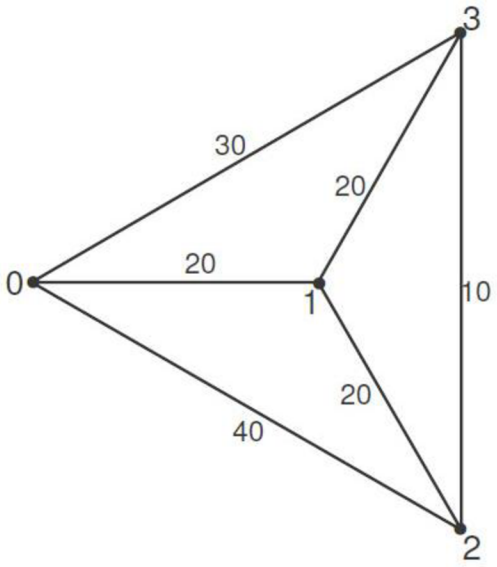
Poznamenejme, že airport game je konkávní.

Příklad 1.2.5 Minimum cost spanning tree games

Skupina N zákazníků, kteří jsou geograficky oddělení a musí být spojení s dodavatelem 0. Například zákazníci mohou být města a dodavatel může být elektrárna. Uživatel může být s dodavatelem spojen přímo, nebo prostřednictvím jiných uživatelů. Ať $N_* = N \cup \{0\}$. Uvažujme neorientovaný graf s množinou uzlových bodů N_*. Cena

spojení i, j ∈ N_*, i ≠ j přes hranu $e_{\{i,j\}}$ je $c_{\{i,j\}}$. Budeme zapisovat e_{ij} namísto $e_{\{i,j\}}$ a c_{ij} namísto $c_{\{i,j\}}$. Pak minimum cost spanning tree game je definována následovně. Ať S ⊆ N. Minimum cost spanning tree $\Gamma_S = (S \cup \{0\}, E_S)$ je strom s množinou uzlových bodů S ∪ {0} a množinou hran E_S, které spojují členy S se společným dodavatelem 0 tak, že celkové náklady všech spojení jsou minimální. Nákladová funkce c nákladové hry (N, c) je definována takto: [3]

$$c(S) = \sum_{e_{ij} \in E_S} c_{ij} \text{ pro všechny } S \subseteq N \ (c(\emptyset) = 0).$$



Obrázek 1: Spanning tree game (Zdroj: [3])

Nyní uvažujme následující příklad. Nechť N = {1,2,3} a nechť náklad jednotlivých hran odpovídá obrázku 1.

Nákladová funkce je dána následovně:

$$(S) = \begin{cases} 0, pro\ S = \emptyset \\ 20, pro\ S = \{1\} \\ 30, pro\ S = \{3\} \\ 50, pro\ S = N \\ 40, \quad jinak \end{cases}$$

1.3 Jednoduché hry

Nechť U je množina hráčů.

Definice 1.3.1.

Jednoduchá hra je dvojice (N, W), kde N je koalice a W je množina podmnožin N splňující:[3]

$$N \in W$$

$$\emptyset \notin W$$

$$(S \subseteq T \subseteq N \; a \; S \in W) \Rightarrow T \in W$$

Soubor W koalic je množina vítězných koalic.

Vlastnost plynoucí z této definice představuje monotónnost jednoduché hry. Intuitivní, jednoduchá hra g = *(N,W)* reprezentuje komisi: Koalice N je množina členů komise a W je množina koalic, které jsou plně pod vlivem rozhodnutí g. Poznamenejme, že každý parlament je komise, každá městská rada je komise a tak dále. [3]

Definice 1.3.3

Nechť g = (N, W) je jednoduchá hra.

Jednoduchá hra g je $\begin{Bmatrix} vyhovující \\ silná \\ slabá \end{Bmatrix}$ když $\begin{Bmatrix} S \in W \Rightarrow N \backslash S \notin W \\ S \notin W \Leftrightarrow N \backslash S \in W \\ V = \bigcap_{S \in W} S \neq \emptyset \end{Bmatrix}$

Členové V se jmenují vetující. Jednoduchá hra g je diktátorská, pokud existuje j ∈ N ("diktátor") takový, že: [3]

$$S \in W \Leftrightarrow j \in S$$

Poznámka 1.3.4.

Nechť g = (N, W) je jednoduchá hra. V mnoha aplikacích je vhodné s g spojovat koaliční hru G = (N, v), kde v(S) = 1, pokud S∈W a v(S) = 0 v jiném případě. Například, toto je případ, kdy komise g musí rozdělit fixní množství peněz mezi své členy. Tento fakt vede k následující definici. [3]

Definice 1.3.5.

Nechť g = (N, W) je jednoduchá hra. *Associated Coalition game with a simple game* (Spojená koaliční hra s jednoduchou hrou) (N, v) je dána:[3]

$$v(S) = \begin{cases} 1 \; pokud \; S \in W \\ 0 \qquad\quad jinak \end{cases}$$

Nechť g = (N, W) je jednoduchá hra a nechť G = (N, v) je spojena koaliční hra. Pak hra G je monotónní. G je také superaditivní právě když g je vyhovující a G je hra s konstantním součtem právě když g je silná.[3]

Všimněme si, že jakákoliv monotónní koaliční hra (N, v), která splňuje v(S) ∈ {0,1} pro všechny S ⊆ N a v(N) = 1 je spojena hra s nějakou jednoduchou hrou. [3]

Definice 1.3.6

Jednoduchá hra je *symetrická*, pokud spojena hra je symetrická. Tedy, jednoduchá hra g = (N, W) je symetrická, jestliže: [3]

$$(S \in W, T \subseteq N, a \; |T| = |S|) \Rightarrow T \in W$$

Definice 1.3.7

Jednoduchá hra (N, W) je *weighted majority game* (vážená většinová hra), pokud existuje kvóta $q > 0$ a váhy $w^i \geq 0$ pro všechny $i \in N$ takové, že pro všechny $S \subseteq N$ platí: [3]

$$S \in W \Leftrightarrow w(S) > q$$

Nechť g = (N, W) je vážená většinová hra s kvótou q> 0 a váhami $w^i \geq 0$ pro všechny i ∈N.

(| N | +1)-tice $(q; (w^i)_{i \in N})$ se nazývá reprezentant g a zapisujeme $g \mathrel{\widehat{=}} (q; (w^i)_{i \in N})$.

Poznámka 1.3.8

Jestliže g = (N, W) je jednoduchá hra, značíme: [3]

$$W^m = \{S \in W | T \subsetneq S \Rightarrow T \notin W\}$$

množinu minimálních vítězných koalic.

Definice 1.3.9.

Nechť g = (N, W) je vážená většinová hra. Reprezentant $(q; (w^i)_{i \in N})$ g je homogenní reprezentant g, pokud: [3]

$$S \in W^m \Rightarrow w(S) = q$$

Vážená většinová hra je homogenní, pokud má homogenního reprezentanta.

Poznámka 1.3.10

Symetrická jednoduchá hra g = (N, W) má homogenního reprezentanta $(k; 1, .., 1)$, kde k označujeme obyčejnou velikost každé minimální vítězné koalice. Takovou hru také označujeme (n, k), kde n = |N|. [3]

Příklad 1.3.11

Rada bezpečnosti OSN je dána hrou

$$g \widehat{=} (39; 7,7,7,7,7,1,1,1,1,1,1,1,1,1,1)$$

Tato hra je slabá (vetující je Velká pětka) a homogenní.

1.4 Vlastnosti a řešení

Nechť U je množina hráčů a *(N, v)* je hra. Označíme: [3]

$$X^*(N,v) = \{x \in \mathbb{R}^N | x(N) \leq v(N)\}.$$

Množina $X^*(N,v)$ je množina všech přípustných výplatních vektorů hry (N,v).

Definice 1.4.1

Nechť Γ je množina her. Řešení na množině Γ je funkce σ, která spojuje s každou hrou $(N,v) \in \Gamma$ podmnožinu $\sigma(N,v) \subseteq X^*(N,v)$. [5]

Intuitivní, řešení je vymezeno systémem "rozumných" omezení v souladu s $X^*(.,.)$. Například, můžeme předepsat určité nerovnosti, které garantují "stabilitu" členů σ (N, v). Alternativně, σ může být charakterizovány skupinou axiomy. Poznamenejme, že každý člen σ (N, v) je považován za možné konečné výplatní rozdělení (N, v).

Definice 1.4.2

Jádro hry (N, v), zapisujeme $\mathcal{C}(N,v)$, je definováno: [5]

$$\mathcal{C}(N,v) = \{x \in X^*(N,v) | x(S) \geq v(S) \; pro \; všechny \; S \subseteq N\}$$

Nechť $x \in X^*(N,v)$. Pak $x \in \mathcal{C}(N,v)$ právě když si žádná koalice nemůže polepšit nad x. Tedy, každý člen jádra má vysoce stabilní výplatní rozdělení.

1.5 Shapleyho hodnota

Nechť *v* označuje charakteristickou funkci, která přiřazuje ke každé koalici S ⊆ V, reálné číslo představující její výplatu nebo hodnotu. Píšeme $2^v \rightarrow \mathbb{R}$. Kooperativní hra ve formě charakteristické funkce je daná dvojicí $\langle V,v \rangle$. Budeme předpokládat, že všichni hráči vytvořili koalici V, potom řešením koaliční hry je metoda rozdělení v(V) zisku koalice mezi hráče. Jedno z takových nejvýznamnějších řešení je Shapleyho hodnota. Shapleyho hodnota je v principu vážený průměr marginálního příspěvku hráče do každé koalice, do které může patřit. Vypočítáme ji vzorcem: [6]

$$\varphi_i(v) = \sum_{S \subset V, v_i \in S} \frac{(|S|-1)! - (|N|-|S|)!}{|N|!} * (v(S) - v(S - \{i\}))$$

1.6 Myersnova hodnota

Podle Myerse je přirozené, že některé koalice nemůžou vzniknout vzhledem k tomu, že hráči v těchto koalicích nemůžou spolu spolupracovat. [7]

Předpokládejme, že $g = 2^N$ je množina všech přípustných koalic. A definujme $\bar{g}$ tak, že $S \cap T \neq \emptyset \Rightarrow S \cup T \in \bar{g}$. Pro takto vzniklé koalice definujeme odlišnou výplatní funkci v‘(s), podle které budeme určovat výplatu těchto koalic. [6, 7]

Koalice, které nemůžou nastat, nazveme *restrikce*. Nechť S ⊂ N je libovolná koalice. Množina g-komponent S je definovaná: [6]

$$C_g(S) = \{T \in \bar{g} \mid T \subset S \; a \; neexistuje \; R \in g: \; T \subsetneq R \subset S\}$$

Potom *g-restrikce* hry v je dána $v_g : 2^N \rightarrow \mathbb{R}$ a platí: [6]

$$v_g(S) = \sum_{T \in C_g(S)} v(T)$$

Myersnova hodnota je potom Shapleyho hodnota pro výplatní funkci v‘(s). [6, 7]

2 ANALÝZA SOUČASNÉHO STAVU

V této části budeme aplikovat poznatky z teoretické části na příkladech. Na začátku na jednoduchém příkladu vysvětlíme postup při výpočtu Shapleyho i Myersnovy hodnoty.

