VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

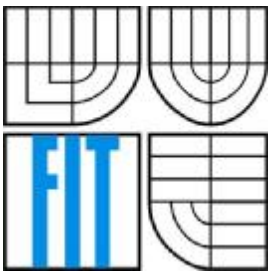
NÁSTROJ PRO ANALÝZU KOOPERATIVNÍCH HER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE Filip Kessner
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NÁSTROJ PRO ANALÝZU KOOPERATIVNÍCH HER

A TOOL FOR ANALYSIS OF COOPERATIVE GAMES

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

FILIP KESSNER

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN HRUBÝ, Ph.D.

BRNO 2013

Abstrakt

Kooperativní hry nejčastěji modelují systémy reálného světa, ve kterých jde o spolupráci nebo o společné rozhodnutí. Zkoumaným systémem může být taková drobnost, jakou je spor o košili, ale i závažný problém jako tvorba volebního systému, alokace nákladů znečištění životního prostředí nebo ochrana hospodářské soutěže. Grafický program, který je součástí této práce, je schopen výpočtu nejznámějších forem řešení a vybraných charakteristik zkoumaných kooperativních her.

Klíčová slova

kooperativní hry, jádro hry, Shapleyho hodnota, Power index, Nukleolus.

Abstract

Cooperative games are most frequently used to simulate systems of the real world in which deal with cooperation and mutual agreement. Systems under investigation can be as simple as dispute over a shirt but investigated problems can be as crucial as designing of an electoral system, pollution cost allocation or protection of market competition as well. This thesis includes a graphical application capable of computation of some well-known solution types and selection of properties in cooperative games context.

Keywords

cooperative games, the Core, Shapley value, Power index, the Nucleolus.

Citace

Kessner Filip: Nástroj pro analýzu kooperativních her. Brno, 2013, bakalářská práce, FIT VUT v Brně.

Nástroj pro analýzu kooperativních her

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Hrubého, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Filip Kessner
15.5.2013

Poděkování

Chtěl bych poděkovat Ing. Martinovi Hrubému, Ph.D za vedení a rady, které mi poskytl při tvorbě této bakalářské práce.

Obsah

Úvod

Teorie her umožňuje matematicky popisovat systémy reálného světa, ve kterých se vybrané subjekty rozhodují nad různými strategiemi, a následně je zkoumat. Právě zkoumáním her lze získat informace o chování jejich subjektů, kterými nejčastěji bývají lidé, společnosti, vlády a korporace, a eventuelně jejich chování a reakce i předvídat. I přes svůj exaktní charakter se tak stává neocenitelným nástrojem humanitních oborů od sociologie, přes politologii až po ekonomii.

Kooperativní hry jsou oblastí teorie her, která se zabývá především spoluprací a fúzemi zúčastněných stran. Stejně jako hry nekooperativní, tak i kooperativní hry modelují neexistující nebo abstrahovaný reálný systém, zkoumají jeho vlastnosti a chování jeho prvků, respektive účastníků. Modelování a zkoumání reálných systémů (zákonodárné orgány, společná zahraniční politika, vznik kartelových dohod) může výrazně pomoci v odhalování jejich slabin a sloužit tak k jejich zkvalitnění, avšak jen pokud je abstrakce modelovaného systému dostatečně kvalitní.

Nástroj vzniklý v rámci této bakalářské práce napomáhá uživateli ve zkoumání vlastností modelovaných kooperativních her výpočtem různých analytických ukazatelů a charakteristik. Nástroj je schopný výpočtu nejznámějších forem řešení - jádra hry, Shapleyho hodnoty, Power indexů a nukleolu a navíc umí ověřit některé vlastnosti kooperativních her, to vše v přívětivém grafickém rozhraní. Díky implementaci v jazyce Java je program kompaktní a navíc umožňuje běh na libovolné platformě pro PC.

Práce začíná kapitolou vymezující pojmy vztažené ke kooperativním hrám obecně a i některé pojmy z oblasti nekooperativních her, neboť obě tyto oblasti spolu úzce souvisí. V další kapitole jsou představeny formy řešení, které program implementuje. Způsob implementace popisuje kapitola třetí, která navíc popisuje program i z pohledu uživatele a jeho možností v návodu k obsluze. Závěrečná kapitola se zabývá experimentováním s nástrojem na konkrétních kooperativních hrách, včetně dvou reálných studií alokačních problémů.

1 Vymezení pojmů

Před představením samotného nástroje je nutné vymezit několik pojmů souvisejících s kooperativními hrami v rámci teorie her a vzhledem k vlastnostem programu i některé pojmy z oblasti nekooperativních her. Naopak zde nebudou vysvětleny základní pojmy z teorie her, u čtenáře se počítá se základním přehledem o dané problematice.

1.1 Kooperativní hry

„Hra je kooperativní, pokud se hráči mohou závazně dohodnout nad rozdělením zisku nebo výběru strategií i v případě, že hra dodržení těchto dohod přímo nezaručuje ani neimplikuje. [1]“. Tolik popis kooperativních her, které se dále dělí na:

- hry s přenositelným užitkem - hráči se mohou dohodnout na strategii a rozdělení zisku
- hry s nepřenositelným užitkem - hráči se mohou dohodnout pouze na strategii

Program se věnuje pouze hrám s přenositelným užitkem, proto zde hry s nepřenositelným užitkem nejsou popsány a v rámci této práce bude pojem kooperativní hry nadále označovat hry s přenositelným užitkem.

Kromě neformálního popisu se třeba znát i formální matematický popis, který naznačuje, jak lze konkrétní kooperativní hru zapsat a jak s ní počítat. Formální definice kooperativní hry je následující: „Kooperativní hra s přenositelným užitkem je dvojice (N,v), kde N je množina hráčů a v je funkce přiřazující reálné číslo $v(S)$ každé podmnožině S z N. Vždy předpokládejme, že $v(\emptyset)=0$. Množinu N nazývejme množinou hráčů a v koaliční funkcí. Pokud se ve hře zformuje koalice S, její hráči realizují zisk $v(S)$. Číslo $v(S)$ se nazývá hodnotou koalice S. [2]“. Důležité pojmy jako koalice a charakteristická funkce (jiný název pro koaliční funkci) jsou rozebrány v následujících podkapitolách.

1.1.1 Charakteristická funkce

Doménou charakteristická funkce jsou všechny možné koalice, které lze utvořit z množiny hráčů. Počet takových koalic je logicky vždy roven $2^{|N|}$, kde N je počet hráčů. Obor hodnot charakteristické funkce vyjadřuje zisk, který má daná koalice (pokud je ve hře zformována) zaručen. Tento princip minimální garantované hodnoty vytváří vazbu na nekooperativní hry, která je vysvětlena v kapitole Nekooperativní hry.

Od hodnot funkce se odvíjí i chování hráčů při formování koalic a veškeré výpočty. Zkoumáním charakteristik této funkce lze zjistit vlastnosti hry, které výrazně zjednodušují některé výpočty a signalizují typické chování těchto her. Charakteristiky použité v programu jsou popsány v následujících podkapitolách.

1.1.1.1 Nepodstatná neboli aditivní hra

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

Jak už název napovídá, nepodstatnou hru nemá smysl řešit, hráči v ní totiž vždy dosahují stejného užitku nezávisle na zvolené koalici i výběru kooperativního či nekooperativního profilu, takže prostor k vyjednávání obsahuje pouze jedno řešení. Nepodstatná hra vzniká, pokud každé dvě disjunktní koalice dostanou v součtu stejné ohodnocení jako je ohodnocení jejich společné koalice.

1.1.1.2 Konvexní hra

$$v(S) + v(T) \leq v(S \cup T) + v(S \cap T);\ S, T \subseteq N$$

Konvexní hry mají volnější definici než aditivní hry a tvoří proto nadmnožinu aditivních her. Tento vztah lze najít i v následujících charakteristikách, pro něž platí vztah:

aditivní => konvexní => super-aditivní => slabě super-aditivní => monotónní

Pokud při zkoumání hry zjistíme, že hra je aditivní, je automaticky i konvexní, super-aditivní, atd. Naopak při zjištění absence byť slabé super-aditivity nemá smysl zkoumat přítomnost vyšších charakteristik. Vlastnosti konvexních her lze využít pro zjednodušení některých výpočtů, zejména ve formě řešení nukleolem.

1.1.1.3 Super-aditivní hra

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

Zjištění přítomnosti super-aditivity ve hře je klíčové pro použití některých zjednodušených algoritmů v programu. Super-aditivní hra zkoumá na rozdíl od konvexní jen disjunktní dvojice koalic, ve výsledku musí dodržovat stejnou podmínku jako aditivní hra s rozdílem, že hodnota společné koalice může být i větší než součet hodnot disjunktních koalic.

1.1.1.4 Slabě super-aditivní hra

$$v(S) + v(\{i\}) \leq v(S \cup \{i\});\ S \subseteq N, i \in N, i \notin S$$

Tento typ hry na rozdíl od super-aditivní hry nezkoumá všechny dvojice disjunktních koalic, ale jen přínos jednotlivých hráčů do koalic, jichž nejsou členy. Na rozdíl od super-aditivní hry nemusí být podmínka splněna pro všechny dvojice koalic, stačí, když je splněna pro takové dvojice, z nichž jedna je vždy jednoprvková.

1.1.1.5 Monotónní hry a hry normalizované k nule

$$v(S) \leq v(T); S \subseteq T \subseteq N \qquad\qquad v(\{i\}) = 0; i \in N$$

V monotónních hrách má každá koalice zaručeno, že přizváním dalších členů či koalic se hodnota koalice nesníží. Ve hrách normalizovaných k nule je hodnota každé koalice s počtem členů nižším než dva nulová. Podle Pelega [1] lze jakoukoliv hru upravit na strategicky ekvivalentní hru normalizovanou k nule.

1.1.2 Koalice

Hráči se formují do koalic za účelem vyšších zisků, aby však hráči vstoupili do koalice, musí být splněno několik podmínek [3]:

- každá kooperativní hra má i nekooperativní profil; zisk hráče v kooperativním profilu musí být vyšší, nežli v profilu nekooperativním.
- musí existovat důvěra ve vymahatelnost dohodnutého chování
- koalice musí být stabilní, tzn. její podkoalice nemohou vystoupením nic získat

Hodnota koalice určuje zisk celé koalice, avšak pro hráče je primární jejich vlastní zisk, k čemuž slouží výplatní vektor - vektor zisků jednotlivých hráčů. Samotný výplatní vektor však nezaručuje splnění výše uvedených podmínek, vymezme tedy ještě pojem imputace. Imputace je výplatní vektor, který rozděluje celý zisk koalice (kolektivní racionalita) a zároveň každému hráči dává zisk, který není menší než nekooperativní řešení tohoto hráče. Formálně lze imputaci definovat následujícím výrazem:

$$(a : \forall\, a_i \geq v(\{i\}), \sum_{i \in N} a_i = v(N))$$

1.2 Vazba nekooperativních her na kooperativní

V nekooperativních hrách je možné hledat kooperativní řešení stejně jako ve hrách kooperativních, pokud jsou hráči schopni a ochotni se dohodnout na volbě strategie a zároveň existují mechanismy na vymáhání dohodnutého chování. Pokud jsou tyto podmínky splněny, zbývá ještě jeden problém, a to převedení N-rozměrného prostoru výplatních vektorů nekooperativní hry na charakteristickou funkci hry kooperativní.

Charakteristická funkce je, jak už bylo řečeno, tvořena zisky, které je hra schopna dané koalici garantovat. K vytvoření charakteristické funkce je proto nutné pro každou koalici zjistit minimální dosažitelnou výplatu v nekooperativní hře při kooperaci všech jejich členů. K tomu slouží Nashovo ekvilibrium, které vypočte všechny přijatelné nekooperativní profily v dané hře, ze kterých lze snadno vypočíst ohodnocení jednotlivých hráčů. Každá nekooperativní hra může obsahovat i více ekvilibrií, avšak vždy musí obsahovat alespoň jedno.

