VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

REDUKCE STRATEGICKÝCH HER NA JEJICH BEST-RESPONSE EKVIVALENTY

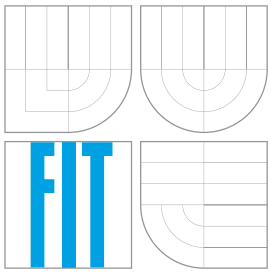
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN GODULA

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

REDUKCE STRATEGICKÝCH HER NA JEJICH BEST-RESPONSE EKVIVALENTY

REDUCTION OF STRATEGIC GAMES TO THEIR BEST-RESPONSE EQUIVALENTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN GODULA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. HRUBÝ MARTIN, Ph.D.

BRNO 2011

Abstrakt

Cieľom tejto práce je navrhnutie a implemetácie knižnice pre redukciu strategických profilov v strategických hrách v normálnej forme. Logika funkčnosti knižnice bude založená na vhodných heuristikách redukcie priestoru hier vychádzajúcich z metód iteratívnej eliminácie dominovaných stratégií a *FDDS*. Funkčnosť výslednej knižnice bude demonštrovaná na vhodne zvolených problémoch.

Abstract

The main goal of this masther thesis is design and implementation of library for reduction of strategy profiles of strategy games in normal form. Logics of library functionality will be based on suitable heuristics founded on methods of iterative elimination of dominated strategies and *FDDS*. Functionality of resultant library will be demonstrated on convenient problems.

Klíčová slova

strategická hra v normálnej forme, Nashovo ekvilibrium, kolerované ekvilibriu, best-response funkcia, dominancia stratégií, redukcia hier, iteratívna eliminácia dominovaných stratégií, *FDDS*

Keywords

normal form strategy game, Nash ekvilibrium, correlated ekvilibrium, best-response function, strategy dominance, game reduction, iterative elimination of dominated strategies, *FDDS*

Citace

Martin Godula: Redukce strategických her na jejich Best-Response ekvivalenty, diplomová práce, Brno, FIT VUT v Brně, 2011

Redukce strategických her na jejich Best-Response ekvivalenty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Hrubého, Ph.D.

.....
Martin Godula
25. mája 2011

Poděkování

Týmto by som chcel poďakovať Ing. Martinovi Hrubému, Ph.D. za poskytnutú pomoc a konzultácie pri tvorbe tejto práce.

Obsah

Kapitola 1

Úvod

Teória hier je matematickou disciplínou, ktorá je často využívaná v širokom spektre vedných odboroch od ekonómie, sociálnej psychológie cez biológiu až po počítačové inžinierstvo pre analýzu rozhodovacích problémov racionálne uvažujúcich jedincov v rôznych strategických situáciách. Teória hier umožňuje modelovanie každodenných ako aj náročných problémov a predikovať ich vývoj alebo správanie pomocou rôznych konceptov riešenia, ako sú napríklad Nashovo alebo korelované ekvilibrium. Tieto metódy však na úspešné vyriešenie určitej hry potrebujú prejsť celým jej stavovým priestorom, čo vzhľadom na komplexnosť a zložitosť mnohých z týchto modelov môže predstavovať značný problém. Aj napriek vysokému výkonu a neustále napredujúcemu vývoju dnešnej výpočtovej techniky, nie je možné riešenie niektorých z týchto úloh konvenčným spôsobom, napríklad z dôvodu nedostatočnej fyzickej pamäti potrebnej na reprezentáciu stavového priestoru skúmanej hry, alebo nereálnosti dopracovania sa k riešeniu v rozumnom čase.

Cieľom tejto diplomovej práce je práve riešenie týchto problémov, a to pomocou redukovania priestoru strategických profilov hier, založeného na metóde iteratívnej eliminácie dominovaných stratégií a metóde *FDDS*. Výsledkom práce by mala byť knižnica na redukciu strategických profilov v strategických hrách v normálnej forme, implementovaná s ohľadom na čo najväčšiu výpočtovú efektívnosť, prenositeľnosť a jej jednoduchú použiteľnosť pre riešenie čo najširšieho spektra problémov matematickej teórie hier.

Následujúca kapitola 2 je venovaná predstaveniu niektorých základných pojmov a celkovému teoretickému uvedeniu do problematiky. Definujeme a vymedzíme kľúčové pojmy ako strategická hra (jej stavebné prvky, rozdelenie a rôzne spôsoby reprezentácie hier), dominancia stratégií, Best-response charakteristiky a ekvivalencia strategických hier. Taktiež budú predstavené koncepty Nashovho a korelovaného ekvilibria a spôsobov ich riešenia, načrtnuté na jednoduchých príkladoch. V závere kapitoly sú predstavené metódy redukcie priestoru strategických profilov hier, ktoré tvoria hlavnú myšlienku pre realizáciu cieľa tejto práce.

Kapitola 3 rozoberá východiská pri samotnom návrhu knižnice. Navrhuje spôsoby uloženia a reprezentácie základných stavebných prvkoch (napríklad strategická hra) prezentovanými v prvej kapitole. Taktiež sa zaoberá problémom spojeným s paralelizáciou výpočtu navrhnutého algoritmu a možnosťami jej implementácie.

V kapitole 4 sú detailne popísané jednotlivé implementované dátové štruktúry, triedy a niektoré zaujímavejšie podúlohy samotného algoritmu.

Predposledná kapitola 5 je venovaná experimentovaniu s implementovaným nástrojom a analýze dosiahnutých výsledkoch. Experimenty boli vykonané na súbore problémov, ktoré zahrňovali niektoré z typických problémov matematickej teórie hier a sériu umelo vygenero-

vaných hier bez akéhokoľvek základu v reálnom svete. Výhodou takto generovaných hier je možnosť parametrizácie viacerých ich vlastnosti, čo je možné využiť pri analýze skúmaného algoritmu.

Záverečná kapitola obsahuje zhrnutie dosiahnutých výsledkov a zamyslenie sa nad možnými rozšíreniami implementovaného nástroja do budúcnosti.

Kapitola 2

Teoretický rozbor

Táto kapitola je venovaná uvedeniu do problematiky matematickej teórie hier a predstaveniu niektorých základných pojmov. Zadefinujeme pojem strategická hra a jej stavebné prvky. Taktiež stručne charakterizujeme rozdelenie strategických hier a spôsobov ich reprezentácie, pričom naša pozornosť bude zameraná najmä na strategické hry v normálnej forme. Stručne načrtnene jednotlivé typické koncepty ich riešenia, ktoré budú demonštrované na jednoduchých ukážkových príkladoch. Záver kapitoly je venovaný metódam redukcie priestoru strategických hier, ktoré sú východiskovým bodom pre túto prácu.

2.1 Strategická hra, jej typy a spôsoby reprezentácie

Strategická hra je modelom interakcie subjektov robiacich rozhodnutia (hráčov). Každý hráč má k dispozícii množinu akcií, nad ktorou robí svoje rozhodnutia. Interakcia hráčov je zapríčinená tým, že každý hráč (jeho zisk/úžitok) je ovplyvňovaný aj akciami ostatných hráčov, nielen jeho vlastnými. Každý hráč má svoje vlastné preferencie nad strategickými profilmi. Strategická hra potom pozostáva z:

- konečnej množiny hráčov,
- množinou rýdzich stratégií pre každého hráča,
- reláciou preferencie nad strategickými profilmi každého hráča.

Hráči sa chovajú racionálne (robia rozumné rozhodnutia za účelom dosiahnutia čo najlepšieho osobného prospechu) a majú úplnú informáciu o pravidlách a štruktúre hry. Tieto informácie ako aj racionalita hráčov je pre nich *common knowledge*, čo znamená, že všetci hráči majú túto informáciu, všetci hráči vedia o tom, že ostatní majú túto informáciu a tak ďalej.

2.1.1 Rozdelenie strategických hier

Strategické hry a formy ich reprezentácie je možné rozčleniť rôznymi spôsobmi na niekoľko typov. Niektoré základné rozdelenia budú stručne načrtnuté v nasledovnej sekcii. Záujmom tejto práce sú výlučne nekooperatívne hry v normálnej forme.

- Kooperatívne a nekooperatívne. V kooperatívnych hrách je snaha hráčov o tvorbu koalícií za účelom získania vyššieho úžitku. Dochádza ku komunikácii medzi hráčmi a musí existovať určitá dôvera voči vymáhaniu dohodnutého chovania.

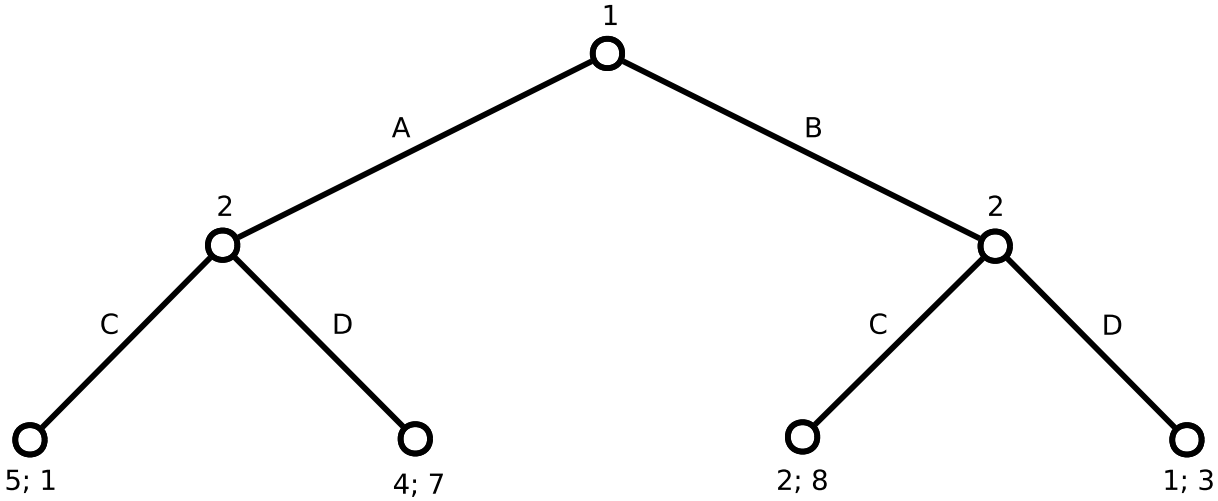
- Simultánne a sekvenčné. V simultánnych hrách vykonávajú hráči voľbu svojich akcií súčasne bez akejkoľvek informácie o voľbe svojich súperov. Naopak, v hrách sekvenčných svoje rozhodnutia robia postupne a majú úplnú alebo aspoň čiastočnú informáciu o voľbách súperov hrajúcich pred nimi.
- Symetrické a asymetrické. V symetrickej hre sú úžitky hráčov závislé iba na zvolených stratégiách bez ohľadu na to, ktorý hráč tieto stratégie hrá. Inak povedané všetci hráči majú k dispozícii rovnaké množiny stratégií a zhodné preferencie nad symetrickými profilmi.

2.1.2 Spôsoby reprezentácie hier

Existujú rôzne spôsoby zápisu hier. V tejto sekcii budú stručne charakterizované najpoužívanejšie z nich.

Hry v extenzívnej forme

Extenzívna forma je typická pre reprezentáciu sekvenčných hier, kde si jednotliví hráči vyberajú svoje stratégie postupne a v momente rozhodnutia majú aspoň čiastočnú informáciu o predchádzajúcej voľbe súpera. Príklad hry v extenzívnej forme je uvedený na obrázku 2.1. Hry sa účastnia dvaja hráči, pričom hráč 1 vykonáva svoju voľbu z dvoch stratégií (A a B) ako prvý. Následne ťahá hráč dva, ktorý má informáciu o voľbe svojho súpera. V listoch stromovej štruktúry sú potom uvedené úžitky pre možné výsledky hry.