V druhé částí této kapitoly se budeme věnovat koaliční hře z prostředí české politiky. Konkrétně se budeme zabývat hlasováním v poslanecké sněmovně. Poslanecká sněmovna ČR je tvořená celkem 200 poslanci, přičemž každý z nich spadá pod určitou politickou stranu a každá z těchto stran má odlišné politické programy a cíle. Pro prosazení libovolného zákona je nutné, aby ho schválilo více než 50% přítomných poslanců. [8]

Koaliční hra z Poslanecké sněmovny bude rozdělena do tří podkapitol, kde v první podkapitole se budeme věnovat popisu složení poslanecké sněmovny. V druhé podkapitole budeme vysvětlovat sestavení koaliční hry a v poslední částí budeme interpretovat získané výsledky.

2.1 Výpočet Myersnovy hodnoty

Pro popsání výpočtů a rozdílů mezi Shapleyho a Myersnovou hodnotou sestavíme jednoduchý vzorový příklad. Uvažujme akciovou společnost o čtyřech akcionářích, kde každý z nich má ve firmě jiný podíl. Jejich podíly jsou 19%, 21%, 25% a 35%. Pro schválení libovolného návrhu při hlasování valné hromady je potřeba, aby pro daný návrh hlasovali akcionáři s podílem alespoň 50%. Úkolem je tedy vypočítat Shapleyho popřípadě Myersnovu hodnotu každého hráče.

Vzhledem k tomu, že zatím nemáme definované žádné restrikce, budeme ze začátku počítat Shapleyho hodnotu každého hráče.

Sestavíme tedy váhový vektor v = (50%; 19%, 21%, 25%, 35%) který udává minimální hodnotu pro výhru koalice a váhy jednotlivých hráčů.

Budeme uvažovat, že síla koalice je daná počtem hlasů jednotlivých akcionářů v případě, že mají dohromady větší než 50% podíl. V případě, že budou mít dohromady podíl 50% a méně, síla této koalice bude nulová. Sestavíme tedy výplatní funkci pro možné koalice.

$$v(s) = \begin{cases} \sum_{i,v_i \in S} v_i & pokud \quad \sum_{i,v_i \in S} v_i > 50 \\ 0 & jinak \end{cases}$$

Vypočítáme tedy sílu jednotlivých možných koalic.

v({1,4}) = 54
v({2,4}) = 56
v({3,4}) = 60
v({1,2}) = 0
v({1,3}) = 0
v({2,3}) = 0
v({1,2,3}) = 65
v({1,2,4}) = 75
v({1,3,4}) = 79
v({2,3,4}) = 81
v({1,2,3,4}) = 100

Dosazením do vzorce z kapitoly 1.5 již můžeme vypočítat Shapleyho hodnotu každého hráče. Neznáma |N| vyjadřuje celkový počet hráč, neznáma |S| vyjadřuje počet hráčů v koalici, v(s) potom uvádí sílu koalice a v(s-{i}) určuje sílu koalice bez i-tého hráče.

Vypočítáme tedy Shapleyho hodnotu pro jednotlivé hráče. Pro výpočet hodnoty prvního hráče budeme počítat s jeho vítěznými koalicemi, tedy koalice, ve kterých se první hráč nachází, jsou to koalice {1,4},{1,2,3},{1,2,4},{1,3,4},{1,2,3,4}.

$$\varphi_1(v) = \frac{(2-1)!\cdot(4-2)!}{4!} \cdot 54 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot 65 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot (75 - 56) +$$
$$+ \frac{(2-1)!\cdot(4-3)!}{4!} \cdot (79 - 60) + \frac{(4-1)!\cdot(4-4)!}{4!} \cdot (100 - 81) = 17.83$$

Výherní koalice hráče 2 jsou {2,4},{1,2,3},{1,2,4},{2,3,4},{1,2,3,4}.

$$\varphi_2(v) = \frac{(2-1)!\cdot(4-2)!}{4!} \cdot 56 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot 65 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot (75 - 54) +$$
$$+ \frac{(2-1)!\cdot(4-3)!}{4!} \cdot (81 - 60) + \frac{(4-1)!\cdot(4-4)!}{4!} \cdot (100 - 79) = 18.83$$

Výherní koalice třetího hráče jsou {3,4},{1,2,3},{1,3,4},{2,3,4},{1,2,3,4}

$$\varphi_3(v) = \frac{(2-1)!\cdot(4-2)!}{4!} \cdot 60 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot 65 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot (79 - 54) +$$
$$+ \frac{(2-1)!\cdot(4-3)!}{4!} \cdot (81 - 56) + \frac{(4-1)!\cdot(4-4)!}{4!} \cdot (100 - 75) = 20.83$$

Stejným způsobem vypočítáme Shapleyho hodnotu čtvrtého hráče. Jeho výherní koalice jsou {1,4},{2,4},{3,4},{1,2,4},{1,3,4},{2,3,4},{1,2,3,4}

$$\varphi_4(v) = \frac{(2-1)!\cdot(4-2)!}{4!} \cdot 54 + \frac{(2-1)!\cdot(4-2)!}{4!} \cdot 56 + \frac{(2-1)!\cdot(4-2)!}{4!} \cdot 60 +$$
$$+ \frac{(3-1)!\cdot(4-3)!}{4!} \cdot 75 + \frac{(3-1)!\cdot(4-3)!}{4!} \cdot 79 + \frac{(2-1)!\cdot(4-3)!}{4!} \cdot 81 + \frac{(4-1)!\cdot(4-4)!}{4!} \cdot$$
$$\cdot (100 - 65) = 42.5$$

Postup výpočtu Shapleyho hodnoty již máme vysvětlený, interpretaci výsledů se budu věnovat u koaliční hry zachycující situaci s reálnými daty. Nyní si tento příklad lehce poupravíme a vysvětlíme na něm výpočet Myersnovy hodnoty.

Uvažujme tedy situaci z předchozího příkladu a přidáme podmínku, že hráč 1 a 4 spolu v koalici být nechtějí. Tím pádem z množiny přípustných koalic odebereme koalice {1,4},{1,2,4},{1,3,4},{1,2,3,4}. Ta bude tedy obsahovat tyto koalice g = {{1},{2},{3},{4},{1,2},{1,3},{2,3},{3,4},{1,2,3},{2,3,4}}. Pro výpočet Myersnovy hodnoty nyní musíme udělat sjednocení dvojice množin, ve kterých je alespoň jeden hráč stejný. Tímto způsobem nám vzniknou 3 nové koalice:

{1,2,4} = {1,2} ∪ {2,4}

{1,3,4} = {1,2} ∪ {2,3,4}

{1,2,3,4} = {1,2} ∪ {2,3,4}

Postup opakujeme i s nově vzniklými koalicemi, nicméně v tomto případě žádná nova koalice tímto sjednocením nevznikne. Množinu takových koalic značíme $\bar{g}$ a sestavíme pro ně zvláštní výplatní funkci v′(s). Můžeme sílu těchto koalic vynásobit nějakým koeficientem 0 < k < 1 a tím snížit sílu takových koalic, protože se dá očekávat, že v těch koalicích bude více docházet k neshodám hráčů. Pro tento příklad si můžeme zvolit k = 0.8. V případě, že bychom zvolili koeficient k = 0, dostali bychom v dalším výpočtu Shapleyho hodnotu jednotlivých hráčů pro přípustné koalice, nikoli Myersnovu. Výplatní funkce koalic z $\bar{g}$ bude tedy vypadat následovně.

$$v'(s) = \begin{cases} \sum_{i, v_i \in S} v_i * k & pokud \quad \sum_{i, v_i \in S} v_i * k > 50 \\ 0 & jinak \end{cases}$$

Vypočítáme tedy vítězné koalice podle výplatních funkcí *v(s)* a *v'(s).*

v({2,4}) = 56

v({3,4}) = 60

v({1,2,3}) = 65

v({2,3,4}) = 81

v '({1,2,4}) = 60

v '({1,3,4}) = 63.2

v '({1,2,3,4}) = 80

Zde si můžeme všimnout, že přítomnost hráče 1 v koalici {1,2,3,4} je poněkud nežádoucí, protože má menší sílu, než když v ní hráč 1 není.

Vypočítáním Shapleyho hodnoty pro vítězné koalice z množiny přípustných koalic g i z množiny $\bar{g}$ dostaneme Myersnovu hodnotu. Vypočítáme tedy Myersnovu hodnotu jednotlivých hráčů.

$$\varphi_1(v) = \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 65 + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot(60 - 56) + \frac{(2-1)!\cdot(4-3)!}{4!}\cdot(63.2 - 60) + \\ + \frac{(4-1)!\cdot(4-4)!}{4!}\cdot(80 - 81) = 5.77$$

$$\varphi_2(v) = \frac{(2-1)!\cdot(4-2)!}{4!}\cdot 56 + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 65 + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 60 + \\ + \frac{(2-1)!\cdot(4-3)!}{4!}\cdot(81 - 60) + \frac{(4-1)!\cdot(4-4)!}{4!}\cdot(100 - 63.2) = 21.03$$

$$\varphi_3(v) = \frac{(2-1)!\cdot(4-2)!}{4!}\cdot 60 + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 65 + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 63.2 + \\ + \frac{(2-1)!\cdot(4-3)!}{4!}\cdot(81 - 56) + \frac{(4-1)!\cdot(4-4)!}{4!}\cdot(100 - 60) = 22.77$$

$$\varphi_4(v) = \frac{(2-1)!\cdot(4-2)!}{4!}\cdot 56 + \frac{(2-1)!\cdot(4-2)!}{4!}\cdot 60 + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 60 + \\ + \frac{(3-1)!\cdot(4-3)!}{4!}\cdot 63.2 + \frac{(2-1)!\cdot(4-3)!}{4!}\cdot 81 + \frac{(4-1)!\cdot(4-4)!}{4!}\cdot(100 - 65) = 30.43$$

Nyní můžeme srovnat Myersnovu i Shapleyho hodnotu jednotlivých hráčů a zjistit, jaký vliv na sílu jednotlivých hráčů měla restrikce koalice {1,4}.

Tabulka 1: Srovnání Myersnovy a Shapleyho hodnoty (Zdroj: vlastní)

Hráč	Shapleyho hodnota	Myersnova hodnota
1	17.83	5.77
2	18.83	21.03
3	20.83	22.77
4	42.5	30.34

Z tabulky srovnání Shapleyho a Myersnovy hodnoty můžeme vidět, že neshody mezi nejslabším a nejsilnějším hráčem můžou mít poměrně velký dopad na vyjednávací sílu nejsilnějšího hráče. Podrobnější interpretaci významu hodnot se budu zabývat v dalším příkladu.