1.2.1 Ryzí Nashovo ekvilibrium

Nashovo ekvilibrium může nabývat dvou forem - ryzí a smíšené. Ryzí ekvilibrium představuje takový nekooperativní profil, ve kterém hráči preferují jednu svou strategii před ostatními. Výběr je stabilní, protože žádný hráč není schopen pouze změnou své strategie dosáhnout vyššího zisku [4].

Výpočet lze demonstrovat na známém příkladu nazvaném Vězňovo dilema, který se týká dvou zločinců podezřelých z loupežného přepadení. Zločinci jsou drženi na celách odděleně a zvažují, zda se přiznat nebo mlčet, aniž by věděli, jak se rozhodne ten druhý. Tabulka č. 1 zobrazuje počet let za mřížemi (záporný zisk), které jim hrozí:

Tabulka č. 1: Vězňovo dilema

	Mluvit	Mlčet
Mluvit	-6,-6	0,-10
Mlčet	-10,0	-1,-1

Zdroj: vlastní

Podtržené zisky ve hře značí ekvilibrijní zisky konkrétních hráčů, které je nenutí změnit strategii. Pokud jsou všechny zisky v některém z profilů ekvilibrijní, pak je tento profil čistým Nashovým ekvilibriem. Nejlepším řešením této hry je pro oba hráče mluvit, což jim paradoxně přinese nejhorší možný výsledek, je to však logické. Při pohledu na možnosti hráčů je v každém případě lepší mluvit nežli mlčet, neboť si hráč zkrátí pobyt za mřížemi o jeden rok nebo šest let.

Pokud by hráči byli na jedné cele a spolupracovali by, mohli by dostat oba jen jeden rok za mřížemi a vyjít z celé situace výrazně lépe, což řeší právě kooperativní hry. Jedinou otázkou v tomto případě zůstává, jak jeden hráč donutí druhého, aby držel slovo, neboť v opačném případě oběma hrozí deset let ve vězení.

1.2.2 Smíšené Nashovo ekvilibrium

„Smíšené Nashovo ekvilibrium modeluje stabilní stav hry, ve které hráči nevolí deterministicky, ale podle pravděpodobnostních pravidel. [4]“. Hráči tedy přiřazují svým strategiím pravděpodobnosti, s jakými plánují danou strategii hrát. Ryzí Nashovo ekvilibrium by v tomto kontextu přidělilo jedné strategii 100% a ostatním 0%. Ve smíšených ekvilibriích se hráči snaží správným nastavením pravděpodobností donutit své protihráče k indiferenci ve volbě jejich strategie, čehož dosáhnou, pokud se očekávané užitky protihráčů z jejich strategií budou rovnat. Tento koncept je poněkud složitější, proto jej vysvětlím na příkladě:

Tabulka č. 2: Házení mincí

	Levá	Pravá
Horní	1,-1	-1,1
Dolní	-1,1	1,-1

Zdroj: Gametheory101.com [5]

Nechť tabulka č. 2 značí nekooperativní hru hráče A a B se strategiemi Horní a Dolní, respektive Levá a Pravá. Hra zjevně neobsahuje ryzí Nashovo ekvilibrium, hráči se tedy mohou pokusit o smíšené ekvilibrium. Hráč A musí volit pravděpodobnosti svých strategií tak, aby očekávaný užitek hráče B byl roven pro všechny (obě) jeho strategie:

$$-1 * pst_{Horní} + 1 * pst_{Dolní} = 1 * pst_{Horní} - 1 * pst_{Dolní}$$

$$pst_{Dolní} + pst_{Horní} = 1$$

Vypočtením předchozích dvou rovnic o dvou neznámých dostáváme pravděpodobnosti 50%, respektive 50% pro hráče A. Očekávaný užitek hráče B je pak skutečné stejný pro obě jeho strategie a je v tomto případě roven nule. Vypočtením rovnic pro hráče B získáme v tomto případě stejné pravděpodobnosti 50%, respektive 50%.

Tyto pravděpodobnosti hráčů A a B tvoří smíšené Nashovo ekvilibrium, při kterém oba hráči získají přesně nulu.

2 Formy řešení

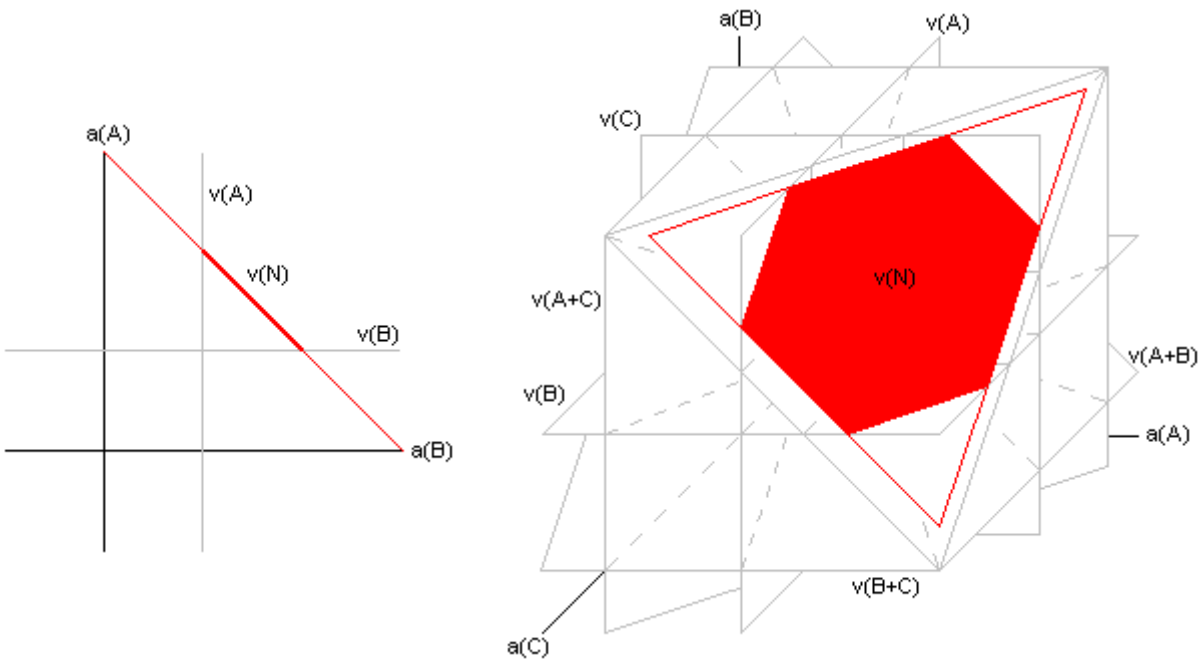
Obsahem této kapitoly je popis několika známých způsobů, jimiž lze řešit kooperativní hry, respektive způsobů spravedlivého rozdělení zisku mezi hráče pro danou hru. Pojem „spravedlivý“ je však v běžném životě záležitostí subjektivní a ani tvůrci těchto forem řešení nejsou v tomto ohledu za jedno. Ke každé formě řešení je uveden i algoritmický postup řešení.

2.1 Jádro hry

„Jádro je taková množina imputací, že každá případná koalice S obdrží alespoň hodnotu své koalice. [3]“. Jádro je tedy ještě úžeji definovanou verzí výplatních vektorů nežli imputace a obsahuje všechna možná rozdělení hodnoty velké koalice, která jsou přijatelná pro všechny koalice ve hře. Pokud je jádro hry prázdné, hráči nebudou chtít utvořit velkou koalici.

Pro výpočet jádra hry je důležité si uvědomit, že jádro je N-rozměrný podprostor prostoru, jehož souřadnice tvoří výplatní vektory a kde N je počet hráčů. Tento podprostor je omezen hodnotami koalic, které „odřezávají“ všechny nižší hodnoty a nakonec je určen řezem celkové hodnoty koalice. Obrázek č. 1 demonstruje příklad pro dva hráče (2D) a tři hráče (3D):

Obrázek č. 1: Zobrazení jádra hry



Zdroj: vlastní

Řezy *v* označují hodnoty koalic, neboli minimální přijatelné hodnoty pro dané koalice, vytvořený podprostor tvoří v řezu *v(N)* jádro hry (na obrázku červené zvýraznění).

Při algoritmickém postupu by však byl výpočet řezů v N-rozměrném spojitém prostoru příliš náročný a výsledek navíc těžko zobrazitelný. V programu je problém vyřešen pouhým výpočtem podprostoru, tvořeného hodnotami koalic, který je program schopen zobrazit pomocí intervalů přípustných zisků pro každou koalici. Každá imputace, která se vleze do všech intervalů, je v jádru hry. Postup výpočtu popisuje algoritmus č. 1:

Algoritmus **č**. 1: V**ý**po**č**et j**á**dra hry

```
input: set of structs COALITIONS, function value(coalition)

PLAYERS = COALITIONS: sizeof(coalition) == 1
foreach (coalition in COALITIONS):
        coalition_min = value(coalition)
        coalition_max = value(PLAYERS) - bestValueOf(N \ coalition)
foreach (coalition s,t: s∩t==Ø in COALITIONS):
        if(s∪t not equal to N):
                if(s∪t _max > s_max + t_max):
                        s∪t _max = s_max + b_max
foreach (coalition s,t: s∩t=Ø in COALITIONS):
        if(s∪t not equal to N):
                if(s_min < s∪t _min - t_max):
                        s_min = s∪t_min - t_max

output: set COALITIONS_MIN, set COALITIONS_MAX
```

Zdroj: vlastní

Funkce *bestValueOf* je v alespoň super-aditivní hře redukována na funkci *value*. Pokud hra není alespoň super-aditivní, můžete se stát, že hodnota koalice není nejvyšší hodnotou, jakou jsou hráči koalice schopni vytvořit. V takovém případě zmíněná funkce vypočítá nejvyšší dosažitelnou hodnotu ze všech možných uskupení členů koalice.

Druhý blok kódu přenáší maxima. Například pokud jsou dva hráči shora omezeni hodnotou *x*, pak jejich společná koalice musí být shora omezená hodnotou *2x*. Třetí blok kódu naopak přenáší minima. Například pokud je minimum společné koalice rovno *x* a jedna z koalic má maximum *y*, pak druhá koalice musí rozdíl vždy doplnit, neboli její minimum musí být alespoň *x - y*. V druhém a třetím bloku velmi záleží na pořadí průchodu koalicemi, což není pro jednoduchost v příkladu zachyceno.

2.2 Shapleyho hodnota

Koncept Shapleyho hodnoty (stejně jako Power indexy a Nukleolus) již nenabízí množinu, ale pouze jedno spravedlivé rozdělení. Shapleyho hodnota spravedlivé interpretuje jako rozdělení, kde každý člen koalice získá podíl odpovídající jeho zásluhám na zisku koalice. Obecně koncept předpokládá, že „Hráč se zúčastní koalice při imputaci, která mu přinese jeho ”spravedlivý” podíl. [3]“.

Shapleyho hodnota se odvíjí od přínosu hráče do koalice, který je třeba vypočítat jako „střední hodnotu přínosu hráče i do všech možných k-členných koalic. [3]“, kde přínos lze vypočítat jako rozdíl mezi hodnotou koalice s hráčem a bez hráče. Postup výpočtu popisuje algoritmus č. 2:

Algoritmus **č**. 2: V**ý**po**č**et Shapleyho hodnoty

```
input: set of structs COALITIONS, function value(coalition)

PLAYERS = COALITIONS: sizeof(coalition) == 1
foreach (player in PLAYERS):
        player_shapley = 0
        foreach (coalition: player ϵ coalition in COALITIONS):
                player_shapley = player_shapley + (
                                value(coalition) - value(coalition \ player)
                                ) / combination(sizeof(PLAYERS) above sizeof(coalition))
        player_shapley = player_shapley / sizeof(PLAYERS)

output: set PLAYERS_SHAPLEY
```

Zdroj: vlastní

Funkce *combination* vyjadřuje počet kombinací koalice ve velké koalici. Shapleyho hodnota je váženým průměrem přínosů hráče do koalic podle počtu koalic stejné velikosti v dané hře. I podle popisu v pseudokódu je vidět, že výpočet Shapleyho hodnoty je velmi snadný, je proto velmi rychlý při určování spravedlivého rozdělení zisku koalice.