Obr. 2.1: Príklad hry v extenzívnej forme

Hry v normálnej forme

Hry v normálnej forme sú najčastejšie reprezentované pomocou matíc zobrazujúcich hráčov, ich stratégie a úžitky. Obecnejšie môže byť hra reprezentovaná aj výplatnou funkciou, ktorá priraďuje úžitok každému hráčovi pre všetky možné kombinácie stratégií (pre všetky strategické profily). Pri zápise hry v normálnej forme sa predpokladá, že hráči si volia svoje

akcie súčasne, alebo aspoň bez akejkoľvek informácie o voľbe súperov. Príklad hry v normálnej forme je načrtnutý v tabuľke 2.1. Špeciálnym prípadom sú hry s nulovým súčtom, v ktorých je súčet úžitkov všetkých hráčov v každom strategickom profile rovný nule (zisk niektorých účastníkov hry je presne vybalancovaný stratou ďalších).

hráč 1 \ hráč 2	D	E	F
A	4, 2	1, 5	4, 3
B	3, 1	0, 4	2, 1
C	7, 2	1, 4	4, 2

Tabuľka 2.1: Príklad hry v normálnej forme.

2.1.3 Nekooperatívne hry v normálnej forme

Každá nekooperatívna hra v normálnej forme sa skladá z nasledujúcich elementov.

1. Konečná množina hráčov $Q = \{1, 2, \ldots, N\}$.
2. Každý hráč i má na výber stratégie s konečnej množiny S_i.
3. Výsledok hry je definovaný strategickým profilom, ktorý sa skladá zo zvolených stratégií jednotlivých hráčov. Množina všetkých strategických profilov je definovaná ako:
$$S = S_1 \times S_2 \times \ldots \times S_N$$
4. Preferencie hráčov nad strategickými profilmi, ktoré môžu byť definované úžitkovou funkciou:
$$u_i : S \ \rightarrow R$$
Všetky informácie o preferenciách hráčov môžu byť reprezentované aj maticovým zápisom.

Strategická hra o N hráčoch je potom $(2N + 1)$-tica:

$$\Gamma = (Q; S_1, S_2, \ldots, S_N; U_1, U_2, \ldots, U_N)$$

alebo skrátene $\Gamma = (Q, S, U)$, kde S značí stavový priestor hry a U úžitkové funkcie hráčov.

Uvažujeme strategický profil $s = (s_1, s_2, \ldots, s_N) \in S$. Potom môžme definovať vybrané stratégie všetkých hráčov okrem hráča i ako $s_{-i} = (s_1, s_2, \ldots, s_{i-+}, s_{i+1}, \ldots, s_N) \in S_{-i}$. Tento vektor je **kontextom** (subprofilom), v ktorom hráč i robí svoje rozhodnutia.

2.2 Dominancia stratégií

V strategických hrách v normálnej forme je bežným javom dominancia stratégií, pričom rozlišujeme dva jej typy:

1. $s_i^{'}$ hráča i striktne dominuje stratégiu $s_i^{''}$ práve vtedy, keď:
$$\forall s_{-i} \in S_{-i} : u_i(s_i^{'}, s_{-i}) > u_i(s_i^{''}, s_{-i})$$

2. $s_i^{'}$ hráča slabo dominuje i stratégiu $s_i^{''}$ práve vtedy, keď:

$$\forall s_{-i} \in S_{-i} : u_i(s_i^{'}, s_{-i}) \geq u_i(s_i^{''}, s_{-i}) \wedge \exists s_{-i} \in S_{-i} : u_i(s_i^{'}, s_{-i}) > u_i(s_i^{''}, s_{-i})$$

V akejkoľvek strategickej hre je pre hráča výhodnejšie hrať striktne dominantnú stratégiu bez ohľadu na stratégie, ktoré volia jeho súperi. Slabo dominujúca akcia je vo všetkých kontextoch rozhodovania hráča vždy aspoň rovnako dobrá, a zároveň aspoň v jednom kontexte lepšia ako akcia, ktorú dominuje.

Stratégia s^* hráča i sa nazýva striktne dominantnou, ak je táto stratégia striktnou best-response na všetky jeho kontexty s_{-i}, ktoré si môžu ostatní hráči zvoliť [9]:

$$\forall s_{-i} \in S_{-i}, \forall s_i^{'} \neq s_i^* : U_i(s_i^*, s_{-i}) > U_i(s_i^{'}, s_{-i})$$

Následne je možné definovať ekvilibrium dominantných stratégií, ktoré pozostáva zo striktne dominantných stratégií všetkých hráčov.

Best-response funkcia

Majme hru $\Gamma = (Q, S, U)$. Stratégia s_i^* hráča i je best-response na jeho kontext rozhodovania s_{-i} práve vtedy, keď [7]:

$$\forall s_i \in S_i : u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$$

Čiže stratégia s_i^* hráča i je best-response na určitý kontext rozhodovania s_{-i} vtedy, keď dáva hráčovi i najvyšší možný úžitok. Ak nájdeme všetky best-response stratégie na kontexty $s_{-i} \in S_{-i}$, dostaneme best-response funkciu hráča i:

$$BR_i(s_{-i}) = \{s_i \in S_i : \forall s_i^{'} \in S_i : u_i(s_i, s_{-i}) \geq u_i(s_i^{'}, s_{-i})\}$$

Na určitý kontext $s_{-i} \in S_{-i}$ môže existovať aj viac best-response stratégií hráča i.

2.3 Ekvilibrium a jeho formy

Cieľom skúmania strategických hier a experimentovania s nimi je obyčajne nájdenie určitého riešenia, ktoré zodpovedá strategickému profilu (množine strategických profilov), ku ktorému budú mať racionálne uvažujúci hráči tendenciu inklinovať. Takýto strategický profil je nazývaný ekvilibriom. Existuje niekoľko rôznych konceptov ekvilibrií v strategických hrách v normálnej forme.

2.3.1 Nashovo ekvilibrium

Keďže strategické hry zriedkakedy obsahujú ekvilibrium dominantných stratégií, bolo nutné nájdenie nového akceptovatelného konceptu riešenia hier, kde bude zachytené racionálne správanie sa hráčov a ich snaha o maximalizáciu vlastného úžitku. Túto myšlienku spĺňa práve Nashovo ekvilibrium, ktoré sa stalo hlavným konceptom riešenia nekooperatívnych hier v normálnej forme s nenulovým súčtom. Nashovo ekvilibrium je stabilným riešením, kde žiadny z hráčov nemôže dosiahnuť zlepšenie svojho zisku odklonením sa od neho.

Vektor stratégií $s^* \in S$ je rýdzim Nashovým ekvilibriom vtedy, keď pre každého hráča i a každú jeho alternatívnu stratégiu $s_i \in S_i$ platí:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*)$$

Inak povedané žiadny hráč i nemôže zlepšiť svoj úžitok zmenou s_i^* na stratégiu s_i za predpokladu, že všetci ostatní hráči ostanú pri svojich pôvodných stratégiách zvolených v s^*. Takýto strategický profil je seba-presadzujúci v zmysle, že pokiaľ hráči tento strategický profil dosiahnu, je v najlepšom záujme každého z nich v ňom zotrvať [6].

Nashové ekvilibrium je možné definovať aj ako prieniky best-response funkcií jednotlivých hráčov. Strategický profil $s = (s_1, s_2, \ldots, s_{|Q|})$ je Nashovým ekvilibriom práve vtedy, ak pre stratégiu s_i každého hráča i platí:

$$s_i \in BR_i(s_{-i})$$

Ekvilibrium dominantných stratégií je vždy aj Nashovým ekvilibriom, navyše v prípade striktne dominantného riešenia je ekvilibrium aj unikátné. Na druhej strane, Nashovo ekvilibrium unikátnym byť nemusí, čoho príkladom môžu byť koordinačné hry, ktoré obsahujú vždy viacero Nashových ekvilibrií. Hráči môžu nadobúdať veľmi rozdielne úžitky v jednotlivých týchto ekvilibriach, v dôsledku čoho môže byť kvalita alebo výhodnosť ekvilibrií výrazne odlišná.

Strategická hra nutne nemusí obsahovať žiadne rýdze Nashovo ekvilibrium. V takomto prípade je možné dosiahnutie stabilného riešenia výberom viacerých stratégií aspoň jedného hráča, pričom každá z nich bude hraná s určitou pravdepodobnosťou. Každý hráč si zvolí pravdepodobnostné rozdelenie nad množinou jeho prípustných stratégií, takáto voľba je nazývaná zmiešanými stratégiami. John Nash dokázal, že akákoľvek hra s konečným počtom hráčov a konečnou množinou stratégií má Nashovo ekvilibrium v zmiešaných stratégiách [5].

Majme množinu Δ_i všetkých zmiešaných stratégií hráča i. Potom zmiešanú stratégiu $\sigma \in \Delta_i$ hráča i môžme definovať ako vektor reálnych čísel $\sigma = (\sigma(s_1), \sigma(s_2), \ldots, \sigma(s_{|S_i|}))$, kde

$$\sum_{k=1}^{|S_i|} \sigma(s_k) = 1 \wedge \forall j \in \{1, 2, \ldots, |S_i|\} : 0 \leq \sigma(s_j) \leq 1\}$$

Očakávaný úžitok π_i hráča i v strategickom profile zmiešaných stratégií definujeme ako:

$$\pi_i(\sigma_i, \sigma_{-i}) = \sum_{s_{-i} \in S_{-i}} \sum_{s_i \in S_i} \sigma_i(s_i)\sigma_{-i}(s_{-i})u_i(s_i, s_{-i})$$

Následne môžme Nashovo ekvilibrium zmiešaných stratégií definovať ako strategický profil $(\sigma_1, \sigma_2, \ldots, \sigma_{|Q|})$ taký, že pre každého hráča $i \in Q$ platí [1]:

$$\pi_i(\sigma_1, \sigma_2, \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i-1}, \ldots, \sigma_{|Q|}) \geq \pi_i(\sigma_1, \sigma_2, \ldots, \sigma_{i-1}, q, \sigma_{i-1}, \ldots, \sigma_{|Q|})$$

pre všetky zmiešané stratégie $q \in \Delta_i$.

Z pohľadu smerovania tejto práce sa budeme zameriavať hlavne na rýdze Nashovo ekvilibrium, pričom zaujímavou je najmä jeho definícia prostredníctvom prieniku best-response stratégií hráčov. Práve na nej je založená neskôr prezentovaná a implementovaná metóda *FDDS*.

2.3.2 Výpočet Nashovho ekvilibria

Hry s rýdzim Nashovým ekvilibriom je možné riešiť pomocou best-response funkcií ako bolo definované v 2.3.1. Pre hry s existenciou iba jedného rýdzeho Nashovo ekvilibria je možné použitie iteratívnej eliminácie striktne dominovaných stratégií, ktorá je popísaná neskôr (2.5.1).

Jedným zo spôsobom nájdenia Nashových ekvilibrií zmiešaných stratégií je použitie algoritmu vymenovania supportov [6]. Pre túto metódu je nutné si zadefinovať *support* zmiešanej stratégie σ - množina všetkých stratégií, ktorým zmiešaná stratégia σ priraďuje nenulovú pravdepodobnosť. Pre hráča i hry $\Gamma = (Q, S, U)$ a jeho zmiešanú stratégiu σ_i:

$$support(\sigma_i) = \{s_i \in S_i \mid \sigma_i(s_i) > 0\}$$

Algoritmus vymenovania supportov pre hru dvoch hráčov [6] je možné popísať nasledovnými krokmi.