2.2 Poslanecká sněmovna

V této kapitole si popíšeme fungování poslanecké sněmovny ČR, které politické strany jsou aktuálně ve sněmovně a jaký mají podíl. Na základě těchto informací sestavíme koaliční hru a vypočítáme Myersovu hodnotu těchto stran včetně interpretace výsledků.

Jak už bylo zmíněno, poslanecká sněmovna má celkem 200 poslanců a aktuálně jsou ve sněmovně tyto politické strany: ČSSD, ANO, KSČM, TOP09, ODS, ÚSVIT, KDU-ČSL. Jedná se o strany, které při volbách v roce 2013, získaly alespoň 5% hlasů. Níže uvedená tabulka popisuje aktuální politické strany působící v poslanecké sněmovně i počet poslanců, kterými ve sněmovně disponují. [8]

Tabulka 2: Poslanecká sněmovna (Zdroj: [9])

Politická strana	Počet poslanců	Procentuální podíl
ČSSD	50	25%
ANO	47	23,5%
KSČM	33	16,5%
TOP09	26	13%
ODS	16	8%
ÚSVIT	14	7%
KDU-ČSL	14	7%

Aby byl libovolný návrh ve sněmovně přijat, je nutné, aby pro něj hlasovala nadpoloviční většina přítomných poslanců. Pro naši koaliční hru budeme uvažovat, že při neúčasti poslanců na hlasováních bude procentuální podíl přítomných poslanců jednotlivých politických stran odpovídat jejích procentuálnímu podílů z tabulky 2. [8]

2.2.1 Sestavení koaliční hry

Z předchozí podkapitoly tedy známe složení poslanecké sněmovny. Pro sestavení koaliční hry je dále nutné vědět, které strany koalice tvořit nebudou – ať už z důvodu odlišných politických programů nebo jiných důvodů. Pro tuto hru jsem zvolil strany, které spolu nebudou spolupracovat následovně:

ČSSD × ODS

ANO × KSČM

TOP09 × KSČM

ODS × KSČM

Následně každé politické straně přiřadíme číslo, které ji bude reprezentovat:

Tabulka 3: Rozřazení politických stran (Zdroj: vlastní)

Číslo	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Strana	ČSSD	ANO	KSČM	TOP09	ODS	ÚSVIT	KDÚ-ČSL

Váhový vektor $v = (q; v_1, v_2, \ldots, v_i)$, kde q je minimální hodnota potřebná pro schválení zákona a $v_1, \ldots, v_i$ jsou jednotlivé podíly hráčů, potom vypadá následovně:

$v = (50; 25, 23.5, 16.5, 13, 8, 7, 7)$

Sestavíme množinu přípustných koalic:

g = {{1};{2};{3};{4};{5};{6};{7};{1,2};{1,3};{1,4};{1,6};{1,7};{2,4};{2,5};{2,6};{2,7};{3,6};{3,7};{4,5};{4,6};{4,7};{5,6};{5,7};{6,7};{1,2,4};{1,2,6};{1,2,7};{1,3,6};{1,3,7};{1,4,6};{1,4,7};{1,6,7};{2,6,7};{2,5,7};{2,5,6};{2,4,7};{2,4,6};{2,4,5};{3,6,7};{4,5,6};{4,5,7};{4,6,7};{5,6,7};{4,5,6,7};{2,5,6,7};{2,4,6,7};{2,4,5,7};{2,4,5,6};{1,4,6,7};{1,3,6,7};{1,2,6,7};{1,2,4,7};{1,2,4,6};{2,4,5,6,7};{1,2,4,6,7}}

Podle definice z kapitoly 1.6. následně dopočítáme koalice, které vzniknou sjednocením přípustných koalic.

$\bar{g}$ = {g;{1,2,3};{1,2,5};{1,2,3,6};{1,2,3,7};{1,2,5,7};{1,2,5,6};{1,2,4,5};
{1,2,5,6,7};{1,2,4,5,7};{1,2,4,5,6};{1,2,3,6,7};{1,2,4,5,6,7};{1,3,4};{1,2,3,4};
{1,3,4,6};{1,3,4,7};{1,3,4,6,7};{1,2,3,4,7};{1,2,3,4,6};{1,2,3,4,6,7};{1,4,5};{1,4,5,6};
{1,4,5,7};{1,4,5,6,7};{1,5,6};{1,5,6,7};{1,5,7};{2,3,6};{2,3,6,7};{2,3,7};{3,4,6};
{3,5,6};{2,3,5,6};{2,3,4,6};{3,4,5,6};{3,4,6,7};{3,5,6,7};{3,4,5,6,7};{2,3,5,6,7};
{2,3,4,6,7};{2,3,4,5,6};{2,3,4,5,6,7};{3,4,7};{3,5,7,};{2,3,5,7};{2,3,4,7};{3,4,5,7};
{2,3,4,5,7};{1,3,5,6};{1,3,5,6,7};{1,3,5,7};{1,2,3,5,6};{1,3,4,5,6};{1,3,4,5,6,7};
{1,2,3,5,6,7};{1,2,3,4,5,6};{1,2,3,4,5,6,7};{1,2,3,5,7};{1,3,4,5,7};{1,2,3,4,5,7}}

Nyní je potřeba definovat výplatní funkci koalic. Budeme uvažovat, že síla koalice bude rovna počtu hlasů, kterými disponuje. Dále z praxe víme, že i když politické strany vytvoří koalici, ne vždy se úplně domluví a většinou se stává, že v daném hlasování nezíská hlasy všech svých poslanců. Proto tedy sílu těchto koalic vynásobíme konstantou $k \in (0,1\rangle$. Pro výplatní funkci v(s) pro přípustné koalice jsem zvolil konstantu $k_1 = 0.95$, která říká, že při hlasování tyto koalice získají 95% hlasů svých poslanců. U koalic vzniklých sjednocením, lze očekávat větší rozpory poslanců při spolupráci s jinými stranami, proto jsem zde pro výplatní funkci v´(s) zvolil konstantu $k_2 = 0.75$. Podle předchozích předpokladů nyní definujeme výplatní funkce koalic.

$$v(s) = \begin{cases} \sum_{i,v_i \in S} v_i * k_1 & pokud \quad \sum_{i,v_i \in Si} v_i * k_1 > q \\ 0 & jinak \end{cases}$$

$$v'(s) = \begin{cases} \sum_{i,v_i \in S} v_i * k_2 & pokud \quad \sum_{i,v_i \in S} v_i * k_2 > q \\ 0 & jinak \end{cases}$$

Z údajů které máme k dispozici, můžeme určit vítězné koalice a jejích sílu.

v({1,2,4}) = 58.425
v({1,2,6}) = 52.725
v({1,2,7}) = 52.725
v({1,3,6,7}) = 52.725
v({1,2,6,7}) = 59.375
v({1,2,4,7}) = 65.075
v({1,2,4,6}) = 65.075
v({2,4,5,6,7}) = 55.575
v({1,2,4,6,7}) = 71.725
v′({1,2,3,6}) = 54
v′({1,2,3,7}) = 54
v′({1,2,4,5}) = 52.125
v′({1,2,5,6,7}) = 52.875
v′({1,2,4,5,7}) = 57.375
v′({1,2,4,5,6}) = 57.375
v′({1,2,3,6,7}) = 59.25
v′({1,2,4,5,6,7}) = 62.625
v′({1,2,3,4}) = 58.5
v′({1,3,4,6,7}) = 51.375
v′({1,2,3,4,7}) = 63.75
v′({1,2,3,4,6}) = 63.75
v′({1,2,3,4,6,7}) = 69
v′({2,3,4,6,7}) = 50.25
v′({2,3,4,5,6}) = 51
v′({2,3,4,5,6,7}) = 56.25
v′({2,3,4,5,7}) = 51
v′({1,2,3,5,6}) = 60
v′({1,3,4,5,6}) = 52.125
v′({1,3,4,5,6,7}) = 57.375
v′({1,2,3,5,6,7}) = 65.25
v′({1,2,3,4,5,6}) = 69.75
v′({1,2,3,4,5,6,7}) = 75
v′({1,2,3,5,7}) = 60
v′({1,3,4,5,7}) = 52.125
v′({1,2,3,4,5,7}) = 69.75

Dosazením do vzorce z kapitoly 1.5 spočítáme Myersnovu hodnotu každého hráče. Pro připomenutí, postup výpočtu je detailněji popsaný ve vzorovém příkladu. V následující tabulce je vypočtena Myersnova hodnota výše uvedených politických stran.

Tabulka 4: Myersnova hodnota politických stran (Zdroj: vlastní)

Strana	Myersova hodnota
ČSSD	18.6411
ANO	19.3149
KSČM	8.4130
TOP09	10.9053
ODS	3.7090
ÚSVIT	8.1628
KDU-ČSL	8.1628

2.3 Interpretace výsledků

Myersnova hodnota jednotlivých hráčů vypovídá o jejích vyjednávací síle. Můžeme ji aplikovat například na politické programy jednotlivých stran.

Výsledky budu interpretovat na současné koaliční vládě složené z ČSSD, ANO a KDU-ČSL. Programy a cíle politický stran se většinou liší a pomoci Myersovy hodnoty můžeme určit, na kolik procent by se měly plnit programy jednotlivých stran v rámci koalice. Vezmeme tedy Myersnovu hodnotu těchto politických stran a vypočítáme podíl jednotlivých stran na součtu Myersnových hodnot celé koalice. Výpočet je proveden v následující tabulce.

Tabulka 5: Interpretace výsledků (Zdroj: vlastní)

Politická strana	Myersnová hodnota	Podíl
ČSSD	18.6411	40.4%
ANO	19.3149	41.9%
KDÚ-ČSL	8.1682	17.7%
∑	46,1188	100%

Strana ČSSD by tedy měla mít v rámci koalice nárok na splnění 40.4% svých cílů, strana ANO 41.9% a strana KDÚ-ČSL by měla mít možnost splnit 17.7% svých cílů.

2.4 Shrnutí analytické části

V této části byly aplikované poznatky z teoretické částí na vzorovém příkladu, na kterém byl vysvětlen postup výpočtu Shapleyho a Myersnovy hodnoty i rozdíly mezi těmito hodnotami. Následně byla sestavená koaliční hra zachycující aktuální situaci v poslanecké sněmovně. Byly stanovené Myersovy hodnoty jednotlivých politických stran a následně byl popsán její význam.

Už na vzorovém příkladu o čtyřech hráčích jsme mohli pozorovat, že výpočet těchto hodnot je poměrně náročný a zdlouhavý. Na příkladu se 7 hráči, respektive politickými stranami, jsme měli 55 přípustných koalic a 60 koalic vzniklých sjednocením. Manuální počítání Myersnovy hodnoty by proto bylo velmi časově náročné.