2.3 Power Indexy

Power indexy jsou spojeny s podmnožinou kooperativních her, zvanou jednoduché hry. V jednoduchých hrách již koalice nemají ohodnocení v oboru reálných čísel, ale pouze v booleovské logice, která koalice dělí na vítěze a poražené. Koncept jednoduchých her se odvíjí od hlasování, ve kterém má každý hráč určitý počet hlasů a koalice je vítěznou pouze pokud její hráči mají dostatek hlasů k prosazení své vůle (rozhodující počet hlasů je dalším parametrem hry).

Power indexy lze v jednoduchých hrách použít k určení síly jednotlivých hráčů, která nemusí vždy být přímo úměrná počtu hlasů. V programu jsou použity dva způsoby výpočtu Power indexu, které jsou popsány v následujících kapitolách.

2.3.1 Banzhafův Power index

Tento koncept počítá sílu hráče jako podíl jeho veto hlasů na celkovém počtu veto hlasů. Veto hlas má hráč v každé vítězné koalici, která by se jeho vystoupením stala poraženou koalicí. Pseudokód výpočtu Banzhafova Power indexu popisuje algoritmus č. 3:

Algoritmus č. 3: Výpočet Banzhafova Power indexu

```
input: set of structs COALITIONS, function value(coalition)

PLAYERS = COALITIONS: sizeof(coalition) == 1
totalVetos = 0
foreach (player in PLAYERS):
        player_vetos = 0
foreach (coalition: value(coalition)==1 in COALITIONS):
        foreach (player in PLAYERS):
                if(value(coalition \ player)==0):
                        player_vetos = player_vetos + 1
                        totalVetos = totalVetos + 1
foreach (player in PLAYERS):
        player_power = player_vetos / totalVetos

output: set PLAYERS_POWER
```

Zdroj: vlastní

2.3.2 Shapley-Shubikův Power index

Shapley-Shubikův index namísto veto hlasů počítá vítězné hlasy, respektive podíl vítězných hlasů hráče na celkovém počtu takových hlasů. Vítězný hlas je vlastně opakem k veto hlasu, protože vítězný hlas má hráč v každé poražené, jež se jeho přidáním stává vítěznou koalicí.

Výpočet je trochu složitější, nežli v případě Banzhafova Power indexu, protože je nutné vytvořit všechny permutace hráčů, což je rekurzivní problém. V rámci každé permutace se zkoumá, který hráč v pořadí svým přistoupením do koalice získá pro koalici vítězství. Postup výpočtu popisuje algoritmus č. 4:

Algoritmus č. 4: Výpočet Shapley-Shubikova Power indexu

```
input: set of structs COALITIONS, function value(coalition)

PLAYERS = COALITIONS: sizeof(coalition) == 1
foreach (player in PLAYERS):
        player_wins = 0
foreach (sequence = permutation of PLAYERS):
        sequence_coalition = Ø
        foreach (player in PLAYERS):
                sequence_coalition = sequence_coalition ∪ player
                if(value(sequence_coalition)==1):
                        player_wins = player_wins + 1
foreach (player in PLAYERS):
        player_power = player_wins / factorial(sizeof(PLAYERS))

output: PLAYERS_POWER
```

Zdroj: vlastní

Proměnná *sequence* vyjadřuje posloupnost hráčů, jež je jednou z permutací množiny hráčů.

2.4 Nukleolus

„Výplatní vektor x je přijatelnější nežli y pouze pokud platí, že $\theta(x) \leq_{lex} \theta(y)$. Nechť Y je podmnožina R^n; nukleolus $N(Y)$ je množina bodů na Y, které jsou první v přijatelnostním kvazi-uspořádání na Y. Nukleolus hry nebo zkráceně Nukleolus je nukleolus množiny výplatních vektorů dané hry. [6]“. Ke kompletní definici je třeba uvést ještě definici uspořádaného vektoru excesů (θ) a definici lexikografického uspořádání ($\leq_{lex}$), základní formální povahu však lze z definice vyjádřit i tak. Nukleolus hry je podmnožinou imputací (podobně jako jádro), které jsou pro všechny hráče nejpřijatelnější, a právě přijatelnost je kritériem spravedlnosti pro koncept nukleolu.

Přijatelnost se v nukleolu kvantifikuje jako rozdíl mezi hodnotou koalice a jejím ziskem. Například přidělený zisk 7 je pro koalici s hodnotou 5 přijatelný s hodnotou -2, tato hodnota se nazývá exces koalice. Pokud excesy jedné imputace ve všech koalicích vyjma prázdné koalice seřadíme od největšího po nejmenší, pak získáme uspořádaný vektor excesů (θ). Přijatelnost vektorů se porovnává lexikografickým uspořádáním ($\leq_{lex}$), které porovnává prvky uspořádaných vektorů od začátku, přičemž jeden vektor je lexikograficky menší nebo roven, pokud žádný jeho prvek není větší, nežli odpovídající prvek v druhém vektoru. Formální definice zní takto [3]:

$$\theta_1 \leq_{\text{lex}} \theta_2 \begin{cases} \text{pravda} & \theta_1 = \theta_2 = \emptyset \\ \text{zbytek}(\theta_1) \leq_{lex} zbytek(\theta_2) & \max(\theta_1) \leq \max(\theta_2) \\ \text{nepravda} & \text{jinak} \end{cases}$$

Nyní už víme, jak porovnat imputace hry z hlediska přijatelnosti, avšak imputací, ze kterých nukleolus vybírá, může být ve spojitém prostoru nekonečně mnoho. Uspořádání spojitého prostoru je na počítači s diskrétním numerickým modelem neřešitelný problém, je tedy nutné z nekonečného prostoru imputací některé vytipovat a porovnání provést už pouze nad nimi. Popis algoritmu na vytipování imputací v pseudokódu by byl příliš složitý, pomůžeme si následující úvahou.

Mějme uspořádaný vektor excesů, jehož první dva prvky jsou si rovny. Pokud změníme imputaci takovým způsobem, že jeden z prvků vektoru excesů zvýší svou hodnotu a druhý ji sníží, pak nově vzniklý vektor excesů bude méně přijatelný kvůli prvku, který svou hodnotu zvýšil. Z toho také plyne, že extrémy nastávají právě při imputacích, jejichž vektor excesů obsahuje dva shodné prvky. Tyto imputace lze nalézt řešením soustav N rovnic o N neznámých, kde N je počet hráčů. Jednou z rovnic je přirozeně $\sum_{i \in N} a_i = v(N)$, kde a_i označuje zisk hráče i. Zbylé rovnice v každé soustavě jsou (N-1)-prvkové kombinace z množiny rovnic, jež vyjadřují rovnost dvou prvků vektoru excesů. Analytický výpočet pro dvou hráčovou hru demonstruje obrázek č. 2:

Obrázek č. 2: Příklad výpočtu nukleolu pro dvouhráčovou hru

Mějme hru s charakteristickou funkcí:

- v(ø) = 0
- v({1}) = 0,5
- v({2}) = 0
- v(N) = 1

Vektor excesů lze vypočíst jako (0,5-a_1; 0-a_2; 1-a_1-a_2 = 0). Množina rovnic vyjadřujících rovnost obsahuje následující rovnice:

- $0{,}5 - a_1 = -a_2$
- $0{,}5 - a_1 = 0$
- $-a_2 = 0$

Hra má dva hráče, budeme počítat tři soustavy dvou rovnic o dvou neznámých, kde jednou z rovnic bude $a_1 + a_2 = 1$, druhá rovnice bude vybrána z množiny získaných rovnic, čímž získáme tři imputace, které potenciálně mohou být v nukleolu hry:

- (0,75; 0,25)
- (0,5; 0,5)
- (1; 0)

Lexikografickým porovnáním zjistíme, že do nukleolu hry patří jen první imputace.

Zdroj: vlastní

Problémem tohoto algoritmu je jeho exponenciální složitost, neboť v dvouhráčové hře stačí vyřešit tři soustavy rovnic o dvou neznámých, zatímco v tříhráčové hře algoritmus řeší až 210 soustav tří rovnic o třech neznámých. Obecný výpočet počtu soustav v N-hráčové hře je $\binom{\binom{2^N-1}{N-1}}{N-1}$. Tento počet lze omezit jen vyřazením duplicitních rovnic (toto program dělá) nebo shlukováním rovnic do skupin, které se liší jen konstantou (například x = 4 a x = 5) a následným kombinováním rovnic v soustavách rovnic jen napříč skupinami. Například soustava {x + y + z = 10, x = 4, x = 5} nemá řešení a tudíž je její výpočet zbytečný. Sloučením všech rovnic typu x = k do jedné skupiny, jejíž členové se dostanou do každé soustavy v počtu maximálně jednoho, se algoritmus výrazně urychlí (toto program dělá).

Nukleolus jakožto množina může obsahovat více než jedno řešení, avšak stává se tak jen zřídka, například nukleolus hry, která je alespoň konvexní, obsahuje vždy pouze jednu imputaci [6]. Nejen z tohoto důvodu program počítá a zobrazuje pouze jedinou imputaci z nukleolu. Algoritmus je však navržen tak, aby byl s malou úpravou schopen najít všechny imputace z nukleolu.

3 Popis aplikace

Obsahem kapitoly je popis nástroje pro analýzu kooperativních her, vytvořeného v rámci této bakalářské práce. V první podkapitole bude popsána implementace knihovny na výpočet ukazatelé popsaných v předchozích kapitolách. Následující podkapitola popisuje grafické programové prostředí, které knihovnu využívá včetně návodu k obsluze.

3.1 Popis implementace

Projekt je zejména kvůli požadavku na přenositelnost implementován v jazyku Java v prostředí NetBeans 7.2.1 a dle zadání je rozdělen do knihoven (resp. balíků) CCL a Demo. CCL je balík implementující analytické výpočty ukazatelů, zatímco Demo obsahuje třídy pro uživatelské rozhraní a práci se soubory.

Dalším požadavkem na aplikaci byla výpočetní efektivita, která se v programu kromě jiného negativně promítá do bezpečnosti kódu, který často opomíjí kontroly vstupních souborů a interních datových reprezentací dle principu defenzivního programování. Efektivita se však v kódu zdaleka neprojevuje jen negativně, jak ostatně prokáže následující podkapitola.

3.1.1 Balík CCL

Výpočetní balík *CCL* obsahuje třídy:

- **CoopGame** - výpočty ukazatelů kooperativních her
- **StrategicGame** - převod strategických her na kooperativní
- **Utils** - pomocné výpočetní funkce

Třídy *CoopGame* a *StrategicGame* mají pro svou obsáhlost vyhrazeny zvláštní podkapitoly. Třída *Utils* obsahuje pomocné kombinatorické funkce (faktoriál, kombinace a s nimi související Pascalův trojúhelník), funkci *rref* na řešení soustav lineárních rovnic pomocí Gaussovy eliminace a některé aplikačně specifické funkce.

Celý balík *CCL* používá pro indexování koalic bitový index (v kódu bitIndex), který nejenže efektivně pracuje s prostorem, ale navíc velmi usnadňuje množinové operace sjednocení, průniku a dalších. Počet koalic je vždy $2^{|N|}$, protože každý člen v ní buď je, nebo není (1 nebo 0). Integrální typ Javy s šířkou 32b tak může pojmout všechny koalice hry do počtu 32 hráčů, programově je tento počet omezen na 20, protože vyšší počet hráčů by vytvářel ohromné nároky na paměť systému, což by v kombinaci s garbage collectorem Javy mohlo způsobit značné snížení výkonu celé aplikace.

Bitový index umožňuje realizovat pomocí jediného logického či aritmetického operátoru libovolnou množinovou operaci (AND ~ průnik, XOR -1 ~ doplněk, atd.).