- Zvolíme $K = \{1, 2, \ldots, min(S_1, S_2)\}$.
- Pre každé $k \in K$ a pre všetky páry (I, J) k-prvkovej podmnožiny množín rýdzich stratégií S_1 respektíve S_2 riešime rovnice:

$$\begin{aligned}
\sum_{i \in I} x_i u_i(i, j) &= v; \forall j \in J \\
\sum_{i \in I} x_i &= 1 \\
\sum_{j \in J} y_j u_i(i, j) &= u; \forall i \in I \\
\sum_{j \in J} y_j &= 1
\end{aligned}$$

pričom pre všetky x_i a y_i musí platiť $x_i \geq 0$ a $y_i \geq 0$.

	C	D
A	4, 4	1, 5
B	5, 1	0, 0

Tabuľka 2.2: Game of chicken [6]

Máme hru v tabuľke 2.2. Riešením algoritmu pre $k = 1$ dostaneme dve rýdze Nashovo ekvilibria $(B, C), (A, D)$. Ďalej zostrojíme rovnice pre $k = 2$, ktorých riešením dostaneme ekvilibrium zmiešaných stratégií $((1/2, 1/2), (1/2, 1/2))$.

$$\begin{aligned}
4x_1 + x_2 &= v \\
5x_1 &= v \\
4y_1 + y_2 &= u \\
5y_1 &= u \\
x_1 + x_2 &= 1 \\
y_1 + y_2 &= 1
\end{aligned}$$

2.3.3 Korelované ekvilibrium

Je ďalším konceptom riešenia strategických hier, navrhnuté matematikom Robertom Aumannom. V porovnaní s Nashovým ekvilibriom sa jedná o všeobecnejší koncept, pričom zároveň platí, že každé Nashovo ekvilibrium je zároveň aj ekvilibriom korelovaným (naopak to neplatí). Korelované ekvilibrium predpokladá existenciu určitého koordinačného subjektu, ktorý hráčom zasiela signál, na základe ktorého potom vykonávajú výber akcie. Korelované ekvilibrium je pravdepodobnostné rozdelenie nad strategickými profilmi $s \in \times_i S_i$. Pravdepodobnosť strategického profilu označíme ako $p(s)$, alebo $p(s) = p(s_i, s_{-i})$ v kontexte rozhodovania jedného hráča. Rozdelenie pravdepodobností nad strategickými profilmi je korelovaným ekvilibriom vtedy, keď pre všetkých hráčov i a všetky stratégie $s_i, s_i^{'} \in S_i$ platí [6]:

$$\sum_{s_{-i}} p(s_i, s_{-i}) u_i(s_i, s_{-i}) \geq \sum_{s_{-i}} p(s_i, s_{-i}) u_i(s_i^{'}, s_{-i})$$

Inak povedané, ak hráč i obdrží navrhovanú stratégiu s_i, jeho očakávaný úžitok nemôže byť zvýšený zmenou tejto stratégie na žiadnu inú stratégiu $s_i^{'} \in S_i$. Nashové ekvilibria určitej hry sú podmnožinou ekvilibrií korelovaných, navyše existencia dôveryhodného synchronizačného mechanizmu umožňuje dosiahnutie vyššieho úžitku pre jednotlivých hráčov.

2.3.4 Výpočet korelovaného ekvilibria

Výhodou korelovaného ekvilibria v porovnaní s Nashovým je menšia zložitosť jeho výpočtu, ktorý si vyžaduje riešenie úlohy lineárneho programovania s cieľom maximalizácie spoločného úžitku všetkých hráčov za dodržania podmienok vyplývajúcich z kontextu hry. Výpočet korelovaného ekvilibria má polynomickú časovú zložitosť, zatiaľ čo výpočet Nashovho ekvilibria je v zložitostnej triede $PPAD$ [8].

	C	D
A	8, 3	1, 2
B	1, 2	3, 4

Tabuľka 2.3: Hra pre zostavenie G-matice

	ac	ad	bc	bd
$a \to b$	7	-2		
$b \to a$			-7	2
$c \to d$	1		-2	
$d \to c$		-1		2

Tabuľka 2.4: G-matica zostrojená pre hru 2.3

Efektívnym spôsobom výpočtu korelovaného ekvilibria je výpočet pomocou G-matice [2] (implementovaný nástroj CE-Solver [10]). V tabuľke 2.3 je zobrazená hra, pre ktorú je zostavená G-matica v tabuľke 2.4. Na základe G-matice je potom možné formulovať nasledovný LP problém, na ktorého riešenie je dostupných mnoho nástrojov, napríklad GLPK solver.

$$
\begin{aligned}
MAXIMISE : Z &= 11p_1 + 3p_2 + 3p_3 + 7p_4 \\
p_{1\ldots4} &\in \langle 0, 1\rangle \\
\sum_{i=1}^{4} p_i &= 1 \\
7p_1 - 2p_2 &\geq 0 \\
-7p_3 + 2p_4 &\geq 0 \\
p_1 - 2p_3 &\geq 0 \\
-p_2 + 2p_4 &\geq 0
\end{aligned}
$$

2.4 Ekvivalencia strategických hier

Dve strategické hry s rovnakou množinou hráčov s rovnakými množinami stratégií sú plne ekvivalentné, ak z úžitkovej funkcie v jednej hre vyplývajú rovnaké preferencie hráčov ako v hre druhej [3]. Najvšeobecnejším spôsobom ako popísať správanie hráčov v hre je pomocou pravdepodobnostného rozloženia nad množinou strategických profilov $S = \times_{i \in Q} S_i$. Pre určitú konečnú množinu Z môžme označiť všetky takéto rozdelenia nad touto množinou ako $\Delta(Z)$. Potom $\Delta(S)$ značí množinu všetkých pravdepodobnostných rozdelení nad strategickými profilmi hráčov v určitej hre Γ. V hre Γ s úžitkovou funkciou u by hráč i preferoval správanie hráčov podľa pravdepodobnostného rozloženia $\mu = \mu(s)_{s \in S}, \mu \in \Delta(S)$ nad rozložením λ, ak mu toto rozloženie μ dáva vyšší očakávaný zisk ako rozloženie λ:

$$\sum_{s \in S} \mu(s) u_i(s) \geq \sum_{s \in S} \lambda(s) u_i(s)$$

V hre Γ' by hráč i preferoval μ pred λ ak:

$$\sum_{s \in S} \mu(s) u_i'(s) \geq \sum_{s \in S} \lambda(s) u_i'(s)$$

Na základe týchto definícii možno povedať, že dve strategické hry Γ a Γ' sú ekvivalentné práve vtedy, keď pre všetkých hráčov $i \in Q$ platí, že hráč i by preferoval μ pred λ v hre Γ, iba ak by preferoval μ pred λ v hre Γ'.

2.4.1 Best-response ekvivalencia

Best-response ekvivalencia [2, 3] je ďalšou možnou definíciou ekvivalencie strategických hier. Jedná sa o menej striktnú definíciu a tým pádom ju splňuje väčšie množstvo hier. Navyše je táto definícia zaujímavá aj z pohľadu zamerania tejto práce. Hra $\Gamma = (Q, S_r, U_r)$ je best-response ekvivalentná inej hre $\Gamma = (Q, S_r, U_r)$ práve vtedy, keď platí:

1. $S_r \subseteq S$
2. $\forall s_r \in S_r, \forall i \in Q : U_{r,i}(s_r) = U_i(s_r)$
3. $\forall i \in Q : \cup_{s_{-i} \in S_{-i}} BR_i(s_{-i}, \Gamma) = \cup_{s_{r,-i} \in S_{r,-i}} BR_i(s_{r,-i}, \Gamma_r)$

Inak povedané, strategická hra Γ' je best-response ekvivalentná hre Γ práve vtedy, keď je hraná rovnakou množinou hráčov, akurát jej strategický priestor S_r je podmnožinou S, pričom žiadna stratégia z množiny best-response stratégií nebola vynechaná.

Vzhľadom na racionalitu hráčov je možné použiť aj ďalšiu definíciu best-response ekvivalencie, na ktorú je nutné zavedenie pojmu **užitočná best-response stratégia**. Stratégiu b_i hráča i je možné považovať za užitočnú best-response stratégiu, ak platí:

$$\exists s_{-i} \in S_{-i} : b_i \in BR_i(s_{-i}) \wedge \exists b_j \in S_j : b_j \in BR_j(s_{-i,-j}, b_i)$$

Ekvivalencia vzhľadom na užitočné best-response stratégie umožňuje opomenúť stratégie, ktoré síce patria do množiny BR, ale kontexty, v ktorých sa prejavujú nebudú nikdy hrané racionálne uvažujúcimi hráčmi. Preto vynechanie takýchto stratégií neovplyvní správanie hráčov.

2.5 Redukcia strategických hier

V dôsledku racionality hráčov, je možné predpokladať, že takýto hráč nebude mať nikdy záujem o použitie dominovanej stratégie. Takáto stratégia sa potom stáva zbytočnou a jej odstránením je možné zmenšiť strategický priestor hry. V tejto sekcii budú predstavené dve metódy redukcie strategického priestoru hier, z ktorých sa bude vychádzať pri návrhu knižnice.

2.5.1 Iteratívna eliminácia dominovaných stratégií

Proces iteratívnej eliminácie striktne dominovaných stratégií prebieha následovne [4]:

- Krok 0: pre každého hráča i zvolíme $S_i^0 = S_i$
- Krok 1: pre každého hráča i zvolíme S_i^1

$$S_i^1 = \{s_i \in S_i^0 \mid \neg\exists\, s_i^{'} \in S_i^0 : u_i(s_i^{'}, s_{-i}) > u_i(s_i, s_{-i}) \forall s_{-i} \in S_{-i}^0\}$$

- Krok $k+1$: pre každého hráča i zvolíme S_i^{k+1}

$$S_i^{k+1} = \{s_i \in S_i^k \mid \neg\exists\, s_i^{'} \in S_i^k : u_i(s_i^{'}, s_{-i}) > u_i(s_i, s_{-i}) \forall s_{-i} \in S_{-i}^k\}$$

- Krok $\propto$: pre každého hráča i zvolíme $S^{\propto}$

$$S^{\propto} = \cap_{k=1}^{\propto} S_i^k$$

  Množiny stratégií hráčov hry v normálnej forme sú konečné, preto musí algoritmus skončiť po konečnom počte krokov, keďže množiny stratégií sa môžu len zmenšovať.

Hra je riešiteľná pomocou metódy eliminácie striktne dominovaných stratégií ak $S^{\propto}$ obsahuje iba jeden strategický profil [4]. Väčšina hier nie je pomocou tejto metódy riešiteľná, ale sú redukovateľné na ekvivalentné hry s menším stavovým priestorom. Pri tejto metóde nezáleží na poradí odstraňovania dominovaných stratégií, keďže tieto stratégie ostanú dominovanými aj v ďalších krokoch a budú odstránené neskôr. Toto však neplatí pre analogickú metódu iteratívnej eliminácie slabo dominovaných stratégií, čo je zachytené na príklade hry 2.5. V tejto hre je možné v prvom kroku eliminovať stratégiu C alebo E. V prvom prípade bude výsledným profilom (A, D), zatiaľ čo v prípade druhom (B, D). V dôsledku toho môže viesť redukcia určitej hry pomocou eliminácie slabo dominovaných stratégií k rôznym výsledkom.