Na příklady tohoto typu by bylo vhodné mít specializovaný software, který by na základě určitých informací dokázal hledané hodnoty vypočítat. V návrhové části se proto budu věnovat tvorbě takového programu, který na základě počtu hráčů a znalosti zakázaných koalic vypočítá Myersnovu popřípadě Shapleyho hodnotu každého hráče.

3 VLASTNÍ NÁVRHY ŘEŠENÍ

V analytické části jsme mohli vidět, že výpočet Myersnovy hodnoty je vzhledem k velkému množství možných koalic poměrně náročný. Součástí bakalářské práce je proto naprogramování aplikace v matematickém programu Matlab pro výpočet Myersnovy hodnoty. Aplikace bude obsahovat uživatelské rozhraní, ve kterém si uživatel bude moci jednoduše nastavit počet hráčů, váhu jednotlivých hráčů, minimální hodnotu potřebnou pro výhru koalice a koeficienty pro výpočet výplatních funkcí.

V této kapitole popíšu postup při programování této aplikace, od vytvoření uživatelského prostředí až po samotný výpočet Myersnovy hodnoty. Popsané budou taktéž jednotlivé funkce, které bylo nutné v rámci aplikace vytvořit, včetně jejích použití. Celý postup bude taky doplněn o obrázky aplikace po jednotlivých blocích kódu.

V závěru kapitoly budou přiložené obrázky z výpočtu Myersnovy hodnoty pro příklad z analytické části. Tento výpočet bude realizovaný v programu, který jsem navrhnul.

3.1 Vytvoření uživatelského prostředí

Na začátku musíme vytvořit okno, ve kterém bude celý program pracovat, toto okno si uložíme do proměnné `mainwindow` a nadefinujeme ho jako `figure` s níže uvedenými parametry.

```
ScreenSize=get(0,'ScreenSize');
    mainwindow=figure('Name','Myersons value',...
                      'NumberTitle','Off',...
                      'Menubar','none',...
                      'Resize','off',...
                      'Units','pixels',...
                      'Position',[0.5*(ScreenSize(3)-800),...
                                  0.5*(ScreenSize(4)-600),...
                                  800,600],...
                      'color',[0.9 0.9 0.9]);
```

Parametr `'Name'` nastaví název okna, který následně vidíme v horní liště vygenerovaného okna.

Parametr `'NumberTitle'` s hodnotou `'Off'` zajistí, že v názvu okna nebude před názvem nastaveným parametrem `'Name'` zobrazován popis „figure(x):“, kde x značí pořadí vygenerovaného okna.

Pomocí `'Menubar','none'` zakážeme zobrazení menu s defaultními ovládacími prvky.

`'Units','pixels'` říká, že číselné hodnoty pro pozici a velikost okna budou zadávané v pixelech.

Okno bude mít velikost 800px × 600px a při zobrazení bude vždy uprostřed obrazovky. To nám zajistí proměnná `ScreenSize`, která pomocí funkce `get(0,'ScreenSize')` získá velikost obrazovky. Tato funkce jako výstupní hodnotu vrací pole hodnot, kde na 3. místě najdeme šířku monitoru a na 4. pozici výšku monitoru. Následně můžeme vypočítat souřadnice středu obrazovky a pomoci parametru `'Position'` nastavíme pozici na ose x, pozici na ose y, šířku, výšku okna.

Parametr `'color'` nastaví barvu pozadí zobrazeného okna.

3.1.1 Pole pro nastavení počtu hráčů

V první řadě je potřeba zjistit, s kolika hráči budeme počítat. Proto do již vytvořeného okna přidáme možnost nastavení počtu hráčů a tlačítko pro potvrzení volby. Jako první vytvoříme panel s popisem, do kterého následně vložíme pole pro zadání hodnoty a tlačítko pro uložení.

```
pocetHracuPanel = uipanel('Title','Počet hráčů',...
                          'Units','pixels',...
                          'Pos',[290,545,220,55]);
```

Tento příkaz vytvoří již zmiňovaný panel pomocí příkazu `uipanel();` a uloží ho do proměnné `pocetHracuPanel`. Parametr `'Title'` následně zajistí popis tohoto panelu.

```
numberOfPlayers = uicontrol('Parent',pocetHracuPanel,...
    'Style','edit',...
    'position',[10,10,100,25],...
    'background','white');
```

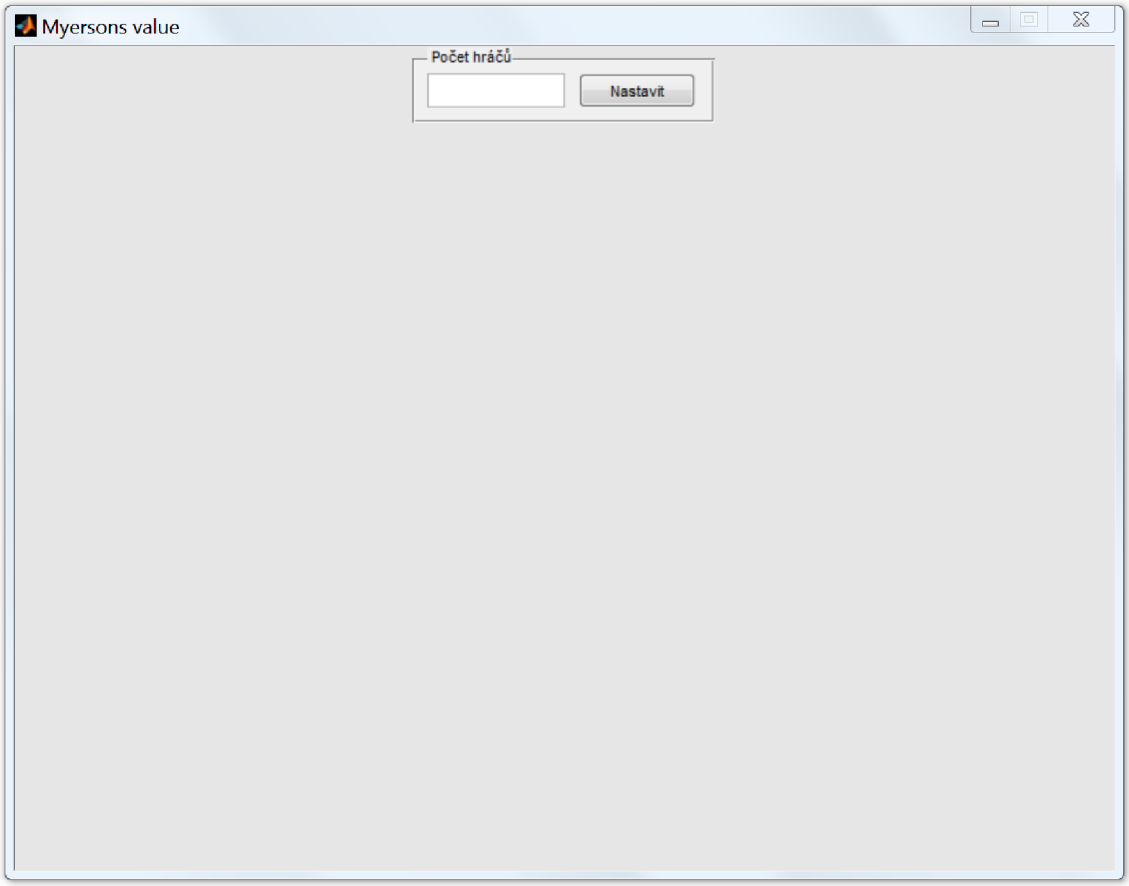
Výše uvedený kód vytvoří textové pole pro zadávání hodnot pomocí příkazu `uicontrol()` a uloží ho do proměnné `numberOfPlayers`.

`'Parent',pocetHracuPanel` nám zajistí, že budeme zadávat pozici tohoto pole vůči již vytvořenému panelu. Pomoci `'Style','edit'` nám říká, že se jedná o textové pole, do kterého můžeme zadávat hodnoty a `'background','white'` nám nastaví barvu pozadí tohoto pole na bílou.

```
uicontrol('String','Nastavit',...
    'Parent',pocetHracuPanel,...
    'position',[120,10,85,25],...
    'callback',@FsetPlayers);
```

Tento příkaz vytvoří zmiňované tlačítko pro nastavení počtu hráčů. Text zobrazovaný uvnitř tlačítka určuje `'String','Nastavit'` a funkci, která se má zavolat po stisknutí tlačítka nastavuje parametr `'callback'`. Samotná funkce `FsetPlayers` bude popsána následně.

Okno se zmiňovanými ovládacími prvky, které vytvoří výše uvedené příkazy, můžeme vidět na následujícím obrázku.



Obrázek 2: Tvorba uživatelského prostředí (Zdroj: vlastní)

3.1.2 Nastavení parametrů hry

Nastavení počtu hráčů bude tedy provádět již zmiňovaná funkce `FsetPlayers`, která také vykreslí tabulku pro zadávání počtu váhy jednotlivých hráčů, nastavení přípustných koalic, pole pro minimální hodnotu pro výhru a koeficientu výplatní funkce v(s) a v´(s).

Nyní popíšu provedení jednotlivých kroků. Na začátku je potřeba definovat samotnou funkci následovně.

```
function FsetPlayers(~,~)
...
end
```

Parametry funkce jsou nastavené na ~,~ z toho důvodu, že tuto funkci voláme callbackem.

Prvním krokem funkce bude uložení počtu hráčů do proměnné, to nám zajistí následující příkaz.

```
players=get(numberOfPlayers,'string');
```

Do proměnné `players` tedy uložíme hodnotu z textového pole `numberOfPlayers`. Je třeba počítat s tím, že tímto způsobem získáme hodnotu z pole ale v datovém typu string. V našem případě ale tato hodnota udává počet hráčů, tudíž se jedná o číslo, a musíme mu změnit datový typ na double. Toho docílíme tímto příkazem.

```
players=str2double(players);
```

Dalším krokem funkce bude vytvoření tabulky pro zadání váhy jednotlivých hráčů a pro přehlednější zobrazování výsledku přidáme možnost každého hráče pojmenovat.

```
columnname = {'Váha','Hráč'};
columnformat = {'numeric','char'};
data=cell(players,2);
for i=1:players
    data{i, 2}='';
end
playersTable = uitable('data',data,...
            'ColumnName', columnname,...
            'ColumnFormat', columnformat,...
            'ColumnEditable', [true true],...
            'RowName','numbered',...
            'position',[0,0,220,300]);
```

Uložíme tedy názvy sloupců do proměnné `columnname` a jejích datový typ do proměnné `columnformat`. Dále bylo potřeba si předpřipravit proměnnou, do které se budou zadané hodnoty ukládat. K tomu jsme využili proměnnou `data` typu cell o velikosti 2 sloupce a počet řádku, který odpovídá počtu hráčů. Do prvního sloupce této proměnné budeme ukládat číselné hodnoty váhy jednotlivých hráčů, do druhého sloupce budeme ukládat jména popřípadě názvy jednotlivých hráčů. Vzhledem k tomu, že tyto názvy budou v datovém typu string, je potřeba předdefinovat hodnoty pro druhý sloupec,

o to se postará uvedený cyklus for. V momentě, kdy jsme si připravili všechny proměnné, můžeme vykreslit samotnou tabulku příkazem `uitable`.