3.1.1.1 Třída StrategicGame

Třída *StrategicGame* obsahuje pouze statické funkce, z nichž jediná funkce *NFGtoCG* je součástí veřejného rozhraní a slouží právě k převodu strategické hry ve formátu programu Gambit[1] na kooperativní hru tohoto programu. Aby se implementace vyhnula práci se soubory, vstupem této funkce jsou seřazené výplatní vektory ze souboru a vektor s počty strategií pro jednotlivé hráče, což v projektu obstarává třída *fileSystem* z balíku *Demo*.

NFGtoCG určuje hodnoty koalic (kromě prázdné a velké) jako nejnižší zaručený zisk hráčů koalice při konfrontaci se zbytkem hráčů. Pro každou koalici se sestaví nekooperativní hru dvou hráčů (zkoumané a opoziční koalice), jejichž množina strategií je kartézským součinem strategií členů koalice. Tyto nekooperativní hry dvou hráčů jsou sestavovány ze vstupních výplatních vektorů funkcí *reducedGame*, nejnižší zaručený zisk těchto her je počítán funkcí *minValueOfNEs*, která počítá zisky jak na základě čistých, tak i ze smíšených Nashových ekvilibrií.

Zatímco výpočet čistých Nashových ekvilibrií je triviální, smíšené strategie jsou na výpočet o něco náročnější, neboť se počítají vícekrát pro jednotlivé úrovně řešení. Ve hře dvou hráčů, kde oba hráči mají tři strategie, se nachází deset potenciálních smíšených ekvilibrií (jedno na úrovni 3 a devět na úrovni 2), které mohou a nemusí být smíšenými ekvilibrii. Výpočet smíšeného Nashova ekvilibria provádí funkce *probsOfMNE*, jejíž výsledek verifikuje funkce *valueOfStubMNE*. Konečně, pravděpodobnost jednotlivých strategií je získána řešením soustav lineárních rovnic. Jejich matematický popis pro prvního hráče lze vyjádřit takto:

- první rovnice: $\sum_{i \in S1} S_i = 1$
- j-tá rovnice: $\sum_{i \in S1} (M_{i12} - M_{ij2})$

Pro druhého hráče:

- první rovnice: $\sum_{i \in S2} S_i = 1$
- j-tá rovnice: $\sum_{i \in S2} (M_{1i1} - M_{ji1})$

M je matice zisků, jejíž třetí rozměr udává hráče, jehož zisk chceme. *S1* a *S2* jsou množiny strategií prvního, respektive druhého hráče. První rovnice vyjadřuje triviální pravidlo, že součet pravděpodobností strategií jednoho hráče je roven jedné. Další rovnice vyjadřují rovnost očekávaného užitku protihráče mezi první a *j*-tou strategií (*J* tak nabývá hodnot 2 až |S|). Obrázek č. 3 uvádí příklad nalezení rovnic pro tři strategie:

[1] www.gambit-project.org - softwarový nástroj na výpočet konečných nekooperativních her

Obrázek č. 3: Příklad výpočtu smíšeného Nashova ekvilibria

Strategie hráče 1 jsou A, B a C; hráč 2 má strategie X, Y a Z. Tyto symboly zároveň zastupují pravděpodobnosti příslušných strategií. Ostatní proměnné označují zisky (např.: *ax2* je zisk hráče 2, při uplatnění strategií A a X).

Soustava rovnic řešící pravděpodobnost strategií hráče 1 obsahuje rovnice:

- $A + B + C = 1$
- $A * (ax2 - ay2) + B * (bx2 - by2) + C * (cx2 - cy2) = 0$
- $A * (ax2 - az2) + B * (bx2 - bz2) + C * (cx2 - cz2) = 0$

Soustava rovnic řešící pravděpodobnost strategií hráče 2 obsahuje rovnice:

- $X + Y + Z = 1$
- $X * (ax1 - bx1) + Y * (ay1 - by1) + Z * (az1 - bz1) = 0$
- $X * (ax1 - cx1) + Y * (ay1 - cy1) + Z * (az1 - cz1) = 0$

Zdroj: vlastní

3.1.1.2 Třída CoopGame

Mnoho základních principů třídy *CoopGame* bylo již vysvětleno ve formě pseudokódu v předchozích kapitolách, tato podkapitola se proto bude věnovat zejména struktuře třídy a některým doposud neprobíraným oblastem. Pro práci s ní je vždy třeba vytvořit instanci, která neprovádí žádné výpočty, tudíž je její instalování nenáročné. Protože Java nemá ekvivalent datového typu záznam, jsou některé ukazatele (charakteristiky, jádro a nukleolus) předávány jako třídy, které jsou do *CoopGame* vestavěny. Třída obsahuje kromě konstruktorů tyto veřejné položky:

- funkci *shapleyVector*
- funkce *banzhafPowerIndex* a *shapleyShubikPowerIndex*
- třídu *Statistics* a funkci *cmpStatistics*
- třídu *Core* a funkci *cmpCore*
- třídu *Nucleolus* a funkci *cmpNucleolus*

Konstruktory umožňují instanciaci provést dvěma způsoby - seřazením podle bitového indexu, což je samozřejmě nejrychlejší, nebo předáním hodnot v libovolném pořadí a identifikací textovými řetězci. Výpočet charakteristik se provádí obohacením instance *CoopGame* metodou *cmpStatistics*, která instanciuje zanořenou třídu *Statistics*. Čtením jejích veřejný boolových a výčtových položek získáme charakteristiky hry.

Velmi podobně probíhá i výpočet jádra, tentokrát metodou *cmpCore* a vnořenou třídou *Core*. V instanci třídy lze z polí min a max číst intervaly, které společně s rovnicí $\sum_{i \in N} a_i = v(N)$ definují

jádro hry. Prázdnost jádra či existenci jediného řešení lze jednoduše ověřit metodami *isEmpty*, respektive *isSingle*, otestování imputace na přítomnost v jádře provádí metoda *isInGameCore*.

Pro výpočet Shapleyho hodnoty a Power indexů slouží samostatné metody *shapleyVector*, *banzhafPowerIndex* a *shapleyShubikPowerIndex*, které přímo vrací příslušné imputace, přičemž poslední dvě funkce z logických důvodů ověřují, zda je hra jednoduchá. Shapley-Shubikův Power index vyžaduje implementaci permutací pořadí hráčů, což program provádí rekurzivní funkcí *shapleyShubikPermutations*.

Stejně jako jádro a charakteristiky hry mají zvláštní zanořené třídy, tak i nukleolus má svou zanořenou třídu *Nucleolus* (inicializace metodou *cmpNucleolus*), která má jedinou veřejnou metodu *getImputation* vracející imputaci z nukleolu hry. Výpočet je však natolik složitý, že si zaslouží větší pozornost.

Prvním krokem výpočtu je vytvoření všech rovnic, které se pracovně dělí na pozitivní, které výplaty členů pouze sčítají (např.: $a_1 + a_4 = 3, a_3 = 1, ...$) a negativní, kde je nejvýše polovina členů odečítána (např.: $a_1 + a_3 - a_2 = 3.14, a_1 - a_2 = 7, ...$). Pravá strana pozitivní rovnice je tvořena odečtením hodnoty dvou koalic, jejichž průnikem je koalice reprezentovaná levou stranou rovnice pro všechny takové dvojice koalic (např.: $a_1 + a_2 = v(\{1,2\}) - v(\emptyset), a_1 + a_2 = v(\{1,2,3\}) - v(\{3\}), ...$). Situace je u negativní rovnice podobná, jen se zde odečítají hodnoty kladné a záporné koalice z levé strany (např.: $a_1 - a_3 = v(\{1\}) - v(\{3\}), a_1 - a_3 = v(\{1,2\}) - v(\{2,3\})$). Nalezené pozitivní i negativní rovnice jsou následně roztřízeny do skupin (jak bylo popsáno v podkapitole 2.4), jejichž prostor se následně projde rekurzivní funkcí *recNucleolusSolver*. Řešení získaných soustav lineárních rovnic nabídne imputace, které se potenciálně nacházejí v nukleolu hry, lexikograficky nejmenší vektor excesů označí hledanou imputaci z nukleolu.

3.1.2 Balík Demo

Balík *Demo* obsahuje uživatelské rozhraní demonstrující výpočetní činnost balíku *CCL*. V balíku *Demo* se nacházejí třídy:

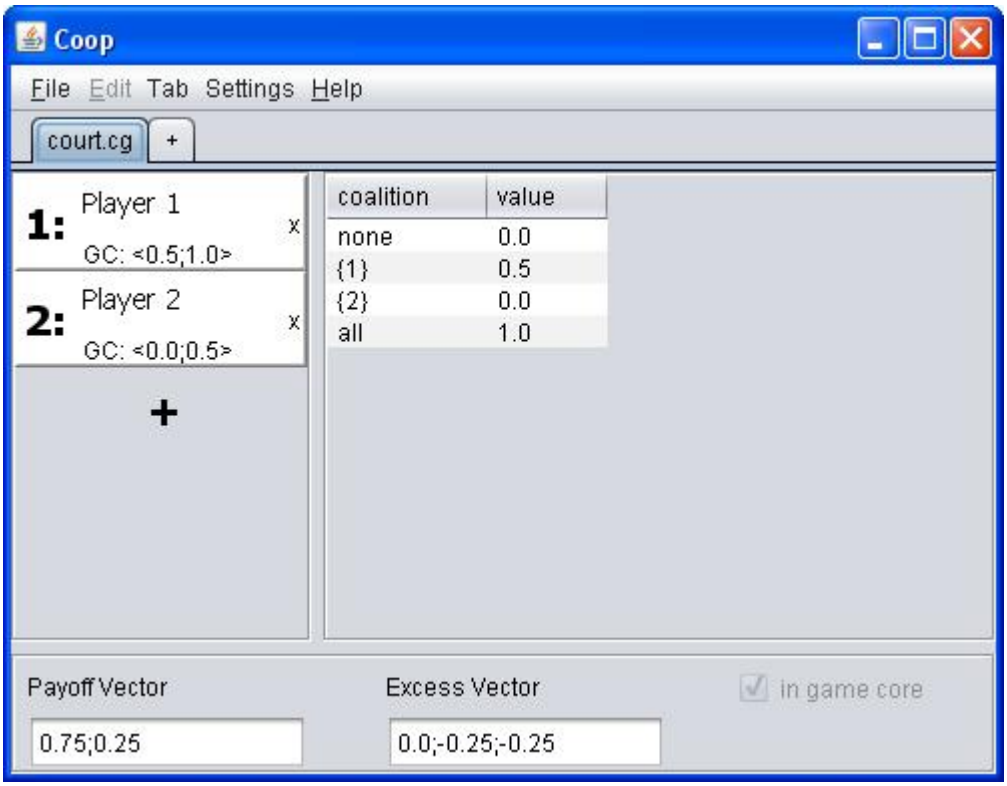
- **GUI** - implementace grafického uživatelského rozhraní
- **fileSystem** - práce se souborovým systémem

K zajištění ukládání a načítání souborů slouží třída *fileSystem*, která pracuje se soubory kooperativních her s koncovkou „.cg“, a umožňuje otevření nekooperativních her programu Gambit s koncovkou „.nfg“. Struktura souboru kooperativní hry obsahuje pouze řádek s počtem koalic, následují řádky se jmény hráčů a na konci jsou řádky s hodnotami koalic, seřazené podle velikosti koalice. Zatímco balík *CCL* používá bitový index pro urychlení výpočtu, balík *Demo* používá primárně indexování podle velikosti koalic, neboť je toto řazení lépe čitelné pro uživatele. Třídu *GUI* nemá smysl dále rozebírat, neboť se jedná o implementaci demonstračního grafického uživatelského prostředí.

3.2 Návod k obsluze

Uživatelské rozhraní programu obsahuje lištu s nabídkami File (Soubor), Tab (Záložka), Settings (Nastavení) a Help (Nápověda) a pracovní plochu rozdělenou do záložek. Každá záložka reprezentuje jednu otevřenou kooperativní hru, jejíž název se zobrazuje v hlavičce záložky. Podrobný postup k vytvoření hry z obrázku č. 4 v prostředí programu lze najít v příloze A.