	C	D	E
A	0, 0	1, 3	1, 3
B	1, 1	1, 1	0, 0

Tabuľka 2.5: Eliminácia slabo dominovaných stratégií

Príklad aplikovania iteratívnej eliminácie striktne dominovaných stratégií

V tabuľke 2.6 je zachytená činnosť algoritmu po jednotlivých krokoch. V nultom kroku položíme $S_1^0 = S_1 = \{A, B\}$ a $S_2^0 = S_2 = \{C, D, E\}$. Stratégia D hráča 2 je striktne dominovaná stratégiou C, a preto racionálne uvažujúci hráč túto stratégiu hrať nikdy nebude. Túto stratégiu je následne možné vylúčiť. Obdobným spôsobom pokračujeme v ďalších krokoch elimináciou stratégie A hráča 1 a stratégie E hráča 2. Keďže množina strategických profilov výslednej hry obsahuje len jeden prvok, bola pôvodná hra riešiteľná metódou iteratívnej eliminácie striktne dominovaných stratégií a jediný strategický profil výslednej hry je Nashovým ekvilibriom.

Krok 0				Krok 1		
	C	D	E		C	E
A	0, 3	1, 1	0, 4	A	0, 3	0, 4
B	1, 4	3, 1	2, 0	B	1, 4	2, 0

Krok 2			Krok 3	
	C	E		C
B	1, 4	2, 0	B	1, 4

Tabuľka 2.6: Aplikácia metódy iteratívnej eliminácie striktne dominovaných stratégií

2.5.2 Rýchla detekcia dominovaných stratégií (*FDDS*)

Metóda *FDDS* [2] je založená na analýze dominantných stratégií. Na vstupe očakáva hru s veľmi rozsiahlym stavovým priestorom, v ktorom je veľmi pravdepodobná existencia dominovaných stratégií, na základe čoho môže byť hra značne zredukovaná. Počas chodu algoritmu je konštruovaná stromová štruktúra, nazývaná graf dosiahnuteľných profilov (*Graph of reachable profiles*, ďalej *GRP*), v ktorom sú zachytené všetky významné stratégie jednotlivých hráčov. Metóda umožňuje analýzou tejto štruktúry niekoľko rôznych foriem výstupu zahrňujúcich napríklad rýchlu detekciu dominovaných stratégií, rýdze Nashovo ekvilibrium, detekcia určitých cyklov reprezentujúcich zmiešanie chovanie (zmiešané Nashovo ekvilibrium).

Graf dosiahnuteľných profilov určitej hry $\Gamma = (Q, S, U)$ je štruktúra $GRP = [V, E]$, kde [2]:

- V je konečná množina uzlov (s, Q_a, Q_b), kde $s \in S$; $Q_a, Q_b \subseteq Q$; $Q_a \cap Q_b = \emptyset$. Q_a je podmnožina hráčov hry, ktorí súhlasia s daným profilom, zatiaľ čo Q_r je podmnožina hráčov hry, ktorí s nim nesúhlasia. Iba hráči, pre ktorých platí $s_i \in BR_i(s_{-i})$, súhlasia s profilom s_i.

- $E \subseteq V \times V \times Q$ je množina hrán. Hrany sú potrebné iba na analýzu topológie stromu a existujú iba pre nesúhlasiacich hráčov, čiže $\forall(v_1, v_2, i) \in E, v_1 = (s, Q_a, Q_r,) : i \in Q_r$. Hrana zobrazuje tendenciu hráča i odkloniť sa od profilu reprezentovaným vrcholom v_1 do profilu reprezentovaným vrcholom v_2.

Algorithm 1 vloženie nových uzlov do GRP (B, v, GRP) [2]

```
for all b ∈ B do
  if s_i = b then
    Q_a ← Q_a ∪ {i}
  else
    Q_r ← Q_r ∪ {i}
    if ∃v' ← (s', Q'_a, Q'_r) ∈ V : s' = (b, s_{-i}) then
      Q'_a ← Q'_a ∪ {i}
    else
      v' ← ((b, s_{-i}), {i}, ∅)
      V ← V ∪ {v'}
    end if
    E ← E ∪ {(v, v', i)}
  end if
end for
```

Hlavný algoritmus konštrukcie GRP [2]:

1. Inicializácia $GRP = [V, \emptyset]$ náhodne vygenerovanými uzlami. Ak $S_{rand} \subset S$ je množina náhodných profilov, potom počiatočná množina uzlov je $V_0 = \{(s, \emptyset, \emptyset) \mid s \in S_{rand}\}$.

2. Náhodne je zvolený uzol $v = (s, Q_a, Q_r) \in V$, ktorý nie je vyriešený $(Q_a \cup Q_r \neq Q)$. Ak taký uzol neexistuje, algoritmus je ukončený.

3. Z množiny $Q \backslash (Q_a \cup Q_r)$ je náhodne zvolený hráč i, pre ktorého je vypočítaná množina best-response stratégií na kontext s_{-i}: $B = BR_i(s_{-i})$. Tento výpočet vyžaduje vyvolanie úžitkovej funkcie u_i pre všetky $s \in \{(s_i, s_{-i}) \mid s_i \in S_i\}$, čo znamená $|S_i|$ krát.

4. Vloženie všetkých $b \in B$ hráča i v strategickom profile v do GRP pomocou algoritmu 1.

5. Opakovanie kroku 2.

Algoritmus je ukončený v prípade vyriešenia všetkých uzlov v GRP, alebo ak je počet uzlov grafu rovný veľkosti stavového priestoru hry. Na základe toho je garantované ukončenie algoritmu. V reálnych situáciách v prípade hier s extrémne veľkým strategickým priestorom tomu tak byť nemusí z dôvodu obmedzenej kapacity fyzických pamätí počítačov.

Praktické experimenty poukazujú na vhodnosť zavedenia explicitných podmienok ukončenia algoritmu ako napríklad [2]:

- $|V|$ prekročí určitý stanovený limit,
- vek GRP (počet iterácií hlavného algoritmu bez vloženia nového uzlu) prekročí určitý stanovený limit.

Výkonnosť algoritmu ako aj kvalita jeho výstupu je úzko spojená s rozlíšiteľnosťou stratégií jednotlivých hráčov[2]. Stavový priestor je možné považovať za **dobre rozlíšiteľný** ak:

$$\forall i \in Q, \forall s_{-i} \in S_i : |BR_i(s_{-i})| = 1$$

V prípade dobrej rozlíšiteľnosti strategického priestoru hráč i vie presne určiť pre každý svoj kontext $s_{-i} \in S_{-i}$ stratégiu, do ktorej sa má presunúť. Opačný prípad má za následok rozsiahlejšie vetvenie v strategickom priestore, čo zapríčiňuje pomalšiu konvergenciu algoritmu.

Z popisu algoritmu *FDDS* je patrné, že jeho názov nebol zvolený najšťastnejším spôsobom, keďže jeho fungovanie je jednoznačne založené na hľadaní best-response stratégií a nie stratégií dominovaných.

Kapitola 3

Návrh riešenia problému

V tejto kapitole sa budeme venovať analýze požiadaviek pre návrh základných dátových štruktúr. Ďalej je rozobraná paralelizácia výpočtu, a to najmä z hľadiska prístupu ku zdieľaným dátam a predstavenia nástrojov umožňujúcich jednoduchú implementáciu paralelných programov. V závere sa snažíme identifikovať niektoré komplexnejšie podúlohy samotného algoritmu *FDDS* a problémy, ktoré by pri ich riešení mohli nastať.

3.1 Reprezentácia modelu hry

Model hry v normálnej forme musí poskytovať nasledujúce informácie o hre:

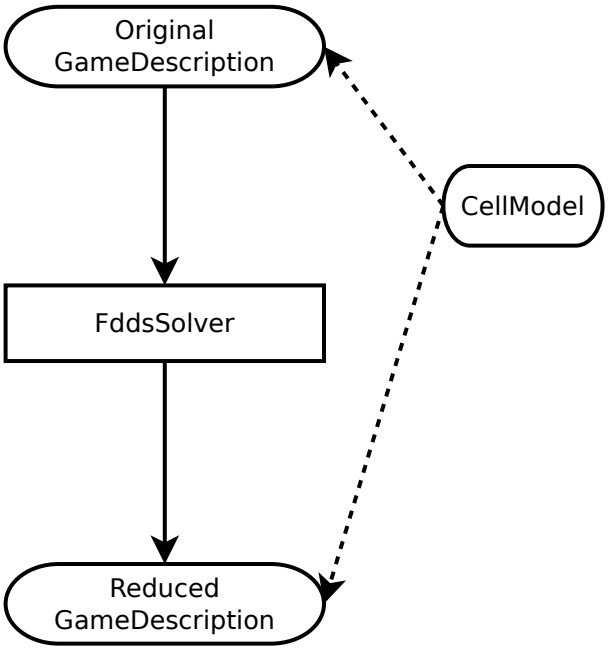
- počet hráčov,
- stratégie jednotlivých hráčov,
- preferencie hráčov nad strategickými profilmi / úžitková funkcia.

Zatiaľ čo implementácia prvých bodov je viacmenej priamočiara, komplikácie nastávajú pri preferenciách hráčov. Strategické hry sú typicky reprezentované zápisom strategického priestoru do maticových štruktúr. Tento prístup je prakticky nepoužiteľný pre veľmi rozsiahle hry z dôvodu obmedzenej kapacity pamäte. Rozumným spôsobom reprezentácie úžitkov hráčov je takzvaný *cellModel* [2]. Jedná sa o funkciu, ktorá má na vstupe nejaký strategický profil, a na výstupe vektor úžitkov jednotlivých hráčov v tomto profile. Vyvolanie funkcie *cellModel* v sebe veľmi často zahrňuje simuláciu alebo experimentovanie so skúmaným modelom v situácii špecifikovanej strategickým profilom. Z tohoto dôvodu je nutné sa pokúsiť čo najviac minimalizovať počet vyvolaní funkcie *cellModel*.

Pri použití tohoto prístupu je možné reprezentáciu hry rozdeliť na dve časti.

1. Popis strategického priestoru hry (hráči a ich stratégie) - *GameDescription*.
2. Definícia úžitkov hráčov - *CellModel*.

Takýto spôsob reprezentácie hry je možno považovať za výhodný pre účel redukcie stavového priestoru hry. Navrhovaný nástroj *FddsSolver* by na svojom vstupe očakával dva vyššie spomínané objekty. Výstupom by mal byť redukovaný stavový priestor definovaný novým objektom *GameDescription*, pričom objekt *CellModel* by ostal použiteľný pre definíciu úžitkov v strategických profiloch hráčov ako hry pôvodnej, tak aj redukovanej. Tieto vzťahy ako aj samotný proces redukcie hry je zachytený na nasledujúcom obrázku 3.1.



Obr. 3.1: Proces redukcie z pohľadu modelu redukovanej hry

3.2 Reprezentácia štruktúry *GRP*

Ako už bolo naznačené, *GRP* je tvorený množinami uzlov a hrán. S pohľadu redukcie hry sú zaujímavé iba uzly dosiahnuteľných profilov, ktoré obsahujú dostatočnú informáciu potrebnú na zredukovanie hry (ak určitý hráč súhlasí s nejakým uzlov *GRP*, potom jeho zodpovedajúca stratégia musí byť prvkom množiny stratégií tohoto hráča v zredukovanej hre). Z tohoto dôvodu nie je nutné sa zaoberať návrhom štruktúr reprezentujúcich hrany *GRP*.