Dále funkce `FsetPlayers()` zajišťuje vykreslení polí pro zadání minimální hodnoty potřebné pro výhru koalice a možnost nastavení koeficientů u výplatních funkcí. Tyto prvky vykreslíme pomocí následujících příkazů.

```
    vsPanel = uipanel('Title','Koeficient pro v(s)',...
        'Units','pixels',...
        'Pos',[1,300,220,55]);
    vsEdit = uicontrol('style','edit',...
        'parent',vsPanel,...
        'Units','pixels',...
        'Pos',[60,10,100,25],...
        'background','white');
    vsiPanel = uipanel('Title','Koeficient pro v`(s)',...
        'Units','pixels',...
        'Pos',[580,300,220,55]);
    vsiEdit = uicontrol('style','edit',...
        'parent',vsiPanel,...
        'Units','pixels',...
        'Pos',[60,10,100,25],...
        'background','white');
```

Význam jednotlivých příkazů a parametrů je vysvětlen v předchozí podkapitole. Posledním úkolem funkce bude vykreslení tabulky pro zadání přípustných koalic a tlačítka pro samotný výpočet. Abychom mohli zobrazit tuto tabulku, musíme si napřed zavést další funkci a výpočet všech možných koalic, tuto funkci nazveme `FCoalitions()` a výpočet těchto koalic provede následovně.

```
function AllCoallitions=FCoalitions(players)
    coallition=[];
    lastrow=1;
    AllCoallitions=[];
    AllCoallitions=zeros(1,players);
    for i=1:players
        c=combnk(1:players,i);
        rows=size(c,1);
        for crow=1:rows
           for cl=1:i
                coallition(1,cl)=c(crow,cl);
           end;
                AllCoallitions(lastrow,:)=0;
                AllCoallitions(lastrow,1:i)=coallition(1,1:i);
            lastrow=lastrow+1;
        end
    end
end
```

Vstupním parametrem funkce je počet hráčů, a následně pomoci funkce `combnk()` spočítáme všechny možné koalice a uložíme je do proměnné `AllCoallitions`.

Nyní se vrátíme zpátky do funkce `FsetPlayers`, zavoláme funkci `FCoalitions()` s parametrem počet hráčů, který se aktuálně nachází v proměnné `players` a její výstup uložíme do proměnné `coallitionsData`.

```
coallitionsData=FCoalitions(players);
```

V tabulce přípustných koalic budeme chtít tyto koalice označovat zaškrtávacím polem „checkbox“. Protože chceme mít v jedné tabulce jak jednotlivé přípustné koalice vyjádřené číselně, tak zmiňovaný checkbox, který pracuje s datovým typem logical, musíme naše koalice v proměnné `coallitionsData` převést do typu cell array a následně přidat sloupec s datovým typem logical. Toto všechno uložíme do proměnné `restrictionData`.

```
restrictionData={};
    for i=1:size(coallitionsData,1)
        for n=1:size(coallitionsData,2)
            restrictionData(i,n)={coallitionsData(i,n)};
        end
        restrictionData(i,players+1)={true};
    end
    columnFormat=cell(1,players+1);
    columnEditable=zeros(1,players+1);
    for i=1:players
       columnFormat{i}='numeric';
       columnEditable(i)=0;
    end
    columnEditable(players+1)=1;
    columnEditable=logical(columnEditable);
    columnFormat{players+1}='logical';
```

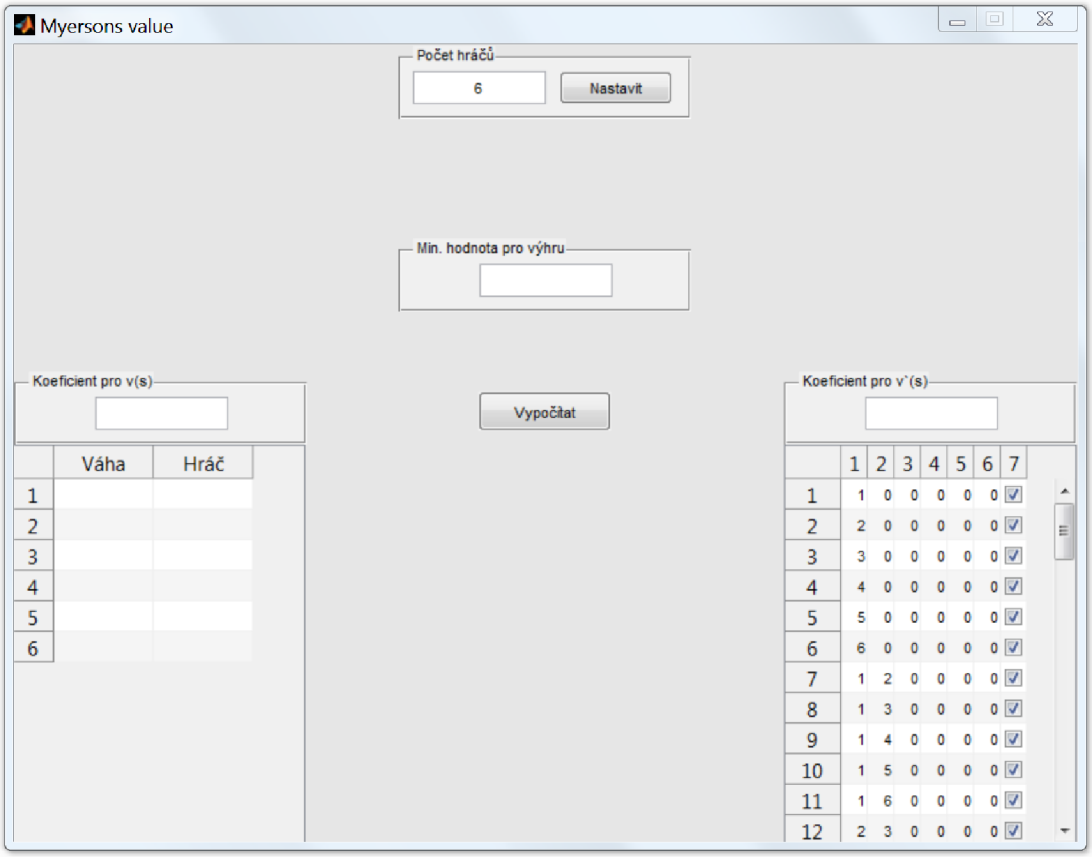
Nyní již máme veškeré potřebné údaje pro vykreslení tabulky přípustných koalic pomocí následujícího příkazu.

```
playersRestrictionTable = uitable('data',restrictionData,...
        'RowName','numbered',...
        'position',[(800-
((20*players)+100)),0,(20*players)+100,300],...
        'columnFormat',columnFormat,...
        'columnEditable',columnEditable,...
        'parent',mainwindow,...
        'columnWidth',{20});
```

Nakonec přidáme tlačítko pro uložení dat a provedení výpočtu.

```
saveData = uicontrol('parent',mainwindow,...
                        'style','pushbutton',...
                        'position',[350,310,100,30],...
                        'string','Vypočítat',...
                        'callback',@FOnclick);
```

Stisknutí tohoto tlačítka zavolá funkci `FOnclick`, která bude popsána v další podkapitole. V aktuálním stavu program vypadá tak, jak ho vidíme na následujícím obrázku.



Obrázek 3: Okno pro zadání vstupních informací (Zdroj: vlastní)

3.2 Průběh výpočtu

O samotný výpočet se stará funkce `FOnClick()`, zmíněna již v předchozí kapitole. Ale abychom mohli tuto funkci popsat a dostatečně vysvětlit, musíme nejdříve nadefinovat funkce, se kterými funkce `FOnClick()` pracuje. V našem případě se jedná o funkce:

`GetCoallitionWeight()`

`getWightedCoallitions()`

`FCountCoallitionPlayers()`

`FGetPlayerNCoallitionWeight()`

`FgetMyersonValue()`

Začneme popisem funkce `GetCoallitionWeight()`, jejímiž vstupními hodnotami jsou: vektor koalic, vektor s váhami hráčů a koeficient pro výpočet výplatní funkce. Jejím úkolem je vypočítat sílu zadané koalice. Zdrojový kód funkce vypadá následovně.

```
function weight=FGetCoalitionWeight(coallition,playerWeight,koef)
    weight=0;
    playerWeight=cell2mat(playerWeight);
    for i=1:size(coallition,2)
        if(coallition(i)==0)
            break
        end
        weight=weight+playerWeight(coallition(i))*koef;
    end
    weight=[coallition weight];
end
```

Výstupem funkce je proměnná `weight`, obsahující danou koalici a její sílu.

Funkce `getWightedCoallitions()` vypočítá sílu všech přípustných koalic i koalic vzniklých sjednocením podle definice v kapitole 1.5. a v případě že síla koalice je větší než minimální hodnota, uloží tuto koalici do proměnné `coallitions`. Vstupními parametry této funkce jsou: počet hráčů, data o hráčích, minimální hodnota pro výhru koalice, zakázané koalice, všechny koalice a koeficienty pro výplatní funkce. Nejdříve opět uvedu kód funkce a následně ho vysvětlím.

```
function
coallitions=getWeightedCoallitions(players,playersData,minValue,restri
ctedCoallitions,AllCoallitions,koefVS,koefVSi)
    playerWeight=playersData(:,1);
    lastrow=1;
    restrictedCoallitionsPlayers = zeros(1,players);
    for i=1:size(restrictedCoallitions,1)
        if(cell2mat(restrictedCoallitions(i,players+1))==0)
restrictedCoallitionsPlayers(lastrow,1:players)=cell2mat(restrictedCoa
llitions(i,1:players));
            lastrow=lastrow+1;
        end
    end
    coallitions=zeros(1,players+1);
    lastrow=1;
    acSize=size(AllCoallitions,1);
    for rc=1:size(restrictedCoallitionsPlayers,1)
        for ac=1:acSize
            if(restrictedCoallitionsPlayers(rc,1:players)==AllCoallitio
            ns(ac,1:players))
                AllCoallitions(ac,:)=[];
                break;
            end
        end
    end
    temp=[];
    sizeAC=size(AllCoallitions,1);
    for i=1:sizeAC
        for c=1:sizeAC
           if(c==i | i>c |
intersect(AllCoallitions(i,1:players),AllCoallitions(c,1:players))==0)
               continue
           end
temp=union(AllCoallitions(i,1:players),AllCoallitions(c,1:players));
           temp=temp(temp~=0);
           bool=false;
           if(size(temp,2)<players)
              temp(1,size(temp,2)+1:players)=0;
           end
           for k=1:size(AllCoallitions,1);
                 if(temp(1,1:players)==AllCoallitions(k,1:players))
                   bool=true;
                   break;
               end
           end
           if(bool==false)
                 AllCoallitions(size(AllCoallitions,1)+1,1:players)=te
                 mp(1,1:players);
           end;
        end
    end
    newCoallitions=size(AllCoallitions,1)-sizeAC;
    for i=1:(size(AllCoallitions,1)-newCoallitions)
            cWeight=FGetCoalitionWeight(AllCoallitions(i,:),playerWeigh
            t,koefVS);
            if(cWeight(1,players+1)>minValue)
                coallitions(lastrow,:)=0;
coallitions(lastrow,1:players)=AllCoallitions(i,1:players);
coallitions(lastrow,players+1)=cWeight(1,players+1);
```

```
                lastrow=lastrow+1;
        end
    end
    for i=(size(AllCoallitions,1)-
newCoallitions+1):(size(AllCoallitions,1))
cWeight=FGetCoalitionWeight(AllCoallitions(i,:),playerWeight,koefVSi);
        if(cWeight(1,players+1)>minValue)
                coallitions(lastrow,:)=0;
coallitions(lastrow,1:players)=AllCoallitions(i,1:players);
coallitions(lastrow,players+1)=cWeight(1,players+1);
                lastrow=lastrow+1;
        end
    end
end
```