Obrázek č. 4: Prostředí demonstrační aplikace



Zdroj: vlastní

3.2.1 Lišta nabídek

Nabídka File obsahuje položky:

- New (Nový) - Vytvoří novou prázdnou hru; stejně lze vytvořit prázdnou hru kliknutím na záložku se symbolem „+“.
- Open (Otevřít) - Otevře uloženou kooperativní hru (koncovka „.cg“) nebo transformuje nekooperativní hru programu Gambit (koncovka „.nfg“) na novou kooperativní hru.
- Exit (Konec) - Ukončí program. Pokud některé otevřené hry obsahují neuložené změny, program nejprve nabídne jejich uložení.

Nabídka Tab se vždy vztahuje na aktuálně zvolenou otevřenou hru, její položky jsou:

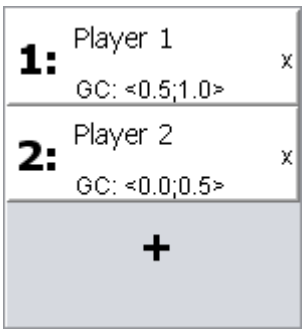
- Save (Uložit) - Uloží aktivní hru do stávajícího souboru. Pokud se jedná o novou hru, která ještě nebyla uložena do souboru, postup je snejný jako u Save as.
- Save as (Uložit jako) - Program otevře okno pro uložení hry do nového souboru.
- Close (Zavřít) - Zavře aktivní hru a případně vyzve uživatele k uložení neuložených změn. Pokud zbývá na pracovní ploše jediná záložka, tlačítko Close se zneplatní.
- Game characteristics (Herní charakteristiky) - Vytvoří nové nemodální okno s výčtem charakteristik aktivní hry.
- Solve game's Core (Vypočti herní jádro) - Vytvoří nemodální okno s intervaly, které popisují jádro hry.
- Solve (Vypočti) - Vypočte příslušnou hodnotu, uloží ji do pole výplatního vektoru a nastaví ji jako vlastnost hráče s příslušným identifikátorem (SV, BPI, SSPI, NUC). Power Indexy lze vypočítat jen pro jednoduché hry. Na výběr jsou tyto hodnoty:
    - Shapley value (Shapleyho hodnotu)
    - Banzhaf Power index (Banzhafův Power index)
    - Shapley-Shubik Power index (Shapleyův-Shubikův Power index)
    - Nucleolus (nukleolus) - Pozor, výpočet nukleolu pro více jak čtyři hráče je velmi náročnou operací; výpočet hry s pěti hráči může trvat i několik minut!

Nabídka Settings aktuálně umožňuje pouze nastavení preferovaného počtu řádků v seznamu koalic tlačítkem Prefered Row Count (Preferovány počet řádků). Zadáním hodnoty 0 se bude počet řádků upravovat podle velikosti okna.

3.2.2 Pracovní plocha

Pracovní plocha každé otevřené hry se sestává ze seznamu hráčů, seznamu koalic a panelu pro zadání výplatního vektoru. Při zadávání čísel do programu je nutné vždy používat desetinnou tečku, jinak nedojde v upravovaném poli ke změně.

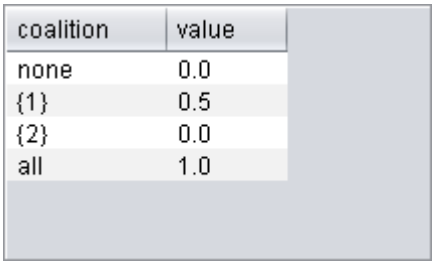
Obrázek č. 5: Seznam hráčů v programu



Zdroj: vlastní

Seznam hráčů (viz obrázek č. 5) nabízí přehled o hráčích aktivní kooperativní hry - jejich názvu, zástupného čísla, které se zobrazuje v seznamu koalic a řádek vlastností, do kterého se ukládají výsledky výpočtů. Přidat dalšího hráče lze kliknutím na symbol „+“ pod posledním hráčem v panelu, odebrat hráče ze hry lze kliknutím na symbol „x“ napravo od jména hráče. V obou případech dojde i k aktualizaci seznamu koalic. Jméno hráče lze upravit přepsáním v seznamu hráčů a jedná se o jediný upravitelný atribut hráče, který se zároveň ukládá do souboru.

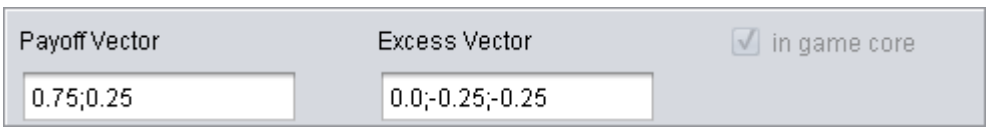
Obrázek č. 6: Seznam koalic v programu



Zdroj: vlastní

Seznam koalic (viz obrázek č. 6) reprezentuje charakteristickou funkci hry a jako takový mapuje koalice hry na hodnoty. Sestává se z dvojic sloupců (zástupné symboly koalice a hodnota), přičemž upravit lze jen sloupce s hodnotami, a to dvojklikem na příslušnou buňku v tabulce (trojklikem se označí aktuální hodnota a lze ji rovnou přepsat). Identifikátorem koalice může být také symbol none (nikdo), který označuje prázdnou koalici a all (všichni), který označuje velkou koalici všech hráčů.

Obrázek č. 7: Panel výplatních vektorů v programu



Zdroj: vlastní

Panel pro zadání výplatních vektorů (viz obrázek č. 7) obsahuje upravitelné pole Payoff Vector (výplatní vektor) a dvě neupravitelné položky - pole Excess Vector (vektor excesů) a zaškrtávací políčko in game core (je v jádře). Po zadání výplatního vektoru se obě neupravitelné položky aktualizují, čímž zobrazí vektor excesů daného výplatního vektoru a otestují jej na přítomnost v jádře. Jednotlivé zisky se v poli oddělují středníkem, nezapsané zisky se nahradí nulou, zisky zapsané nadbytečně jsou ignorovány.

4 Experimenty

Vytvořením programu vznikl nástroj k výpočtu různých ukazatelů z kooperativních her, s jehož využitím bylo provedeno několik experimentů popsaných dále. Jedná se zejména o příklady demonstrační, které dále vysvětlují koncepty forem řešení a jejich použití v praxi.

4.1 Ukázkové příklady

CD této bakalářské práce obsahuje i ukázkové příklady ke všem formám řešení, které program dokáže zpracovat. Ke každé formě řešení existuje pět přiložených ukázkových příkladů, z nichž jeden pro každou z forem je blíže popsán v některé z následujících podkapitol. Na závěr jsou formy řešení srovnány na všech prezentovaných příkladech.

4.1.1 Jádro - robbery.cg

Tři lupiči Alex, Bill a Carl si z poslení loupeže odnesli 100000 dolarů, z nichž si zbývá rozdělit ještě 35000 dolarů. Šéf Alex drží podíl 30000 dolarů, protřelý Bill drží 20000 dolarů a Carl pobral 15000 dolarů. Alex s Billem se znají dost dlouho a celkový podíl nižší než 80000 dolarů by mohl zapříčinit, že s lupem společně uprchnou. Jakou sumu může každý z nich celkově získat?

Řešení tohoto příkladu vede přirozeně na jádro kooperativní hry tří hráčů, jehož vyřešením získáme intervaly možných zisků hráčů. Hodnoty koalic shrnuje tabulka č. 3:

Tabulka č. 3: Hodnoty koalic příkladu robbery.cg

ø	Alex	Bill	Carl	Alex+Bill	Alex+Carl	Bill+Carl	N
0	30000	20000	15000	80000	45000	35000	100000

Zdroj: vlastní

Jádro hry obsahuje imputace, které Alexovi nabízí od 30000 do 65000 dolarů, Bill může získat 20000 až 55000 dolarů a Carl si přijde na 15000 až 20000 dolarů. Součet horních hranic dává v součtu 140000 dolarů, takže nikdo z lupičů nemá zaručenou horní hranici zisku. Kolik ve výsledku získají, lze určit Shapleyho hodnotou či nukleolem (viz podkapitola 4.1.5).

4.1.2 Shapleyho hodnota - project.cg

„Předpokládejme situaci šéfa a čtyř zaměstnanců. Řeší se projekt, který ztroskotá bez účasti šéfa. Každý zaměstnanec má přínos do projektu 50000 korun, šéf má přínos nulový. Jak spravedlivě rozdělit zisk z projektu? [3]“.

V této kooperativní hře má šéf paradoxní pozici, neboť jeho přínos do koalice je sice nulový, ale bez něj má koalice vždy nulovou hodnotu. Charakteristickou funkci zde lze definovat matematicky jako:

$$v(S) = \begin{cases} 50000 * (|S| - 1) & \text{šéf} \in S \\ 0 & jinak \end{cases}.$$

Vyřešením Shapleyho hodnoty získáme imputaci, která šéfovi přiřazuje zisk 100000 korun a každému ze zaměstnanců dává 25000 korun. Obecné řešení pro libovolný počet zaměstnanců říká, že zaměstnanec by měl v podobných hrách vždy odevzdat polovinu svého zisku šéfovi, který tak získá polovinu veškerého zisku [3].

4.1.3 Power index - council.cg

„Předpokládejme rozhodování většinou hlasů v radě sestávající se ze stran A, B, C a D, které mají 3, 2, 1 a 1 (v tomto pořadí) hlasů. Hraniční počet hlasů pro přijetí rozhodnutí jsou 4 hlasy. [7]“.

Jak již bylo řečeno, Power indexy odhalují síly jednotlivých hráčů v jednoduchých hrách. V této hře se na první pohled nachází tři skupiny hráčů s různými schopnostmi ovlivnit výsledek hlasování. Vítězné koalice (s ohodnocením 1) jsou v tomto příkladě dvoučlenné koalice s hráčem A, tříčlenné koalice a velká koalice. Už z popisu charakteristické funkce lze vytušit, že strany B, C a D mají rovnocenný vliv, což potvrzuje i výpočet libovolného Power indexu. Polovinu moci v radě má strana A, o zbytek se rovným dílem dělí ostatní strany, takže strana B nemá větší moc než strana C, byť má dvojnásobný počet hlasů.

Paradoxně, sloučením strany C a D by se nezměnila její síla v radě, ale jen by se přerozdělila síla zbývajících dvou stran ve prospěch strany B. V realitě se však může stát, že se zástupci jedné strany nedohodnou a hlasují pro různé varianty, s čímž koncept jednoduchých her nepočítá. Vypočtené síly stran proto nelze v realitě brát absolutně.

Stejného výsledku lze dosáhnout i výpočtem Shapleyho hodnoty. Výhodou výpočtu Power indexů je počítání v celých číslech, což se ale v programu bohužel neděje.

4.1.4 Nukleolus - court.cg

„Dva drží jeden kus oděvu. Jeden volá: „celý je můj”, druhý volá: ”má je polovina”. Potom první dostane tři čtvrtiny a druhý jednu čtvrtinu. *Talmud*

Rabi Raši (1105) komentuje řešení: druhý přiznává prvnímu polovinu, ta je tedy jasná. Zbývá rozdělit tu druhou polovinu rovným dílem. [3]“.

Zadání tohoto příkladu krásně demonstruje podstatu nukleolu - vyhledání řešení, které vzbudí nejméně námitek zúčastněných stran. Pro jistotu si řešení z Talmudu ještě ověříme výpočtem nukleolu kooperativní hry popsané tabulkou č. 4:

Tabulka č. 4: Hodnoty koalic příkladu court.cg

ø	Jeden	Druhý	Oba
0	0,5	0	1

Zdroj: Příklad z historie [3]

Zisk prvního hráče je dle nukleolu skutečně 0,75 a druhého 0,25. Stejnou hodnotu lze získat i vypočtením Shapleyho hodnoty, avšak není tomu tak v každé kooperativní hře. Při experimentování s programem tyto situace nastávaly ve hrách konvexních či aditivních, avšak jedná se pouze o hypotézu, kterou by bylo nutné nejprve dokázat či vyvrátit.