Uzly *GRP* by mohli byť nápomocné aj pri znížení zaťaženia najslabšieho miesta algoritmu, ktorým je prístup ku cache vypočítaných úžitkov respektíve exekúcie funkcie *CellModel* pri výpočte best-response stratégií. Uvažujeme hráča i, kontext jeho rozhodovania s_{-i}, pre ktorý chceme vypočítať best-response stratégie hráča i. Pred samotným týmto výpočtom je možné nahliadnuť do *GRP*. Ak sa v *GRP* nachádza uzol $((s_i, s_{-i}), Q_a, Q_r) \mid i \in Q_a \wedge s_i \in S_i$, potom vieme okamžite určiť hľadanú best-response stratégiu, keďže hráč i sa bude chcieť v danom kontexte s_{-i} vždy presunúť do profilu, s ktorým súhlasí. Pri tomto prístupe by navyše nebolo nutné uchovávať úžitkové vektory pre vyriešené uzly *GRP*.

3.3 Paralelizácia výpočtu

Algoritmus *FDDS* popísaný v 2.5.2 je pripravený na paralelnú implementáciu, konkrétne kroky 2 až 5 zodpovedajúce riešeniu jedného strategického profilu. Z pohľadu zdieľaných dát, bude nutné synchronizovať prístup k nasledovným dátovým štruktúram.

- Fronty uzlov:
  - čakajúcich na vyriešenie,

  - vyriešených,
  - pozastavených.

- Hašovacia tabuľka uzlov - z dôvodu rýchleho prístupu k nim (namiesto prehľadávania front).
- Hašovacia tabuľka úžitkov

OpenMP

Na implementáciu paralelizmus algoritmu je možné použiť niektorý z voľne dostupných nástrojov/knižníc ako napríklad *OpenMP*. Jedná sa o súbor direktív pre prekladač a knižničných procedúr pre paralelné programovanie. Poskytuje jednoduché a flexibilné rozhranie pre vývoj paralelných aplikácií v jazykoch C, C++ a Fortran na rôznych architektúrach. Ukážka kódu 3.1 demonštruje prácu s *OpenMP* na jednoduchom príklade. Použitím pragmy *omp parallel num_threads(4)* vytvorí hlavné vlákno (master thread) ďalšie tri, ktoré budú spolu s ním paralelne vykonávať príslušný blok kódu. Pomocou funkcie *omp_get_thread_num()* je možné získať identifikátori jednotlivých vlákien, pričom pre hlavné vlákno je jeho hodnota rovná nule.

Listing 3.1: Ukážka práce so systémom OpenMP

```
int main()
{
    #pragma omp parallel num_threads(4)
    {
        int id = omp_get_thread_num();
        printf("thread %d\n", id);
    }
    return 0;
}
```

Okrem vytvárania paralelných regiónov kódu poskytuje systém *OpenMP* aj súbory klauzúl pre synchronizáciu vlákien a riadenie prístupu k zdielaným dátam. Z pohľadu paralelizácie algoritmu *FDDS* nie je potrebná žiadna špeciálna synchronizácia vlákien, ak neberieme do úvahy prípady ako nasledujúci. Vlákno dokončí riešenie určitého strategického profilu a vyžiada si z fronty ďalší. V tomto momente sa môže stať, že je fronta práve prázdna. Vzniknutú situáciu je možné vyriešiť pozastavením vlákna na určitú dobu, po uplynutí ktorej môže dotaz opäť zopakovať. Vzhľadom na to je teda nutné zabezpečiť akurát prístup k zdielaným dátam, čo je možné dosiahnuť použitím mutexu, napríklad z knižnice *boost* (*boost::mutex* - najjednoduchší typ mutexu, uzamknuteľný iba raz). Funkcionalita spojená s odomykaním a zamykaním mutexu je riešená použitím objektu *boost::mutex::scoped_lock*, ktorému je do konštruktora predaná referencia na mutex. Uzamknutie takýmto spôsobom je zobrazené na ukážke 3.2. Pri zaniknutí objektu *scoped_lock* je v rámci jeho deštruktora mutex odomknutý.

Listing 3.2: Ukážka práce s objektom boost::mutex

```
boost::mutex mutex;
...

int F()
{
```

```
    boost::mutex::scoped_lock lock(taskMutex);
    ...
}
```

3.4 Výpočet best-response

Jednou z hlavných podúloh pri implementácii algoritmu *FDDS* je nájdenie best-response stratégií určitého hráča na daný strategický profil, napríklad pomocou algoritmu 2. Pre rýchlu konvergenciu k výsledku sa predpokladá dobrá rozlíšiteľnosť strategického priestoru, čo by pre výstup algoritmu znamenalo jednoprvkovú množinu. V prípade nesplnenia tohoto predpokladu je možné zabrániť nechcenému vetveniu v stavovom priestore použitím rôznych prístupov k množine best-response stratégií. Ak táto množina obsahuje viacero prvkov je možné použitie

1. všetkých prvkov,
2. jedného z prvkov (napríklad náhodne vybraného).

Výhodou druhého spôsobu je rýchlejšie dopracovanie sa k výsledku, ale cenou za to môže byť vynechanie niektorých dôležitých stratégií v zredukovanej hre. Z tohto dôvodu by bola vhodná vo výslednom nástroji možnosť voľby medzi oboma spôsobmi.

Algorithm 2 Algoritmus Best-response

Require: $i \in Q$
Require: $s \in S$
  $MAX \leftarrow - \propto$
  $BR \leftarrow \emptyset$
  for all $s_i \in S_i$ **do**
    $U \leftarrow CellModel(s_i, s_{-i})$
    if $MAX \leq U_i$ **then**
      if $MAX < U_i$ **then**
        $MAX \leftarrow U_i$
        $BR \leftarrow \emptyset$
      end if
      $BR \leftarrow BR \cup s_i$
    end if
  end for
  return BR

3.5 Množina počiatočných uzlov

Jedným samostatným podproblémom je zvolenie vhodnej množiny počiatočných uzlov. Túto množinu je nutné zvoliť s ohľadom na čo najväčšiu pravdepodobnosť odhalenia všetkých dôležitých stratégií jednotlivých hráčov, ale zároveň je dôležité vyhnutie sa zbytočne veľkému prehľadávaniu stavového priestoru. V prípade existencie dominantných stratégií, je na ich odhalenie postačujúce začať s jediným uzlom, a to z dôvodu, že každá dominantná stratégia určitého hráča i sa prejaví v každom jeho kontexte rozhodovania s_{-i}. Avšak pre

úspešne riešenie hier bez dominantných stratégií je nutná voľba širšej množiny uzlov. Autor metódy *FDDS* navrhuje vygenerovanie takzvanej *Initial safe base* 2.5.2 (ďalej *ISB*). Jedná sa o množinu takých strategických profilov, v ktorých je obsiahnutá každá stratégia každého hráča jedenkrát (v prípade rozdielneho počtu stratégií jednotlivých hráčov, budú musieť byť niektoré stratégie hráčov s ich menším počtom použité viackrát). Počet počiatočných strategických profilov tejto množiny je rovný veľkosti najväčšej množiny stratégií jednotlivých hráčov.

$$\forall i \in Q, s_i \in S_i : \exists s \ = (s_i, s_{-i}) \in S_{ISB}$$

Spôsob vygenerovania *ISB* je zachytený na následujúcej ukážke algoritmu 3.

Algorithm 3 generovanie *Initial safe base*

Require: Q
Require: $S_i, \forall i \in Q$
  $ISB \leftarrow \emptyset$
  for all $q \in Q$ **do**
    ${S'}_q \leftarrow S_q$
  end for
  $end \leftarrow false$
  while $\neg end$ **do**
    $s \leftarrow \emptyset$
    $end \leftarrow true$
    for all $q \in Q$ **do**
      if ${S'}_q \neq \emptyset$ **then**
        $i \leftarrow uniform(0, |{S'}_q|)$
        $s[q] \leftarrow {S'}_q[i]$
        ${S'}_q \leftarrow {S'}_q \setminus {S'}_q[i]$
        $end \leftarrow false$
      else
        $i \leftarrow uniform(0, |S_q|)$
        $s[q] \leftarrow S_q[i]$
      end if
      if $\neg end$ **then**
        $ISB \leftarrow ISB \cup s$
      end if
    end for
  end while
  return ISB

Použitie takejto množiny počiatočných uzlov pre dvojhráčovú hru znamená prehľadanie celého stavového priestoru. Je to dôsledkom toho, že táto množina obsahuje všetky existujúce kontexty s_{-i} pre každého hráča $i \in Q$ a následne pri hľadaní best-response stratégie daného hráča sú postupne na každý z týchto kontextov aplikované všetky jeho stratégie. S vyšším počtom hráčov však už toto neplatí a je prehľadávaná iba menšia časť stavového priestoru. Túto závislosť budeme experimentálne demonštrovať v kapitole 5.

Na druhej strane je nevýhodou možnosť neodhalenia úplne všetkých best-response stratégií. Takýto prípad môže nastať najmä vtedy, keď sa určitá best-response stratégia prejavuje iba v jednom (alebo inom relatívne nízkom počte vzhľadom na veľkosť hry) kontexte s_{-i}. Tabuľka 3.1 je zobrazená hra troch hráčov demonštrujúca takýto prípad.

1. Za ISB zvolíme profily: $ISB = \{[A, D, G], [B, E, H], [C, F, I]\}$
2. S prvými dvoma z nich súhlasia všetci hráči.
3. S tretím profilom $[C, F, I]$ súhlasia iba hráči 1 a 2, zatiaľ čo hráč 3 sa odkloní do profilu $[C, F, H]$
4. S profilom $[C, F, H]$ už potom súhlasia všetci hráči.
5. Výsledná štruktúra GRP by obsahovala následovné profily:
   - $[A, D, G]$ - súhlasia všetci hráči
   - $[B, E, H]$ - súhlasia všetci hráči
   - $[C, F, I]$ - súhlasia hráči 1 a 2
   - $[C, F, H]$ - súhlasia všetci hráči
6. Z toho vyplývajúca zredukovaná hra by obsahovala pre jednotlivých hráčov tieto stratégie $S_1^{'} = \{A, B, C\}, S_2^{'} = \{D, E, F\}, S_3^{'} = \{G, H\}$

hráč 1, stratégia A			
hráč 2 \ hráč 3	G	H	I
D	8, 8, 8	1, 1, 3	1, 1, 1
E	1, 1, 1	1, 3, 1	8, 1, 8
F	1, 1, 1	4, 4, 1	1, 1, 1

hráč 1, stratégia B			
hráč 2 \ hráč 3	G	H	I
D	1, 1, 1	1, 1, 1	9, 9, 9
E	1, 1, 1	8, 8, 8	1, 1, 1
F	8, 8, 8	1, 1, 1	1, 1, 1

hráč 1, stratégia C			
hráč 2 \ hráč 3	G	H	I
D	1, 4, 1	1, 1, 1	1, 1, 1
E	8, 8, 8	1, 1, 1	1, 1, 1
F	1, 1, 1	7, 7, 7	8, 8, 0

Tabuľka 3.1: Demonštrácia neodhalenia všetkých best-response stratégií

Z príkladu je patrné, že by výsledná zredukovaná hra neobsahovala jednu z best-response stratégií hráča 3 a s ňou aj jedno z rýdzich Nashových ekvilibrií. V dôsledku toho by sa dalo predpokladať, že algoritmu *FDDS* by mohol mať problémy s odhaľovaním izolovaných best-response stratégií prejavujúcich sa iba v malom počte strategických profilov. O analýzu takýchto situácií sa pokúsime aj v niektorých z testovacích príkladov.