Na začátku si tato funkce do proměnné `playerWeight` uloží hodnoty váhy jednotlivých hráčů. Následně ze vstupní proměnné `restrictedCoallitions` která je typu cell array, vytáhne zakázané koalice a uloží je jako vektor do proměnné `restrictedCoallitionsPlayers`. V dalším kroku porovná všechny koalice v proměnné `AllCoallitions` se zakázanými koalicemi v proměnné `restrictedCoallitionsPlayers` a tyto zakázané koalice z proměnné `AllCoallitions` vymaže. V tuto chvíli se v proměnné `AllCoallitions` nachází pouze přípustné koalice. Nyní dopočítáme nové koalice, které vzniknou sjednocením přípustných koalic podle definice z kapitoly 1.6. a tyto koalice uložíme do stejné proměnné, pod přípustné koalice. Následně do proměnné `newCoallitions` uložíme počet takto nově vzniklých koalic.

V momentě, kdy známe všechny přístupné koalice i koalice vzniklé sjednocením a známe jejích počet, můžeme pro každý řádek matice `AllCoallitions` zavolat funkci `FGetCoalitionWeight()` s parametry: řádek matice AllCoallitions, váhový vektor, koeficient pro výpočet výplatní funkce. Akorát je potřeba rozlišovat přípustné koalice a koalice vzniklé sjednocením – pro tyto dva typy koalic je potřeba zvlášť volat funkci `FGetCoalitionWeight()` s příslušnými koeficienty pro výpočet výplatní funkce nastavenými uživatelem. V případě, že síla koalice je vyšší než předem nastavená minimální hodnota, uložíme tuto koalici i s její sílou do proměnné `Coallitions`, což je zároveň výstupní hodnota této funkce.

Dále si vysvětlíme funkci `FCountCoallitionPlayers()`. Tato funkce je poměrně jednoduchá a jejím úkolem je spočítat, kolik hráčů obsahuje koalice, která je popsána vektorem obsahující koalici a její sílu – tento vektor jsme si nadefinovali v předchozí funkci a je uložen v proměnné `Coallitions`. Vstupním parametrem této funkce je tedy vektor koalice s její váhou a počet hráčů. Funkci nadefinujeme následovně.

```
function count=FCountCoallitionPlayers(coallition,players)
    members=coallition(1,1:players);
    members( :, all(~members,1) ) = [];
    count=size(members,2);
end
```

Do proměnné `members` uložíme z proměnné `coallitions` počet hodnot odpovídající počtu hráčů. Následně z proměnné `members` odstraníme sloupce obsahující nulovou hodnotu a do proměnné `count` uložíme velikost tohoto vektoru, čímž dostaneme právě počet hráčů v koalici. Další funkci, kterou si nadefinujeme a následně vysvětlíme, je funkce `FGetPlayerNCoallitionWeight()`. Tato funkce má za úkol na základě vstupních parametrů: číslo hráče (i), počet hráčů, koalice obsahující hráče i a koalice neobsahující hráče i, vypočítat sílu koalice neobsahující hráče i. Výstupem této funkce je hodnota v(s-{i}), kterou následně dosadíme do vzorce z kapitoly 1.5. Funkci opět nejprve uvedu a následně popíšu.

```
Function NCoallitionWeight=FGetPlayerNCoallitionWeight(player,
players,coallition,playerNCoallitions)
        find=false;
        coallition=coallition(1,1:players);
        coallition=coallition(coallition~=player);
        if(size(coallition,2)<players)
            coallition(1,players)=0;
        end
        for i=1:size(playerNCoallitions,1)
            if(coallition==playerNCoallitions(i,1:players))
                vsi=playerNCoallitions(i,players+1);
                find=true;
                break;
            else
                find=false;
            end
        end
        if(find==false)
            vsi=0;
        end
        NCoallitionWeight=vsi;
End
```

Na začátku je nutné nadefinovat kontrolní proměnnou, v našem případě je to proměnná `find` typu logical a nastavíme ji hodnotu false. Následně si ze vstupního vektoru koalice uložíme do proměnné `coallition` počet hodnot odpovídající počtu hráčů a vymažeme z něj sloupec obsahující hráče, který je uvedený ve vstupním parametru funkce. Abychom mohli následně vektory porovnávat, musíme vektor `coallition` zpátky doplnit na stejný počet sloupců jako má proměnná `playerNCoallitions`. Následně hledáme řádek v proměnné `playerNCoallition`, který odpovídá hodnotě v `coallition`. V případě že takový řádek respektive koalici najde, uloží její sílu do proměnné `vsi`, v opačném případě do této proměnné uloží nulu. Následně hodnotu `vsi` přiřadíme do výstupní proměnné funkce.

Poslední funkci, kterou si budeme muset nadefinovat předtím, než se vrátíme k původní `FOnClick()`, je funkce `FgetMyersonValue()`. Tato funkce na základě vstupních parametrů: číslo hráče, počet hráčů a vítězné koalice vypočítá Myersnovu hodnotu daného hráče.

```
function myers=FgetMyersonValue(player,players,coallitions)
    n=players;
    rows=size(coallitions,1);
    playerCoallitions=[];
    playerNCoallitions=[];
    playerCheck=true;

    for row=1:rows
        playercheck=true;
        for column=1:players
            if(coallitions(row,column)==player)
                playerCoallitions=[  coallitions(row,:)
                                     playerCoallitions];
                playerCheck=true;
                break;
            else
                playerCheck=false;
            end
        end
        if (playerCheck==false)
           playerNCoallitions=[  coallitions(row,:)
                                     playerNCoallitions];
        end
    end
    myers=0;
    for i=1:size(playerCoallitions,1)
```

```
        vs=playerCoallitions(i,players+1);
vsi=FGetPlayerNCoallitionWeight(player,players,playerCoallitions
(i,:),playerNCoallitions);

s=FCountCoallitionPlayers(playerCoallitions(i,:),players);
        myers=myers+((factorial(s-1)*factorial(n-
s))/factorial(n))*(vs-vsi);
    end
end
```

Nejprve si nadefinujeme proměnné, se kterými budeme ve funkci pracovat. Následně koalice z proměnné `coallitions` rozdělíme na koalice, ve kterých daný hráč figuruje a koalice, ve kterých daný hráč není. Koalice, obsahující hráče z proměnné `player` uložíme do proměnné `playerCoallitions`. A koalice, ve kterých se nevyskytuje do proměnné `playerNCoallitions`. Následně pro každý řádek v `playerCoallitions` vypočítáme hodnoty potřebné pro výpočet Myersnovy hodnoty (kapitola 1.5.). Hodnotu neznáme N udává proměnná `players`, hodnotu S získáme zavoláním funkce `FCountCoallitionPlayers()`, kterou jsme si popsali v předchozí podkapitole. Hodnotu neznáme v(s) získáme zavoláním funkce `playerCoallitions()`, kterou jsme si taktéž dříve popsali. Poslední hodnotu, kterou musíme získat, reprezentuje neznáma v(s-{i}) a tu dostaneme zavoláním funkce `FGetPlazerNCoallitionWeight()`. Nyní už pouze tyto hodnoty dosadíme do vzorce pro výpočet Myersnovy hodnoty.

Nyní již máme definované veškeré funkce, které využívá dříve zmiňovaná funkce `FOnClick()`. Připomeňme tedy, že tato funkce se spouští kliknutím na tlačítko „Vypočítat“, které jsme si vytvořili na konci podkapitoly 3.1.2.

Po spuštění tato funkce uloží veškerá data zadaná uživatelem, na základě těchto dat provede výpočet Myersnovy hodnoty každého hráče, tyto výsledky vypíše do tabulky a nakonec vykreslí graf, kde budeme moct porovnat podíl jednotlivých hráčů i jejích Myersnovu hodnotu. Jedná se opět o funkci spouštěnou callbackem, proto její vstupní parametry musí být (~,~).

```
function FOnclick(~,~)
        playersData=get(playersTable,'data');
      restrictedCoallitions=get(playersRestrictionTable,'data');
        minValue=get(minValueEdit,'string');
        minValue=str2double(minValue);
        koefVS=get(vsEdit,'string');
        koefVS=str2double(koefVS);
        koefVSi=get(vsiEdit,'string');
        koefVSi=str2double(koefVSi);
coallitions=getWeightedCoallitions(players,playersData,minValue,
restrictedCoallitions,coallitionsData,koefVS,koefVSi);
        names=cell(players,1);
        values=cell(players,1);
        for i=1:players
            names(i,1)=playersData(i,2);
        end
        for i=1:players
            myers=FgetMyersonValue(i,players,coallitions);
            values(i,1)={myers};
        end
        result=[names values];
        resultTable = uitable('data',result,...
            'RowName','numbered',...
            'position',[300,0,200,300],...
            'parent',mainwindow,...
            'columnName',{'hrac','myerons value'},...
            'columnWidth',{60,100});
        fg=figure('Position',[0.5*(ScreenSize(3)-600),...
                              0.5*(ScreenSize(4)-500),...
                              600,500]);
        figure(fg);
        barValues=[];
        for i=1:players
            barValues=[barValues; cell2mat(values(i))
playersData(i,1);];
        end
        bar(cell2mat(barValues))
        title('Myersnová hodnota a podíl jednotlivých hráčů');
        xlabel('Hráči');
        ylabel('Hodnota [myers podíl]');
        legend('myers','podíl')
        set(gca, 'XTick', 1:players, 'XTickLabel', names);
        maximum=get(gca,'ylim');
        ylim([0,maximum(2)*1.2]);

end
```