4.1.5 Srovnání forem řešení

Zatímco jádro nenabízí jediné spravedlivé řešení a Power Index lze použít jen na jednoduché hry, výpočtem Shapleyho hodnoty či nukleolu lze spravedlivé řešení najít vždy, i když ne vždy bývá pro obě formy řešení shodné. Srovnání výsledků Shapleyho hodnoty a nukleolu z předchozích příkladů nabízí tabulka č. 5:

Tabulka č. 5: Srovnání Shapleyho hodnoty a nukleolu na příkladech z kapitoly 4.1

	robbery.cg	project.cg	council.cg	court.cg
Shapleyho hodnota	46666⅔, 36666⅔, 16666⅔	100000, 25000, 25000, 25000	½, ⅙, ⅙, ⅙	¾, ¼
nukleolus	46250, 36250, 17500	100000, 25000, 25000, 25000	⅖, ⅕, ⅕, ⅕	¾, ¼

Zdroj: vlastní

Hned první příklad naznačuje rozdíl mezi „spravedlností“ v podání Shapleyho hodnoty a nukleolu. Obecně nelze říci které řešení je lepší, neboť se obě nacházejí v jádru hry a jsou tedy pro všechny hráče přijatelná.

Druhý příklad vykazuje stejné hodnoty, což by mohlo být způsobeno konvexností hry. Tvrzení je pouze hypotetické, neboť vztah konvexní hry implikující shodnost Shapleyho hodnoty a nukleolu v této práci nebyl dokázán.

Byť je rozdělení volební síly podle nukleolu zajímavé, nemá v tomto kontextu zvláštní význam, neboť reálně volební sílu hráče neodráží. Rozdělení podle nukleolu by se možná dalo použít na problémy typu rozdělení ministerstev ve vládní koalici apod.

Každý problém, kde figurují pouze dva hráči, má shodné řešení pro Shapleyho hodnotu i nukleolus. V dvouhráčových hrách tedy lze náročný výpočet nukleolu vždy nahradit výpočtem Shapleyho hodnoty.

4.2 Alokační modely

„Nechť *U* je množina hráčů. Alokační úlohou rozumíme hru (N,*c*), kde N je koalice a *c* (koaliční funkce) je nákladovou funkcí úlohy. N reprezentuje množinu potenciálních zákazníků veřejné služby či statku. Každý zákazník může být obsloužen nebo nemusí (pozn. v závislosti na existenci služby či statku). Nechť S je podmnožinou N. Potom *c*(S) vyjadřuje nejnižší možnou cenu, za kterou lze obsloužit členy S. Hra (N,*c*) je alokační hrou. [2]“. Jinými slovy v alokačních úlohách se účastníci snaží namísto maximalizace zisků minimalizovat náklady, což je koneckonců jedno a to samé. Alokační hru lze transformací *v*(x)=zisk(x)-*c*(x) vždy převést na běžnou kooperativní hru.

4.2.1 Těžební kvóty zemí OPEC

OPEC je organizací sdružující vybrané země produkující ropu do nejznámějšího kartelu na světě. Jednou z politik OPECu je určování produkčních kvót jak členským zemím, tak i celé organizaci za účelem udržení vysoké ceny ropy. Nastavení spolupráce členských zemí OPEC zkoumaného roku 2005 popisuje obrázek č. 6.

Tabulka **č**. 6: Nastavení produkce OPECu v roce 2005

	Kvóta	Produkce	Kapacita
Saudská Arábie	10099	9800	12500
Irán	4110	3700	3750
Kuvajt	2247	2500	2600
Spojené Arabské Emiráty	2444	2500	2600
Venezuela	3225	2340	2450
Nigérie	2306	2250	2250
Angola	1900	1700	1700
Libye	1500	1650	1700
Alžír	894	1360	1430
Katar	726	810	850
Ekvádor	520	500	500
CELKEM	29971	29110	32330

Zdroj: Internet, Wikipedia: OPEC [8]

Tabulka zobrazuje (v tisících barelech) denní kvótu, průměrnou denní produkci a denní kapacitu produkce každého státu. Ze souhrnných čísel je vidět potenciál zemí, převyšující aktuální produkci o 3,22 milionů barelů denně. Trh s ropou má zvláštní charakter - při zvýšení ceny se poptávaný objem příliš nesníží (ekonomika musí běžet dál), avšak při zvýšení nabízeného objemu se

cena ropy znatelně propadá (co s přebytečnou ropou?). Stabilní nárůst nabízeného množství ropy o 3,22 milionů barelů denně (+ 4,49%) by poslal cenu ropy hodně hluboko.

Cílem následujícího experimentu (soubor „opec.cg“) je navrhnout lepší rozdělení produkčních kvót a ověření celkové stability systému. Uvažujme kooperující koalice jako sdružení zemí, které se rozhodly omezit svou produkci (celkově na 90% kapacity; přibližně jako v roce 2005) a vytvořit OPEC, jejich hodnoty jsou celkové tržby koalice. Nekooperující koalice je naopak taková, že jako jediná dohodu OPEC poruší, zatímco ostatní členové ji respektují. Tržby koalice se počítají podle následujících vzorců:

$$R_n(S) = cap(S) * P * d(S)$$
$$R_c(S) = cap(S) * 0{,}9 * P * d(N \backslash S)$$

$$d(S) = 1 - \left(\frac{0{,}1 * cap(S)}{Q}\right)^2 * 100 = 1 - \frac{cap(S)^2}{Q^2}$$
$$cap(S) = \sum_{i \in S} capacity_i$$

Výraz ***cap(S)*** vyjadřuje celkovou kapacitu produkce koalice, ***d(S)*** je diskontní faktor ceny, související se zvýšenou produkcí, P je výchozí cena 50$ [9] a Q je výchozí objem produkce 72 mil. barelů denně [10]. Tržby jsou značeny výrazy $R_n(S)$ pro nekooperující koalice, respektive $R_c(S)$ pro kooperující koalice. Strany kvůli jejich množství pro jednoduchost sloučíme do skupin naznačených v tabulce přerušovanou čarou. Toto rozdělení záměrně tvoří nesymetrické koalice, které lépe odhalí vyjednávací pozice různě velkých producentů. Tabulka č. 7 zobrazuje tržby, aneb hodnoty koalic v nekooperativní i kooperativní variantě (pozn.: první sloupec zobrazuje koalice pomocí jejich zástupných čísel):

Tabulka **č**. 7: Tr**ž**by koalic v p**ří**kladu OPEC

	Produkce (tbd)				Potenciální zisk (tis. $)	
	kapacita	90%	rozdíl	relativní rozdíl	nekooperativní	kooperativní
1	16250	14625	1625	2,26%	771 113	730 885
2	7650	6885	765	1,06%	378 182	343 846
3	5650	5085	565	0,78%	280 760	253 901
4	2780	2502	278	0,39%	138 793	124 889
1+2	23900	21510	2390	3,32%	1 063 326	1 075 353
1+3	21900	19710	2190	3,04%	993 693	985 293
1+4	19030	17127	1903	2,64%	885 031	856 058
2+3	13300	11970	1330	1,85%	642 309	598 082
2+4	10430	9387	1043	1,45%	510 556	468 916
3+4	8430	7587	843	1,17%	415 722	378 932
1+2+3	29550	26595	2955	4,10%	1 228 627	1 329 730
1+2+4	26680	24012	2668	3,71%	1 150 827	1 200 526
1+3+4	24680	22212	2468	3,43%	1 089 009	1 110 475
2+3+4	16080	14472	1608	2,23%	763 898	723 231
N	32330	29097	3233	4,49%	1 290 572	1 454 850

Zdroj: vlastní

Výpočtem jádra v programu zjistíme, že je sice neprázdné, avšak prostor pro vyjednávání je velice omezený. Pro největšího hráče nabízí jádro hry zisk v intervalu od 731 031 000 do 731 619 000 dolarů, což odpovídá kvótě 14 620,62 až 14 632,38 tbd - rozdíl pouhých 11760 barelů denně. U nejmenší koalice je tento rozdíl dokonce jen 4620 barelů denně. Nárůst tržeb zvyšováním počtu členů je velmi malý, obecně význam sloučení roste s rostoucím počtem členů.

Výpočtem Shapleyho hodnoty v kooperativní variantě koalic získáme spravedlivé rozdělení tržeb mezi členy, potažmo (vydělením cenou barelu ropy) jim přidělené kvóty, které se jen nepatrně liší od 90% jejich kapacity. Spravedlivého rozdělení pomocí nukleolu jde ve zvýhodnění velkých producentů ještě dál, jak ukazuje tabulka č. 8:

Tabulka č. 8: Srovnání rozdělení kvót k příkladu OPEC

	Shapleho hodnota	nukleolus	stávající rozdělení
Saudská Arábie, Irán	14627,3	14629,18	14209
Kuvajt, SAE, Venezuela	6884,24	6884,96	7916
Nigérie, Angola, Libye	5084,12	5083,76	5706
Alžír, Katar, Ekvádor	2501,34	2499,9	2140

Zdroj: vlastní

Navržená rozdělení se zdají být lepšími než to stávající, kde některé země obdrží kvótu vyšší, než kolik dovedou vyprodukovat a některé země jsou naopak omezeny o více jak třetinu, což jistě významným způsobem nabourává morálku zemí v dodržování kvót.

Na druhou stranu nekooperativní zisky koalic naznačují, že dodržování kvót pro ně není výhodné, neboť vystoupením z velké koalice zpravidla mohou získat více, což činí celé uskupení velice nestabilním. Organizace dosáhne kolektivně nejvyšších zisků sloučením všech výrobců na světě, avšak pro jednotlivce je nejvýhodnější donutit zbytek světa ke sloučení, zatímco sám bude vyrábět na hranici svých možností. Tento postup je podle nekooperativních tržeb koalic nejvýhodnější pro malé producenty.

V experimentu je vidět, že nejvýhodnějším řešením pro producenty je řady zemí OPECu rozšířit na maximální možnou míru a vytvořit mechanizmy, které všechny země účinně odradí od nadprodukce nebo vykazování vyšších než skutečných kapacit. Návrh takových mechanizmů je však doménou spíše nekooperativních her. Situace by se mohla změnit, pokud by nastal výraznější nesoulad mezi nabízeným a poptávaným množstvím, který by malé producenty motivoval k porušení kvót i přes veškerá opatření.

4.2.2 Zpoplatnění studia na vysokých školách v ČR

Ještě před několika měsíci bylo zpoplatnění studia na VŠ velice žhavým tématem. Návrh na zpoplatnění jednoho semestru studia na VŠ částkou 10000 Kč na studenta „zvedlo ze židle“ velkou část občanů (zejména těch s tristní znalostí ekonomie). V rámci tohoto experimentu bych chtěl vytvořit vhodnou cenově-diskriminační politiku, která by mezi občany vzbudila menší pozdvižení. Cenová diskriminace by mohla být uplatňována například dle nějakého objektivního hodnocení jednotlivých škol, kdy kvalitnější škola by měla poplatek vyšší a naopak.