Kapitola 4

Implementácia knižnice

Táto kapitola bude venovaná bližšiemu popisu implementácie nástroja *FddsSolver* a jednotlivých jeho súčastí. Naša pozornosť bude zameraná najmä na vytvorené dátové štruktúry, realizáciu zdieľaných dát a niektoré zaujímavejšie aspekty samotného algoritmu. Pri implementácii nástroja sme dbali o dosiahnutie čo najväčšej efektivity výpočtu, ako aj využitia nástroja pre riešenie širokého spektra problémov a prípadného jednoduchého rozšírenia do budúcnosti. Pre ľahšiu orientáciu v opisovaných triedach je na obrázku 4.1 k dispozícii zjednodušený diagram tried.

4.1 Dátové štruktúry

Základnými stavebnými prvkami nástroja sú objekty vyplývajúce zo základných definícií prezentovaných v kapitole 2. Jedná sa predovšetkým o reprezentáciu elementov počnúc stratégiami a úžitkami hráčov, až po komplikovanejšie prvky, ako sú napríklad samotná hra či graf dosiahnuteľných profilov.

4.1.1 Reprezentácia hry

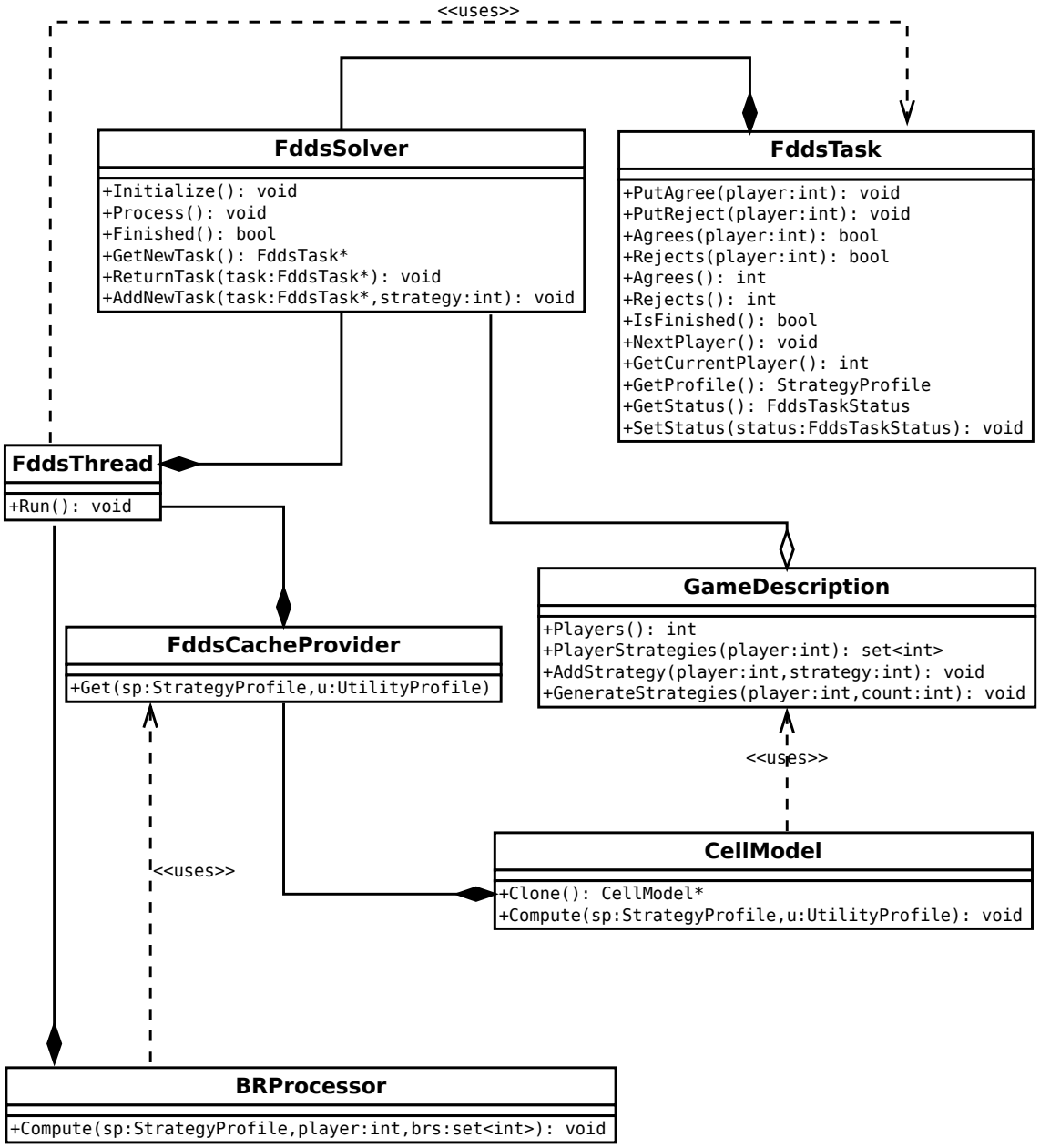
Zadanie hry *FddsSolveru* je realizované vytvorením dvoch objektov. Prvým krokom je vytvorenie inštancie triedy *GameDescription* popisujúcej strategickú hru. Pre popis hry je nutné zadať

1. počet hráčov hry,
2. stratégie pre jednotlivých hráčov. Inicializačné metódy ich umožňujú zadávať samostatne alebo ako intervaly.

Následne je nutné vytvorenie potomka abstraktnej triedy *CellModel*, ktorej je predaná referencia na *GameDescription*. Abstraktná trieda obsahuje dve virtuálne metódy *Clone* a *Compute*. Prvá z nich má za úlohu vytvárať svoje kópie (potrebné pre jednotlivé vlákna vytvorené neskôr). Metóda *Compute* počíta samotné úžitky jednotlivých hráčov pre zadaný strategický profil.

Pre dva základné stavebné prvky týchto objektov boli zvolené tieto dátové typy.

- Pre stratégiu bol z praktických dôvodov zvolený dátový typ *int* (strategický profil bude kľúčom do hašovacích tabuliek uchovávajúcich vypočítané úžitky a graf dosiahnuteľných profilov). V prípade ak je stratégia viacrozmerný rozhodovací problém, je nutné mapovanie na jednu hodnotu.



Obr. 4.1: Zjednodušený diagram tried

- Pre úžitok bola vytvorená šablóna, ktorá umožňuje voľbu jedného zo základných dátových typov. Výsledkom simulácií reálnych modelov spravidla nebýva iba jedno číslo. Pre tieto prípady je možnosť definovania úžitku ako štruktúry viacerých hodnôt. Jediným obmedzením je nutnosť definovania operátora <= pre tento typ (v prípadne nutnosti aj kopírovacieho konštruktora), keďže sú potrebné pre výpočty best-response stratégií.

4.1.2 Reprezentácia uzlu *GRP*

Uzol grafu *GRP* je reprezentovaný triedou *FddsTask*. Jedná sa o jednoduchú triedu uchovávajúcu informáciu o strategickom profile, ku ktorému sa viaže a hráčoch, ktorí s daným profilom súhlasia alebo nesúhlasia. Tieto zoznamy hráčov sú z dôvodu efektivity implementované pomocou asociatívneho kontajneru *std::set*, ktorý uchováva unikátné (čo indexy jednotlivých hráčov splňujú) elementy, ktoré slúžia sami sebe ako kľúče. Účelom tohoto kontajneru je rýchly prístup k elementom podľa ich kľúča, typicky sa jedná o implementáciu pomocou binárneho stromu. Inštancia tejto triedy počas svojho životného cyklu prechádza niekoľkými vnútornými stavmi (pripravená na spracovanie, spracovávaná, zrušená alebo vyriešená).

4.2 Zdieľané dátové štruktúry

Táto sekcia bude venovaná popisu implementovaných zdieľaných dátových štruktúr *FddsSolveru*, medzi ktoré sa radí cache vypočítaných úžitkov, cache uzlov *GRP* a s nimi úzko spojené listy uzlov vyriešených, zrušených a pripravených na riešenie (zmysel cache tu spočíva iba v rýchlom prístupe k položkám jednotlivých listov).

4.2.1 Cache

Oba spomínané typy cache sú implementované pomocou použitia asociatívneho kontajneru *unordered_map*. Jedná sa o hašovaciu tabuľku, ktorá je súčasťou prichádzajúceho štandardu *C++0x*. Túto triedu je možné použiť špecifikovaním dvoch až piatich šablón, pričom pre potreby *FddsSolvera* sú postačujúce prvé tri.

Listing 4.1: Deklarácia hašovacej tabuľky

```
template <typename TValue>
struct FddsHashType
{
    typedef std::unordered_map <StrategyProfile, TValue, FddsHashProfile> CacheType;
};
```

- *StrategyProfile* - strategický profil, ktorý je unikátnym identifikátorom elementu v hašovacej tabuľke
- *TValue* - šablóna pre typ elementu ukladaného do *unordered_map* (pole úžitkov alebo uzol *GRP*).
- *FddsHashProfile* - je hašovacia funkcia mapujúca strategický profil na číselnú hodnotu typu *size_t* (4.2)

$$S \rightarrow \text{size\_t}$$

Listing 4.2: Deklarácia použitej hašovacej funkcie

```
struct FddsHashProfile
{
    size_t operator() (const StrategyProfile &profile) const
    {
        size_t r = 0;
```

```
            for(std::array<int, PlayersCount>::const_iterator i = profile.begin(); i != profile.end(); ++i)
            {
                r *= 2251;
                r += (*i);
            }
            return (size_t) r;
        }
    };
```

Takto deklarovaný typ je následne zapúzdrený v objektoch *FddsCache* a *FddsSynchronizedCache*, ktoré poskytujú rozhranie pre prácu s touto štruktúrou. Druhý z nich navyše rieši aj riadenie prístupu z pohľadu paralelizácie výpočtu.

4.2.2 Cache vypočítaných úžitkov

Jedná sa o kľúčovú dátovú štruktúru z pohľadu výkonnosti a efektívnosti algoritmu. Dôvodom je veľmi častý prístup vlákien riešiacich jednotlivé uzly *GRP*. Pri riešení strategického profilu pre hráča i (hľadanie množiny best-response stratégií na profil s_{-i}) dôjde k tomuto prístupu $|S_i|$-krát bez ohľadu na to, či sa hľadané hodnoty pre daný profil v cache nachádzajú (v oboch prípadoch je nutný minimálne dotaz na existenciu úžitkov pre daný profil). Samotná hašovacia tabuľka je odvodená z typu *FddsSynchronizedCache* špecializovaním šablóny (*TUtility* je šablóna deklarujúca typ úžitku hráča).

Listing 4.3: Deklarácia cache úžitkov

```
FddsSynchronizedCache<std::array<TUtility, PlayersCount>>
```

4.2.3 Cache uzlov *GRP*

Táto dátová štruktúra je odvodená špecializovaním typu *FddsCache*:

Listing 4.4: Deklarácia cache uzlov *GRP*

```
FddsCache<FddsTask *>
```

Na rozdiel od cache z predchádzajúceho odstavcu nerieši synchronizáciu prístupu jednotlivých vlákien. Táto zodpovednosť je prenechaná nadradenému objektu.