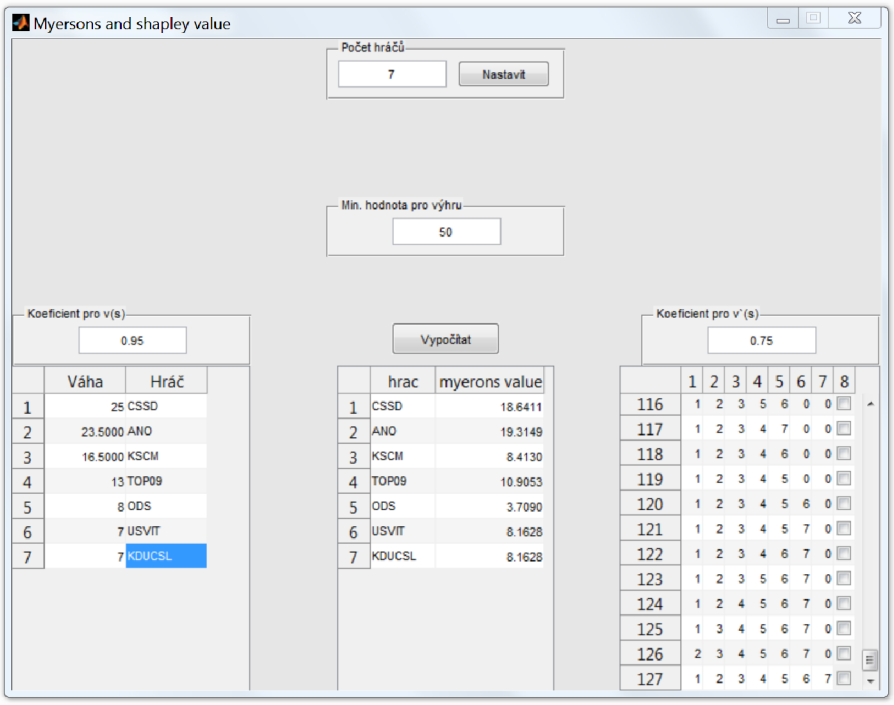
Po stisknutí tlačítka pro výpočet, naše funkce uloží veškeré údaje do příslušných proměnných a převede do požadovaných datových typu. Následně zavolá funkci `getWeightedCoallitions()` se vstupními parametry, které jsme si definovali při vytváření této funkce. Výstup z volané funkce uloží do proměnné `coallitions`. Připomeňme, že tímto získáme matici vítězných koalic a jejích sílu. V dalším kroku si nadefinujeme proměnné, které využijeme pro zobrazení tabulky výsledků. Jedná se o proměnnou `names` datového typu cell, do které uložíme jména, respektive názvy hráčů a proměnnou `values` taktéž datového typu cell, do které uložíme Myersnovu hodnotu jednotlivých hráčů. Tu získáme tak, že zavoláme funkci `FgetMyersonValue()`, kterou jsme si definovali v předchozí kapitole.

Ve chvíli, kdy máme tyto dvě proměnné naplněné, je společně uložíme do proměnné `result`. Následně vykreslíme tabulku výsledků pomocí příkazu `uitable` a jako data zvolíme proměnnou `result`, kterou jsme si v předchozím kroku vytvořili. Zbývá pouze vykreslit graf podílů jednotlivých hráčů a jejích Myersnových hodnot. Pro vykreslení grafu si musíme uložit dvojice zmiňovaných hodnot každého hráče do proměnné `barValues`. Následně vykreslíme graf příkazem `bar` a jako hodnoty nastavíme proměnnou `barValues`. Ta je ale momentálně v datovém typu cell, proto jí musíme pomoci funkce `cell2mat()` převést na číselné hodnoty. V posledním kroku u grafu nastavíme popisky osy X a Y a legendu.

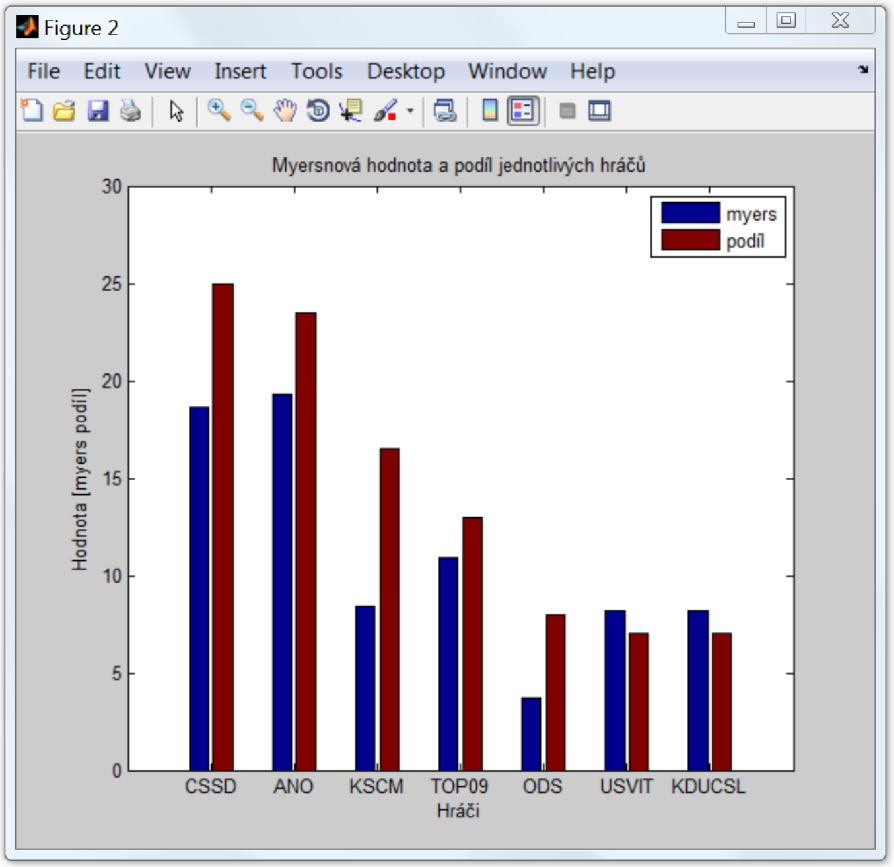
Tímto jsme dokončili celý program pro výpočet Myersnovy hodnoty. Zdrojový kód v celku je uvedený v příloze 1.

3.3 Výstupy programu

Příklad z analytické část vypočítaný pomocí tohoto programu můžeme vidět na obrázku 4 a 5 na následující straně.



Obrázek 4: Zobrazení číselného výpočtu (Zdroj: vlastní)



Obrázek 5: Grafické znázornění výpočtu (Zdroj: vlastní)

ZÁVĚR

S aplikací teorie her se v praxi setkáváme čím dál častěji. V tomto oboru velmi známy profesor Bruce Bueno de Mesquita z New York University již více než 20 let provádí úspěšné politické předpovědi využíváním teorie her. Stejné principy jako na politické scény lze přenést i k rozhodování na úrovní firem či jednotlivců. [10]

V teoretické částí bakalářské práce byly definované základní pojmy a východiska teorie her. Dále byly popsané jednotlivé druhy a typy koaličních her. V závěru této části byla definována Shapleyho a z ní odvozená Myersnova hodnota. Tyto hodnoty byly následně využívané v dalších částech práce.

V analytické části byl na vzorovém příkladu popsán postup při výpočtu Shapleyho a Myersnovy hodnoty. V další podkapitole byly všechny získané poznatky aplikované pro výpočet vyjednávací síly jednotlivých politických stran ČR. V této částí bylo také zjištěno, že výpočet těchto hodnot je poněkud náročný a toto zjištění vedlo k naprogramování software pro výpočet těchto hodnot.

Návrhová část práce se tedy zabývala návrhem a programováním již zmíněné aplikace. Popsány byly jednotlivé kroky při vytváření takové aplikace s využitím matematického software Matlab, včetně náhledu aplikace v jednotlivých krocích. Tato aplikace byla následně použitá pro výpočet Myersnovy hodnoty politických stran pojednávaných v analytické části.

Vzhledem k tomu, že momentálně neexistuje žádná dostupná literatura v českém jazyce zabývající se koaličními hrami v teorií her, přínosem bakalářské práce kromě samotné aplikace na výpočet Myersnovy hodnoty, je taktéž shrnutí teorie z více zahraničních zdrojů a jejich překlad.

SEZNAM POUŽITÉ LITERATURY

[1] DLOUHÝ, Martin a Petr FIALA. *Úvod do teorie her*. Praha: Oeconomica, 2007. ISBN 978-80-245-1273-0.

[2] OWEN, Guillermo. *Game theory*. 4th ed. Bingley, UK: Emerald, 2013. ISBN 978-178-1905-074.

[3] BEZALEL PELEG, Peter Sudhölter. *Introduction to the theory of cooperative games*. 2nd ed. Berlin: Springer, 2007. ISBN 978-354-0729-457.

[4] PETERS, Hans J. M. *Game theory: a multi-leveled approach*. 4th ed. Berlin: Springer, 2008. ISBN 978-3-540-69290-4.

[5] SCHOFIELD, Norman. *Mathematical methods in economics and social choice*. Berlin: Springer, 2003. Theory and decision library, v. 44. ISBN 978-354-0000-860.

[6] LOMUSCIO, Alessio, Paul SCERRI, Ana BAZZAN, a Michael HUHNS *Proceedings of the 2014 International Conference on Autonomous Agents & Multiagent Systems*. Paříž*:* AMMAS 2014. ISBN 978-1-4503-2738-1.

[7] GILLES, Robert P. *The cooperative game theory of networks and hierarchies*. Heidelberg: Springer, c2010. Theory and decision library, v. 44. ISBN 978-364-2052-828.

[8] ČESKO. Zákon č. 90 z roku 1995. In: *Sbírka zákonů České republiky*. 1995.