„Vzbudit menší pozdvižení“ intuitivně vede na řešení nukleolem, pojďme však nejdřív definovat hru (soubor „university fee.cg“). Přínos pro každého studenta je individuální, tento experiment je založen na vlastním odhadu struktury studentů, který je popsán tabulkou č. 9:

Tabulka č. 9: Odhadované rozložení studentů VŠ podle jejich ocenění studia

	Podíl (%)	Přínos jednoho semestru studia (Kč)	
		individuální	celkový
skupina A	50	5000	1 000 000 000
skupina B	42	20000	3 360 000 000
skupina C	8	50000	1 600 000 000

Zdroj: vlastní

Sloupec podíl udává procentuelní zastoupení skupiny studentů na celkovém počtu 400 tisíc [11] v roce 2010, sloupec přínos pak značí ocenění jednoho semestru studia na VŠ studenty dané skupiny. Při takovém rozložení by celkový výběr školného činil 2 mld. korun, neboť částku 10000 Kč by byla ochotna zaplatit pouze polovina studentů. Stanovme v tomto příkladě částku 2 mld. Kč jako minimální cenu statku, která podmiňuje fungování vysokého školství. Vypočtené hodnoty koalic zobrazuje tabulka č. 10, hodnota koalice je definována výrazem $max(0;\ celkový\_přínos\_koalice - 2 * 10^9)$

Tabulka č. 10: Hodnoty koalic z příkladu zpoplatnění studia na VŠ

	Přínos pro koalici (Kč)	Hodnota koalice (Kč)
A	1 000 000 000	0
B	3 360 000 000	1 360 000 000
C	1 600 000 000	0
A+B	4 360 000 000	2 360 000 000
A+C	2 600 000 000	600 000 000
B+C	4 960 000 000	2 960 000 000
N	5 960 000 000	3 960 000 000

Zdroj: vlastní

Výpočtem některé z forem řešení lze získat spravedlivé rozdělení zbylého přínosu jednotlivých skupin (cenu, kterou by skupina byla ještě ochotna obětovat, ale nemusí, neboť náklady jsou již pokryty). Zbylý přínos tvoří jakousi slevu hráče z jím původně nabízené hodnoty. Řešení hry Shapleyho hodnotou a nukleolem zobrazuje tabulka č. 11:

Tabulka **č**. 11: Poplatky za studium

	Řešení Shapleyho hodnotou (Kč)	Řešení nukleolem (Kč)
Skupina A	2000	2333
Skupina B	5357	2778
Skupina C	21875	33333

Zdroj: vlastní

Pro srovnání ještě uvádím vektory excesů řešení Shapleho hodnotou a nukleolem:

Shapleyho hodnota: (0; -400; -600; -700; -900; -900; -1100)

nukleolus: (0; -466⅔; -466⅔; -533⅓; -533⅓; -1066⅔; -1533⅓;)

Řešení nukleolem je poměrně kontroverzní, avšak vždy záleží na způsobu aplikace cenové diskriminace. Například při zpoplatnění pouze dvou nejprestižnějších škol v republice nejvyšší částkou by mohlo být takové řešení přijato, ideální by bylo zpoplatnění, které by například exponenciálně rostlo v závislosti na kvalitě školy. Každopádně by v modelovém příkladě bylo uspokojeno více studentů, vytvořil by se tlak na kvalitu škol a zároveň by se kvalitnější vzdělání dostalo těm studentům, kteří si ho více váží (nebo jejichž rodiče mají tučné konto, bohužel). Samozřejmě by musely vzniknout mechanizmy, které by umožnily nést chudším nadějným studentům vyšší náklady (stipendia, studentské půjčky). Tabulka č. 12 uvádí pro srovnání dvě alternativní situace, kde se některá ze skupin odmítla podílet na poplatcích za studium.

Tabulka **č**. 12: Poplatky za studium p**ř**i ne**úč**asti vybran**ý**ch skupin student**ů**

	Neúčast skupiny A		Neúčast skupiny C	
	SH	nukleolus	SH	nukleolus
Skupina A			2500	2500
Skupina B	7143	7143	8929	8929
Skupina C	25000	25000		

Zdroj: vlastní

Závěr

V práci byly představeny principy a postupy analytických výpočtů vybraných forem řešení a vlastností kooperativních her, které nástroj implementuje, včetně jejich algoritmické podoby. Právě implementace výpočtu popsaných forem řešení se ukázala být největší překážkou při tvorbě programu, neboť vyžadovala detailní nastudování složitých matematických postupů, nejčastěji dostupných pouze v angličtině. V případě nukleolu si implementace kvůli nedostatku informací vyžadovala matematické odvození podpořené dedukcí z konkrétních příkladů. Dále byla činnost programu demonstrována na sérii experimentů, včetně dvou aktuálních případových studií z oblasti alokačních modelů.

Požadavek na implementaci nástroje ve formě knihovny a demonstrační aplikace byl dodržen, neboť se jedná o oddělené Javovské balíky s jednosměrnou závislostí demonstrační aplikace na výpočetní knihovně. Požadavek na výpočetní efektivitu se nesl celou implementací a ve většině programu byl dodržen. Nukleolus se z tohoto pravidla jediný vymyká, náročnost jeho algoritmu je exponenciální a výpočet pro vyšší počet hráčů trvá nepřiměřeně dlouho, což nabízí prostor pro jeho další vylepšení. Požadavek na přenositelnost programu je díky implementaci v Javě zcela zřetelně dodržen a navíc je program velmi kompaktní. Rozhodně se tedy jeví jako použitelný nástroj pro analýzu kooperativních her.

Z hlediska dalšího vývoje projektu se nabízí hned několik vylepšení stávajícího programu, zejména optimalizace algoritmu pro výpočet nukleolu, ale i dalších ukazatelů. Například počítání Power indexu v integrálních číslech pro zlepšení výkonu programu ve hrách s velkým počtem hráčů. Implementace využívá číselný typ double, který je náchylný na zaokrouhlovací chyby, které v některých částech programu způsobují zhoršení přesnosti. Převod celého programu k použití zlomkových nebo přesných finančních číselných typů by tento problém spolehlivě odstranil.

Kromě technických vylepšení by program mohl být vylepšen i usnadněním obsluhy, zlepšením grafické stránky a vytvořením lehkého klienta pro příkazovou řádku. Aplikace nyní obsahuje pár načatých konceptů, jako podporu undo/redo operací, jiné zobrazení hodnot koalic, podpora pro tisk výstupu a usnadnění zadávání hodnot koalic.

Literatura

[1] **Peleg, B., Sudhölter, P.** *Introduction ti the Theory of Cooperative Games.* Berlin : Springer, 2007. ISBN 978-3-540-72944-0.

[2] **Hrubý, M.** *Kooperativní hry a vyjednávání.* [Prezentace] Brno : Vysoké učení technické v Brně, 2011.

[3] **Osborne, M. J., Rubinstein, A.** *A Course in Game Theory.* Massachusetts : The MIT Press, 1994. ISBN 0-262-65040-1.

[4] **Spaniel, W.** The Mixed Strategy Nash Equilibrium Algorithm. *Game Theory 101*. [Online] 2013. http://gametheory101.com/Mixed_Strategies.html.

[5] **Schmeidler, D.** The Nucleolus of a Characteristic Function Game. *SIAM Journal on Applied Mathematics.* 1969, Sv. 17, 6.

[6] **Lihn, S.** Shapley–Shubik power index. *Wikipedia.* [Online] 2013. http://en.wikipedia.org/wiki/Shapley%E2%80%93Shubik_power_index.

[7] **Koslowski, O.** OPEC. *Wikipedia.* [Online] 2013. http://en.wikipedia.org/wiki/OPEC.

[8] **McMahon, T.** Historical Oil Prices (Table). *InflationData.com.* [Online] 2013. http://inflationdata.com/Inflation/Inflation_Rate/Historical_Oil_Prices_Table.asp.

[9] **IndexMundi.** World Crude Oil Production and Consumption by Year (Thousands Barrels per Day). *IndexMundi.* [Online] 2012. http://www.indexmundi.com/energy.aspx?region=xx&product=oil&graph=production+consumption.

[10] **Český statistický úřad.** Studenti a absolventi vysokých škol v ČR celkem. *Český statistický úřad.* [Online] 2011. http://www.czso.cz/csu/redakce.nsf/i/studenti_a_absolventi_vysokych_skol_v_cr_celkem/$File/1_VS_studenti_celkem_11.pdf.