4.2.4 Zoznamy uzlov *GRP*

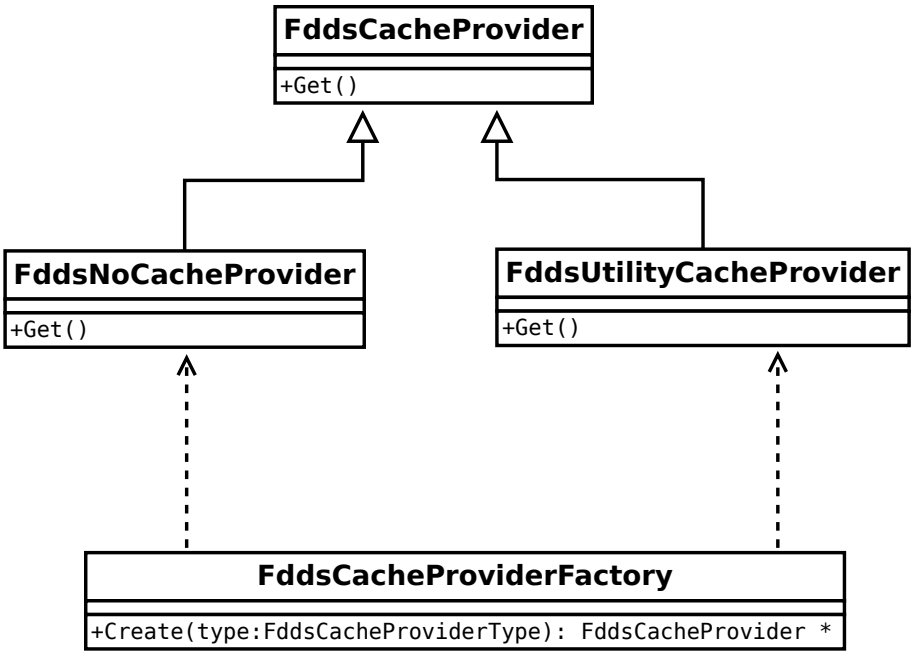
Tieto zoznamy sú rovnako ako cache ulov *GRP* súčasťou objektu *FddsSolver*. V zozname *taskList* sú uchované referencie na uzly *GRP*, ktoré majú byť riešené. Po vyriešení je uzol presunutý do zoznamu vyriešených uzlov, analýzou ktorého je po ukončení chodu algoritmu vygenerovaná redukovaná hra. Keďže tieto zoznamy a cache uzlov *GRP* pracujú s rovnakými dátami, musia nevyhnutne podliehať spoločnému riadeniu prístupu.

4.3 Výpočet úžitkov v strategických profiloch

Ako už bolo spomínané úžitky hráčov v jednotlivých strategických profiloch budú počítané metódou *Compute* objektu *CellModel*. Vzhľadom nato, že algoritmus *FDDS* potrebuje opakovane pristupovať k úžitkom v určitom strategickom profile, je vhodné ukladanie už vypočítaných hodnôt. Navyše s rastúcou komplexnosťou a časovou náročnosťou *CellModelu*

sa toto stáva nevyhnutnosťou. Takýto prístup však môže znamenať zbytočnú pamäťovú náročnosť pre jednoduchšie hry, kde výpočet úžitkov zaberá iba relatívne krátku dobu. Preto bola vytvorené abstraktná *FddsCacheProvider* zapúzdrujúca prístup k objektu *CellModel* a dovoľujúcej realizáciu rôznych stratégií uchovávania vypočítaných hodnôt úžitkov. *FddsSolver* umožňuje použitie dvoch takýchto prístupov, ktoré sú implementované dvoma potomkami abstraktnej triedy *FddsCacheProvider*:

- *FddsNoCacheProvider* - ako vyplýva z názvu, jedná sa o spôsob bez ukladania vypočítaných úžitkov. Pri každej požiadavke je priamo vyvolaný výpočet pomocou *CellModel*
- *FddsUtilityCacheProvider* - spôsob s využitím ukladania do cache, ktorá musí byť k dispozícii všetkým vláknam, a preto tu dochádza k potrebe synchronizácie ich prístupu.



Obr. 4.2: Triedy poskytujúce prístup k úžitkom v strategických profiloch

Algoritmus *FDDS* môže počas svojho chodu počítať best-response stratégie určitého hráča opakovane pre ten istý kontext rozhodovania. Na základe tejto myšlienky bola pri implementácii zvažovaná aj možnosť s prístupom ku grafu dosiahnuteľných profilov. V prípade ak už daný kontext rozhodovania určitého hráča bol riešený, je možné v *GRP* nájsť profil, do ktorého chce tento hráč prejsť (prípadne ostať v aktuálnom). Týmto spôsobom by pri riešení profilu pre nejakého hráča bolo možné nahradiť prístupy do cache vypočítaných úžitkov prístupmi do *GRP*. Výhodou tohoto by bolo presunutie určitého počtu prístupov z viac vyťaženej na menej vyťaženú zdieľanú dátovú štruktúru. Na druhej strane by si takéto riešenie vyžadovanú zvýšenú réžiu spojenú najmä s riadením prístupu k zdieľaným dátam ako aj určitú komplikáciu a zneprehľadnenie rozhraní jednotlivých medzi sebou komunikujúcich objektov, a preto bolo od tejto myšlienky nakoniec upustené.

4.3.1 Trieda *FddsUtilityCacheProvider*

Úlohou tohto objektu je abstrakcia prístupu k úžitkom hráčov pre požadovaný strategický profil. Pri takejto požiadavke na profil úžitkov určitého vlákna môžu nastať tri rôzne situácie.

1. Nie sú prítomné. Požadovaný strategický profil je uložený do zoznamu práve počítaných profilov a je zahájený výpočet pomocou objektu *CellModel*. Po jeho ukončení je výsledok uložený do cache, profil je odobraný zo zoznamu aktuálne riešených profilov a výsledok je vrátený.

2. Sú prítomné v cache. Požadované dáta sú vytiahnuté z cache a poskytnuté dotazovateľovi

3. Sú práve riešené. V tomto prípade je žiadajúce vlákno na krátku dobu uspané. Po jej uplynutí sa celý proces opakuje až kým výpočet pre daný profil nie je ukončený.

Pre správnu funkcionalitu je nutné, aby niektoré z uvedených operácií prebehli atomicky. Pri negatívnej odpovedi na dotaz o existencii dát pre daný profil, je tento profil súčasne vložený do zoznamu riešených profilov, keďže sa predpokladá, že bude okamžite zahájený jeho výpočet. Profil je z tohto zoznamu odstránený atomicky spolu s vkladaním výsledných úžitkov do cache.

4.4 Implementácia algoritmu

Samotné telo algoritmu *FDDS* je implementované pomocou dvoch tried, ktoré využívajú a pracujú s dátovými štruktúrami popísanými na predchádzajúcich stranách. Vonkajší cyklus algoritmu pomocou triedy *FddsSolver* a vnútorný cyklus, ktorý je možno spracovávať paralelne, triedou *FddsThread*. Následujúce sekcie popisujú tieto triedy spolu aj s jednotlivými podproblémami spojenými s ich implementáciou.

4.4.1 Trieda *FddsSolver*

Túto triedu je možné považovať za samotný mozog algoritmu, ktorého hlavnou úlohou je riadenie algoritmu spočívajúce v inicializácii všetkých potrebných dátových štruktúr a vlákien, riadenie prístupu k zdieľaným dátovým štruktúram, rozdeľovanie úloh jednotlivým vláknam ako aj rozhodovanie o splnení ukončujúcich podmienok algoritmu. Navyše je možné túto triedu považovať za akési rozhranie pre užívateľa implementovaného nástroja. Životný cyklus inštancie triedy *FddsSolver* je možno rozdeliť do troch etáp.

- Inicializácia algoritmu, ktorá zahrňuje vytvorenie a nastavenie vnútorných objektov na základe konštrukčných parametrov, inicializácia počiatočných uzlov *GRP*.

- Samotný chod algoritmu, vyžadujúci vytvorenie požadovaného počtu vlákien, riadenie algoritmu a rozdeľovanie úloh.

- Výstup algoritmu, čiže vygenerovanie zredukovanej hry.

Ukážka kódu 4.5 demonštruje prácu s objektom triedy *FddsSolver*. Pri vytvorení objektu je nutné špecializovať šablónu na dátový typ použitý pre reprezentáciu úžitku hráča. Parametre konštruktora sú potomok triedy *CellModel* a inštancia triedy *GameDescription*.

Ďalším voliteľným argumentom je spôsobom prístupu k úžitkom v strategických profiloch (v tomto prípade s použitím cache úžitkov). Následne je nutné inicializovať počiatočné uzly v *GRP* a spustiť chod algoritmu. Po jeho dokončení sa nechá vygenerovať zredukovaná hra.

Listing 4.5: Ukážka práce s objektom *FddsSolver*

```
FddsSolver<int> *solver = new FddsSolver<int>(cellModel, game, FDDS_UTILITY_CACHE);
solver->Initialize();
solver->Process();
GameDescription *reducedGame = solver->ReducedGame();
```

Zahájenie výpočtu je spustené vyvolaním metódy *Process*. Táto metóda vytvorí jednotlivé vlákna, ktorých počet je špecifikovaný hodnotou konštanty *ThreadCount* nachádzajúcej sa v hlavičkovom súbore *fdds.h*. V tomto súbore je prítomná aj ďalšia konštanta *PlayersCount*, ktorá špecifikuje počet hráčov hry. Prítomnosť tejto konštanty a nutnosť jej modifikácie čiastočne komplikuje používanie nástroja. Jej prítomnosť je nutná z dôvodu možnosti definície statických polí pre vektory stratégií a úžitkov hráčov. Možnosťou bolo aj zavedenie konštanty udávajúcej maximálny počet hráčov, pričom by pri jeho nenaplnení ostali niektoré jeho prvky nevyužité. Nevýhodou tohoto riešenia je okrem zvýšenej pamäťovej náročnosti aj nastávajúci problém pri hašovaní strategického profilu, ktoré používa všetky prvky poľa stratégií. Takto by hašovanie záviselo aj na niekoľkých neplatných hodnotách (v prípade menšieho ako maximálneho počtu hráčov).

Činnosť vlákien je okamžite po vytvorení spustená pričom *FddsSolver* preberá úlohu poskytovateľa prístupu k zdielaným dátam (konkrétne k zoznamom a cache *FddsTaskov*). Nutnosť tohoto prístupu nastáva v štyroch prípadoch, každý z nich je ošetrený samostatnou metódou *FddsSolvera*:

- detekcia splnenia ukončovacích podmienok algoritmu - *Finished*,
- pridelenie novej úlohy vláknu - *GetNewTask*,
- vrátenie úlohy vláknom - *ReturnTask*,
- požiadanie o vytvorenie novej úlohy - *AddNewTask*.

Detekcia ukončenia algoritmu

Metóda vráti hodnotu *true* v prípade naplnenia ukončovacích podmienok algoritmu. Tento prípad nastáva ak je fronta uzlov čakajúcich na spracovanie prázdna a zároveň je počet práve riešených uzlov rovný nule. Keďže táto rozhodovacia podmienka je závislá na zdielaných dátach, je telo funkcie chránené mutexom (toto platí aj pre následujúce metódy, ktoré pristupujú k rovnakým dátam).

Pridelenie novej úlohy

Vracia prvý nevyriešený (čiastočne vyriešený alebo nezahájený) uzol z fronty uzlov čakajúcich na spracovanie a nastavuje status uzlu na práve spracovávaný. Ak je fronta prázdna vracia hodnotu *NULL*. V tomto prípade by sa mohlo zdať, že vlákno žiadajúce o nový uzol môže ukončiť svoju činnosť. To však samozrejme pravda nie je z dôvodu možnosti existencie ďalšieho práve spracovávaného uzlu.