Dostupný také z: http://www.psp.cz/docs/laws/1995/90.html. ISSN 1211-1244

[9] Výsledky voleb v České republice. *IDNES.cz* [online]. 2013 [cit. 2016-05-30]. Dostupné z: http://volby.idnes.cz/poslanecka-snemovna-2013.aspx

[10] JANOUCH, Viktor. Manažerské rozhodování s podporou teorie her. *IT Systems*. 2012. 3, s. 34-35. ISSN 1802-615X

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

PŘÍLOHA 1

```
function myers
    close all
    clear all
    clc
players=0;
global playersData playersTable minValueEdit data minValue
restrictedCoallitions coallitionsData playersRestrictionTable
koefVS koefVSi vsEdit vsiEdit
ScreenSize=get(0,'ScreenSize');
    mainwindow=figure('Name','Myersons value',...
                      'NumberTitle','Off',...
                      'Menubar','none',...
                      'Resize','off',...
                      'Units','pixels',...
                      'Position',[0.5*(ScreenSize(3)-800),...
                                  0.5*(ScreenSize(4)-600),...
                                  800,600],...
                      'color',[0.9 0.9 0.9]);

pocetHracuPanel = uipanel('Title','Počet hráčů',...
                          'Units','pixels',...
                          'Pos',[290,545,220,55]);
uicontrol('String','Nastavit',...
    'Parent',pocetHracuPanel,...
    'position',[120,10,85,25],...
    'callback',@FsetPlayers);

numberOfPlayers = uicontrol('Parent',pocetHracuPanel,...
    'Style','edit',...
    'visible','on',...
    'position',[10,10,100,25],...
    'background','white');

function FsetPlayers(~,~)
    players=get(numberOfPlayers,'string');
    players=str2double(players);
    % Nazev a format sloupcu
    columnname = {'Váha','Hráč'};
    columnformat = {'numeric','char'};
    % data, aby bylo mozne ukládat string
    data=cell(players,2);
    for i=1:players
        data{i, 2}='';
    end
    % Tabluka pro zadani hodnty hracu %
    playersTable = uitable('data',data,...
                'ColumnName', columnname,...
                'ColumnFormat', columnformat,...
                'ColumnEditable', [true true],...
                'RowName','numbered',...
                'position',[0,0,220,300]);
```

```
    % Okno pro minimalni hodnotu %
    minValuePanel = uipanel('Title','Min. hodnota pro výhru',...
        'Units','pixels',...
        'Pos',[290,400,220,55]);
    minValueEdit = uicontrol('style','edit',...
        'parent',minValuePanel, 'Units',...
        'pixels', 'Pos',[60,10,100,25],...
        'background','white');
    % Okno pro koeficient v(s) koalic
    vsPanel = uipanel('Title','Koeficient pro v(s)',...
        'Units','pixels',...
        'Pos',[1,300,220,55]);
    vsEdit = uicontrol('style','edit',...
        'parent',vsPanel,...
        'Units','pixels',...
        'Pos',[60,10,100,25],...
        'background','white');
    % Okno pro koeficient v`(s) koalic
    vsiPanel = uipanel('Title','Koeficient pro v`(s)',...
        'Units','pixels',...
        'Pos',[580,300,220,55]);
    vsiEdit = uicontrol('style','edit',...
        'parent',vsiPanel,...
        'Units','pixels',...
        'Pos',[60,10,100,25],...
        'background','white');
    % Zjisti vsechny mozne koalice %
    coallitionsData=FCoalitions(players);
    %coallitionsData=AllCoallitions;
    restrictionData={};
    for i=1:size(coallitionsData,1)
        for n=1:size(coallitionsData,2)
            restrictionData(i,n)={coallitionsData(i,n)};
        end
        restrictionData(i,players+1)={true};
    end
    columnFormat=cell(1,players+1);
    columnEditable=zeros(1,players+1);
    for i=1:players
       columnFormat{i}='numeric';
       columnEditable(i)=0;
    end
    columnEditable(players+1)=1;
    columnEditable=logical(columnEditable);
    columnFormat{players+1}='logical';

    playersRestrictionTable = uitable('data',restrictionData,...
        'RowName','numbered',...
        'position',[(800-
((20*players)+100)),0,(20*players)+100,300],...
        'columnFormat',columnFormat,...
        'columnEditable',columnEditable,...
        'parent',mainwindow,...
```

```
        'columnWidth',{20});
    % Tlacitko - Provede vypocet %
    saveData = uicontrol('parent',mainwindow,...
                         'style','pushbutton',...
                         'position',[350,310,100,30],...
                         'string','Vypočítat',...
                         'callback',@FOnclick);
end

function FOnclick(~,~)
        tic
        playersData=get(playersTable,'data');

restrictedCoallitions=get(playersRestrictionTable,'data');
        minValue=get(minValueEdit,'string');
        minValue=str2double(minValue);
        koefVS=get(vsEdit,'string');
        koefVS=str2double(koefVS);
        koefVSi=get(vsiEdit,'string');
        koefVSi=str2double(koefVSi);
        % Provede vypocet koalic %
coallitions=getWeightedCoallitions(players,playersData,minValue,
restrictedCoallitions,coallitionsData,koefVS,koefVSi);
        %vypocita myersovou hodnotu pro kazdeho hrace %
        names=cell(players,1);
        values=cell(players,1);
        for i=1:players
            names(i,1)=playersData(i,2);
        end
        for i=1:players
            myers=FgetMyersonValue(i,players,coallitions);
            disp('#### MYERS')
            values(i,1)={myers};
            disp('###')
        end
        result=[names values];
        % Tabulka vysledku %
        resultTable = uitable('data',result,...
            'RowName','numbered',...
            'position',[300,0,200,300],...
            'parent',mainwindow,...
            'columnName',{'hrac','myerons value'},...
            'columnWidth',{60,100});
        % okno pro zobrazeni grafu %
        fg=figure('Position',[0.5*(ScreenSize(3)-600),...
                              0.5*(ScreenSize(4)-500),...
                              600,500]);
        % vykresleni grafu %
        figure(fg);
        barValues=[];
        for i=1:players
            barValues=[barValues; cell2mat(values(i))
playersData(i,1);];
```

```
        end
        bar(cell2mat(barValues))
        title('Myersnová hodnota a podíl jednotlivých hráčů');
        xlabel('Hráči');
        ylabel('Hodnota [myers podíl]');
        legend('myers','podíl')
        set(gca, 'XTick', 1:players, 'XTickLabel', names);
        maximum=get(gca,'ylim');
        ylim([0,maximum(2)*1.2]);
        toc
end
end

function AllCoallitions=FCoalitions(players)
    coallition=[];
    lastrow=1;
    AllCoallitions=[];
    AllCoallitions=zeros(1,players);
    for i=1:players
        c=combnk(1:players,i);
        rows=size(c,1);
        for crow=1:rows
           for cl=1:i
                coallition(1,cl)=c(crow,cl);
           end;
                AllCoallitions(lastrow,:)=0;
                AllCoallitions(lastrow,1:i)=coallition(1,1:i);
            lastrow=lastrow+1;
        end
    end
end

function
coallitions=getWeightedCoallitions(players,playersData,minValue,
restrictedCoallitions,AllCoallitions,koefVS,koefVSi)
    playerWeight=playersData(:,1);
    lastrow=1;
    restrictedCoallitionsPlayers = zeros(1,players);
    for i=1:size(restrictedCoallitions,1)
        if(cell2mat(restrictedCoallitions(i,players+1))==0)

restrictedCoallitionsPlayers(lastrow,1:players)=cell2mat(restric
tedCoallitions(i,1:players));
            lastrow=lastrow+1;
        end
    end
    coallitions=zeros(1,players+1);
    lastrow=1;
    acSize=size(AllCoallitions,1);
    for rc=1:size(restrictedCoallitionsPlayers,1)
        for ac=1:acSize
```

```
if(restrictedCoallitionsPlayers(rc,1:players)==AllCoallitions(ac
,1:players))
                AllCoallitions(ac,:)=[];
                break;
            end
        end
    end
    % zjisti nove mozne koalice %%%
    temp=[];
    sizeAC=size(AllCoallitions,1);
    for i=1:sizeAC
        for c=1:sizeAC
            if(c==i | i>c |
intersect(AllCoallitions(i,1:players),AllCoallitions(c,1:players
))==0)
                continue
            end


temp=union(AllCoallitions(i,1:players),AllCoallitions(c,1:player
s));
            temp=temp(temp~=0);
            bool=false;
            if(size(temp,2)<players)
               temp(1,size(temp,2)+1:players)=0;
            end
            for k=1:size(AllCoallitions,1);

if(temp(1,1:players)==AllCoallitions(k,1:players))
                    bool=true;
                    break;
                end
            end
            if(bool==false)

AllCoallitions(size(AllCoallitions,1)+1,1:players)=temp(1,1:play
ers);
            end;
        end
    end
    AllCoallitions
    newCoallitions=size(AllCoallitions,1)-sizeAC;
    %---- urci silu puvodnich koalici v AllCoalitions %
    for i=1:(size(AllCoallitions,1)-newCoallitions)

cWeight=FGetCoalitionWeight(AllCoallitions(i,:),playerWeight,koe
fVS);
        if(cWeight(1,players+1)>minValue)
                coallitions(lastrow,:)=0;

coallitions(lastrow,1:players)=AllCoallitions(i,1:players);
```

```
coallitions(lastrow,players+1)=cWeight(1,players+1);
                lastrow=lastrow+1;
        end
    end
    %--- urci silu nove vzniklych koalici---%
    for i=(size(AllCoallitions,1)-
newCoallitions+1):(size(AllCoallitions,1))

cWeight=FGetCoalitionWeight(AllCoallitions(i,:),playerWeight,koe
fVSi);
        if(cWeight(1,players+1)>minValue)
                coallitions(lastrow,:)=0;

coallitions(lastrow,1:players)=AllCoallitions(i,1:players);

coallitions(lastrow,players+1)=cWeight(1,players+1);
                lastrow=lastrow+1;
        end
    end
    coallitions
end

function
weight=FGetCoalitionWeight(coallition,playerWeight,koef)
    weight=0;
    playerWeight=cell2mat(playerWeight);
    for i=1:size(coallition,2)
        if(coallition(i)==0)
            break
        end
        weight=weight+playerWeight(coallition(i))*koef;
    end
    weight=[coallition weight];
end

function myers=FgetMyersonValue(player,players,coallitions)
    n=players;
    rows=size(coallitions,1);
    playerCoallitions=[];
    playerNCoallitions=[];
    playerCheck=true;

    for row=1:rows
        playercheck=true;
        for column=1:players
            if(coallitions(row,column)==player)
                playerCoallitions=[  coallitions(row,:)
                                   playerCoallitions];
                playerCheck=true;
                break;
            else
                playerCheck=false;
```

```
            end
        end
        if (playerCheck==false)
           playerNCoallitions=[  coallitions(row,:)
                                    playerNCoallitions];
        end
    end
    myers=0;
    for i=1:size(playerCoallitions,1)
        vs=playerCoallitions(i,players+1);

vsi=FGetPlayerNCoallitionWeight(player,players,playerCoallitions
(i,:),playerNCoallitions);

s=FCountCoallitionPlayers(playerCoallitions(i,:),players);
        myers=myers+((factorial(s-1)*factorial(n-
s))/factorial(n))*(vs-vsi);
    end
end

function count=FCountCoallitionPlayers(coallition,players)
    members=coallition(1,1:players);
    members( :, all(~members,1) ) = [];
    count=size(members,2);
end

function
NCoallitionWeight=FGetPlayerNCoallitionWeight(player,players,coa
llition,playerNCoallitions)
        find=false;
        coallition=coallition(1,1:players);
        coallition=coallition(coallition~=player);
        if(size(coallition,2)<players)
            coallition(1,players)=0;
        end
        for i=1:size(playerNCoallitions,1)
            if(coallition==playerNCoallitions(i,1:players))
                vsi=playerNCoallitions(i,players+1);
                find=true;
                break;
            else
                find=false;
            end
        end
        if(find==false)
            vsi=0;
        end
        NCoallitionWeight=vsi;
end
```