Seznam algoritmů

Seznam tabulek

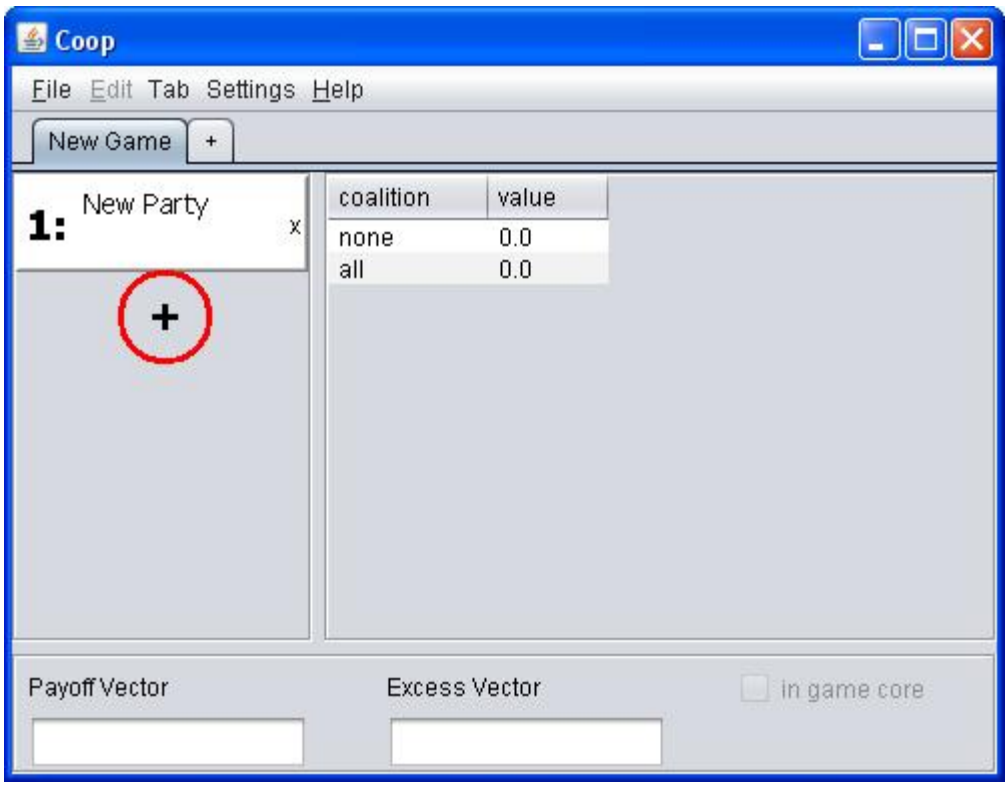
Seznam obrázků

Seznam příloh

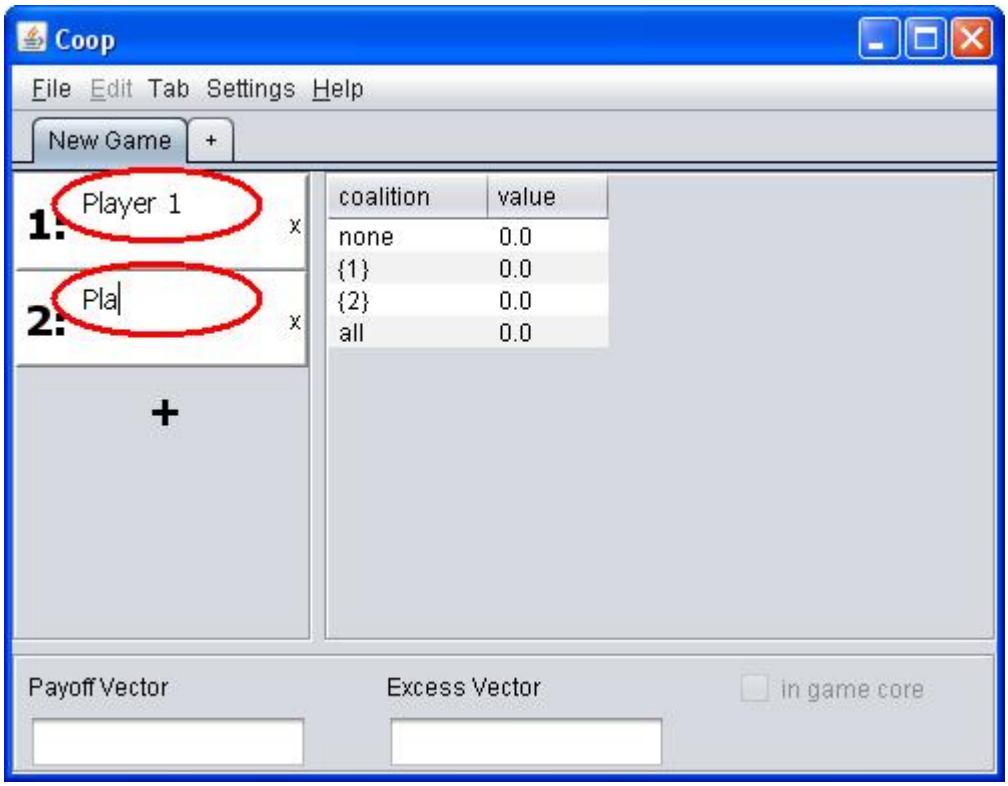
Příloha A

Návod k vytvoření konkrétní hry v uživatelském prostředí programu

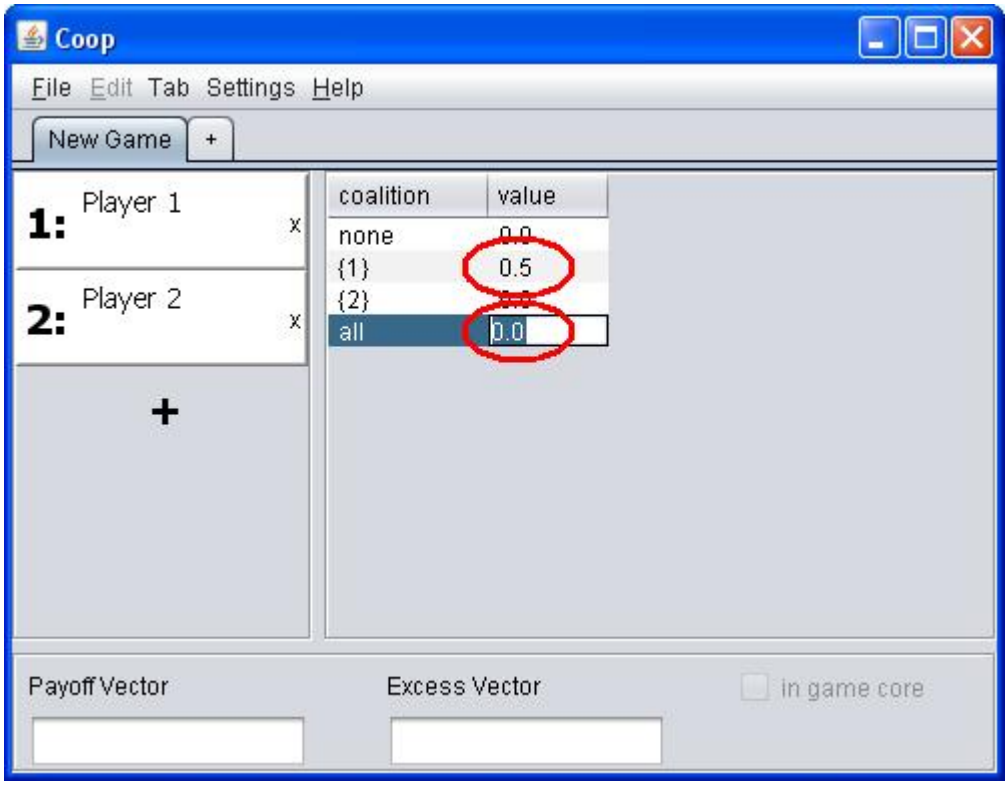
1. Po otevření programu se zobrazí nová hra bez hráčů, prvním krokem bude jejich přidání.



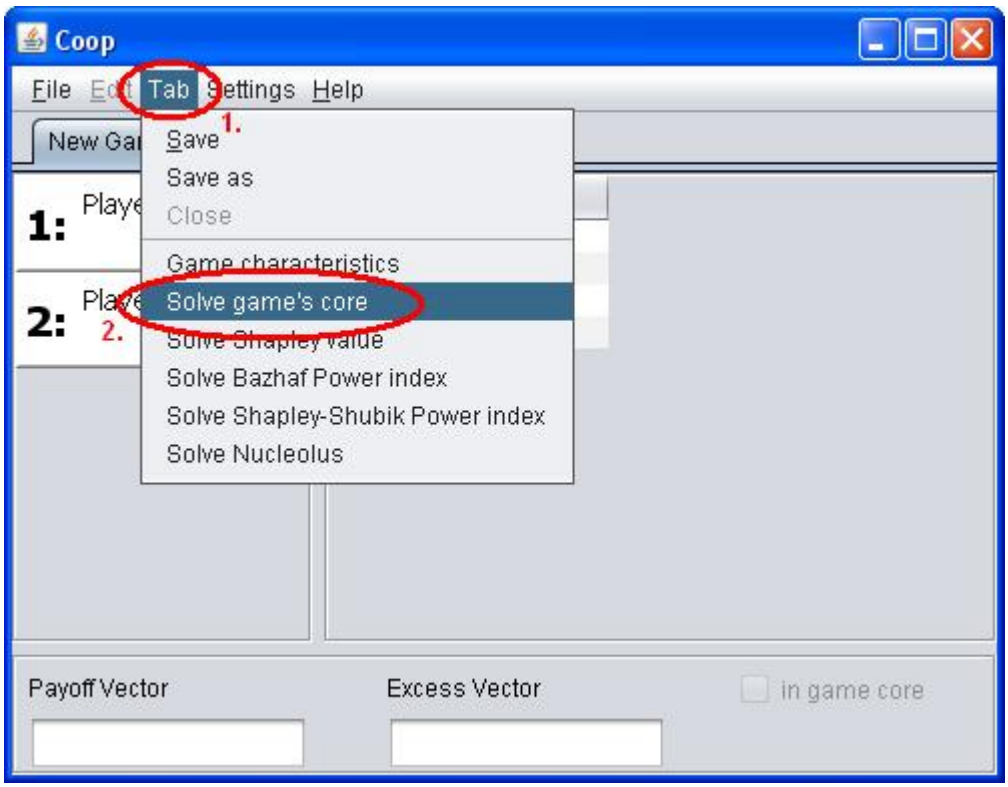
2. V dalším kroku upravíme jména hráčů označením textu „New Party“ a jeho přepsáním.



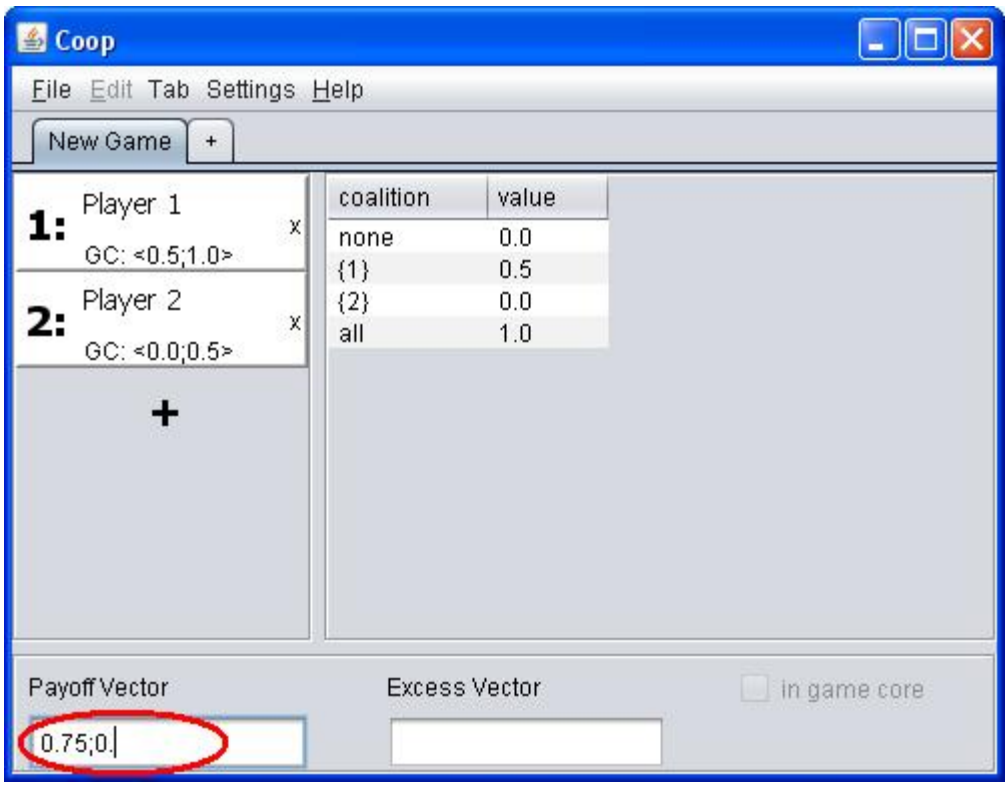
3. Trojklikem na příslušné políčko v tabulce jej označíme a následně upravíme.



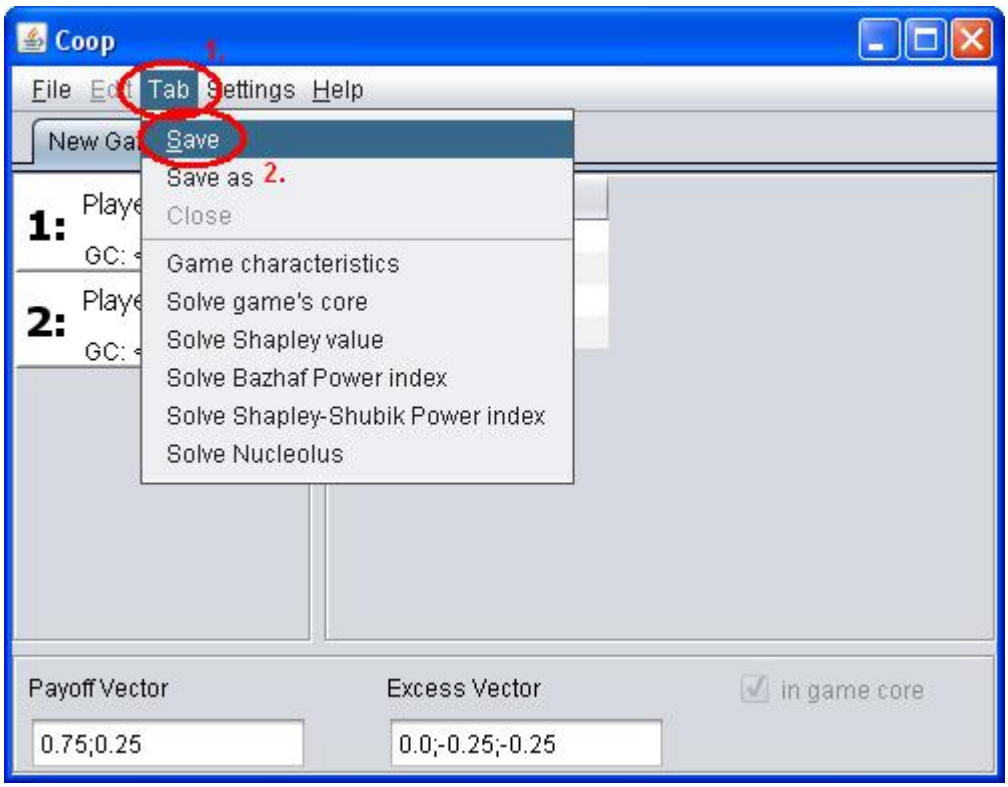
4. Jádro hry vypočteme kliknutím na nabídku Tab a následným kliknutím na položku Solve game's core.



5. Otevřené okno s jádrem hry potvrdíme klikem na OK a do pole Payoff Vector zapíšeme hodnotu 0.75 a 0.25, oddělené středníkem a zapsané s desetinnou tečkou.



6. Po zapsání imputace do pole se ve vedlejším poli objeví vektor excesů (0.0;-0.25;-0.25) a zatrhne se zaškrtávací políčko in game core, které signalizuje, že imputace patří do jádra. Kliknutím na nabídku Tab a následně na položku Save hru uložíme do souboru.



Příloha B

CD s programem, zdrojovými texty a ukázkovými soubory kooperativních her