Vrátenie úlohy

Spracovávaný uzol je po jeho vyriešení pre určitého hráča vrátený *FddsSolveru*, ktorý detekuje, či je riešenie uzlu už dokončené a následne ho podľa toho zaradí do zoznamu čakajúcich na spracovanie alebo vyriešených uzlov.

Vytvorenie novej úlohy

V prípade ak sa pri riešení strategického profilu pre daného hráča zistilo, že hráč s daným profilom nesúhlasí a chce sa odkloniť do iného, je nutné požiadať solver o zaradenie cieľového uzlu do fronty uzlov čakajúcich na spracovanie. Ak požiadaný uzol ešte neexituje, je nutné ho vytvoriť (pričom je možné hráča, ktorý sa chcel odkloniť od pôvodného profilu, nastaviť ako súhlasiaceho) a vložiť do fronty. Pre už existujúci *FddsTask* pre daný profil je nutné postupovať podľa jeho stavu. Ak nájdený *FddsTask* nie je práve spracovávaný niektorým z vlákien, je nutné nastavenie odkloňujúceho sa hráča ako súhlasiaceho (ak je task už čiastočne vyriešený, tak iba v prípade ak daný hráč ešte ako súhlasiaci nastavený nie je). Pre práve spracovávaný task toto nie je možné vykonať, keďže s ním práve pracuje iné vlákno. To aj napriek tomu, že telo metódy je chránené mutexom synchonizujúcim prístup k zdieľaným dátam, a to z dôvodu, že tieto dáta obsahujú iba referencie na inštancie triedy *FddsTask*.

4.4.2 Vlákno algoritmu

Trieda *FddsThread* v postate implementuje kroky 2 až 4 algoritmu *FDDS*. Jej úlohou je teda vyzdvihnutie si uzla z fronty uzlov, zvolenie jedného hráča, pre ktorého daný uzol ešte nie je vyriešený a nakoniec vyriešenie uzla pre daného hráča. Algoritmus 4 popisuje tento proces detailnejšie. Jednou z podúloh je nájdenie best-response stratégií pre riešeného hráča v danom strategickom kontexte. Pre dosiahnutie tohoto je nevyhnutný prístup k úžitkom hráčov v potrebných profiloch (prístupom do cache alebo pomocou *CellModel*) s využitím triedy *CacheProvider* spomenutej v predošlých častiach kapitoly.

Algorithm 4 Činnosť vlákna

```
while ¬solver.Finished() do
  task ← solver.GetNewTask()
  if task ≠ NULL then
    if ¬task.IsFinished() then
      q ← task.NextPlayer()
      s ← task.GetProfile()
      BR ← ComputeBrs(q, s)
      if s_q ∈ BR then
        task.PutAgree(q)
      else
        task.PutReject(q)
      end if
      for all b ∈ BR \ s_q do
        solver.AskNewTask(task, b)
      end for
    else
      solver.ReturnTask(task)
    end if
  else
    sleep()
  end if
end while
```

Kapitola 5

Analýza výsledkov experimentov

Funkčnosť implementovaného nástroja bola experimentálne odskúšaná na niekoľkých príkladoch, ktoré skúmajú a analyzujú rôzne vlastnosti algoritmu. Začneme jedným z typických problémov teórie hier, ktorým je Cournotov model oligopolu. Toto však bude jediným príkladom so základmi v reálnom svete. V ďalších experimentoch bola daná prednosť umelo vygenerovaným hrám, s určitými vlastnosťami vhodnými na testovanie algoritmu *FDDS*.

5.1 Cournotov model oligopolu

Jedná sa o ekonomický model trhu, kde skupina firiem Q súperí s predajom určitého homogénneho produktu. Každá firma $i \in Q$ si nezávislo volí množstvo jednotiek a_i, ktoré vyprodukuje s určitými nákladmi $C_i(a_i)$ (náklady na jednotlivé kusy sú zhodné), ktoré rastú lineárne s množstvom vyprodukovaných jednotiek. Všetky produkty sú predané za rovnakú cenu $P(A)$, ktorá je daná dopytom a ponukou A (celkové množstvo vyprodukované všetkými firmami spolu). P je nazývaná inverznou funkciou dopytu, ktorej hodnota klesá so zvyšujúcou sa ponukou, pokiaľ už nie je nulová.

$$\begin{aligned} C_i(a_i) &= ca_i \quad c > 0 \\ A &= \sum_{i \in Q} a_i \\ P(A) &= \begin{cases} \alpha - A, & \text{ak } A \leq \alpha \\ 0, & \text{ak } A > \alpha \end{cases} \end{aligned}$$

α a c sú konštanty reprezentujúce dopyt a náklady na výrobu jednotky produktu. Taktiež je nutné predpokladať nerovnosť $c < \alpha$, a to z dôvodu existencie ponuky A, pre ktorú by trhová cena bola vyššia ako celkové náklady firiem C. Pre obrat R_i a zisk π_i určitej firmy i platí:

$$\begin{aligned} R_i(a_1, a_2, \ldots, a_{|Q|}) &= a_i P(a_1 + a_2 + \ldots + a_{|Q|}) \\ \pi_i(a_1, a_2, \ldots, a_{|Q|}) &= R_i(a_1, a_2, \ldots, a_{|Q|}) - C_i(a_i) \end{aligned}$$

Cournotov model oligopolu obsahuje jedno unikátné Nashovo ekvilibrium:

$$a^* = (a_1^*, a_2^*, \ldots, a_{|Q|}^*) : a_i^* = \frac{\alpha - c}{|Q| + 1}, \quad \forall i \in Q$$

Z toho odvodený a implementovaný model obsahuje následovné voliteľné parametre:

1. počet hráčov (firiem) Q,
2. dopyt M,
3. náklady na jednotku produktu.

Jednotlivým hráčom sú vygenerované stratégie, reprezentujúce zvolené množstvá produktov, v intervale $\langle 0, M \rangle$. Ďalšie stratégie by nemali zmysel, keďže by zaručene viedli k záporným úžitkom. Pre úžitok hráčov bola vytvorená dvojdimenzionálna štruktúra. Jednou hodnotou je zisk hráča daný funkciou π_i a druhá hodnota je počet jednotiek produktu, ktoré tento hráč vyrobil. Predpokladom je, že ak pre určitého hráča i je jeho zisk v dvoch strategických profiloch rovnaký, bude preferovať ten, v ktorom vyrobil menšie množstvo produktov. Tento testovací príklad teda okrem modelu oligopolu aj demonštruje použitie *FddsSolvera* v kombinácii s vlastnou štruktúrou reprezentujúcou úžitky (preferencie) hráčov. Pre takúto štruktúru je nutná definícia operátora <=, čo je pre náš model docielené nasledovným spôsobom 5.1.

Listing 5.1: Definícia štruktúry reprezentujúcej preferencie hráčov

```
struct Utility
{
    int profit;
    int volume;
};

bool operator <=(Utility &a, Utility &b)
{
    if (a.profit < b.profit) return true;
    if (a.profit == b.profit && a.volume >= b.volume) return true;
    return false;
}
```

V tabuľke 5.1 sú zaznamenané výsledky vykonaných simulácií. Vo všetkých prípadoch zredukovaná hra obsahovala stratégie hráčov v intervale $\langle 0, \frac{M-c}{2} \rangle$ oproti pôvodným v intervale $\langle 0, M \rangle$. Tieto stratégie sú očakávaným a správnym výstupom, pričom hraničné stratégie určitého hráča sú best-response na profily, kde súčet produkcie ostatných hráčov je rovný alebo väčší M, respektíve rovný 0. Racionálne uvažujúci hráči však väčšinu týchto profilov hrať nebudú, a preto by sa dali zodpovedajúce stratégie považovať za zbytočné (aj napriek tomu, že spĺňajú definíciu užitočných best-response stratégií, ktorú sme si zaviedli). Výstupom algoritmu je aj Nashovo ekvilibrium, ktoré je v modeli oligopolu prítomné. V prípade so štyrmi hráčmi sa jedná o Nashovo ekvilibrium zmiešaných stratégií zodpovedajúce profilom $[10, 11, 10, 10]$, $[10, 10, 11, 10]$, $[11, 10, 10, 10]$ a $[10, 10, 10, 11]$.

Q	2	3	4
$\lvert S_i^r \rvert$	27	27	27
PNE	[17, 17]	[13, 13, 13]	-

Tabuľka 5.1: Výsledok redukcie modelu oligopolu ($M = 60$, c = 8)

5.2 Umelo vytvorené hry

Hlavnou myšlienkou tohoto typu testov je vytvorenie hier obsahujúcich stratégie zaujímavé z pohľadu algoritmu *FDDS*, teda stratégie, ktoré budú obsiahnuté aj vo výslednej zredukovanej hre. Boli navrhnuté dva hlavné spôsoby generovania hier, kde je možné zvoliť množinu stratégií, ktoré:

- sú dominantné voči ostatným nezvoleným stratégiám 9pričom medzi sebou žiadne dve z nich dominantné nie sú),
- tvoria množinu best-response stratégií jednotlivých hráčov (existencia dominovaných stratégií je v tomto prípade malá a zmenšuje sa s narastajúcim stavovým priestorom hry.

Hry vygenerované druhým spôsobom sú zaujímavejšie z pohľadu experimentovania s algoritmom *FDDS*, a preto bol v neskôr prezentovaných príkladoch použitý výlučne tento typ. Ďalšími možnosťami pri nastavovaní vlastnosti daných hier je možnosť modifikovania rozlíšiteľnosti stratégií v jednotlivých kontextoch hráčov, počtu kontextov rozhodovania hráčoch, v ktorých sa prejavia jeho jednotlivé best-response stratégie a takisto aj užitočnosť týchto stratégií.

5.2.1 Spôsob generovania hier

Generovanie prvého typu hier je relatívne jednoduché. V prípade, ak určitý hráč hrá nejakú zo zvolených stratégií, je postačujúce jeho úžitok v danom profile vynásobiť určitou konštantou (alebo konštantu pripočítať), čím sa zaručí, že daná stratégia bude dominovať všetky nezvolené. Ďalším krokom je zabránenie možnej dominancie medzi zvolenými stratégiami (alebo inak povedané, aby každá z týchto stratégií bola best-response aspoň na jeden kontext rozhodovania daného hráča). Toto je možné dosiahnuť tak, že každej stratégii priradíme profil (prípadne viac profilov), v ktorom bude mať daný hráč v danom kontexte najvyšší úžitok.

U druhého typu hier sa v podstate jedná o namapovanie každého kontextu s_{-i} hráča i na jednu z jeho best-response stratégií. Toto je dosiahnuté nasledovným spôsobom. Každému kontextu s_{-i} je priradený určitý index, ktorý je následne zahašovaný pomocou hašovacej funkcie (z dôvodu dosiahnutia rozumného rozloženia - napríklad aby dva za sebou nasledujúce strategické profily neboli zakaždým namapované na susedné best-response stratégie). Posledným krokom je takto získanú hodnotu namapovať na jednu zo zvolených stratégií. V prípade ak by sme mali záujem o dosiahnutie rôznej rozlíšiteľnosti stratégií, je nutné mapovanie jednotlivých kontextov na viacero stratégií. Toto sa dá jednoducho docieliť použitím viacerých rôznych hašovacích funkcií v druhom kroku.

Algoritmus 5 znázorňuje výpočet úžitku hráča i v strategickom profile s v takýmto spôsobom generovanej hre. U_{br} a U_{normal} sú funkcie generujúce úžitky hráča pre profily s jeho best-response stratégiou respektíve normálnou stratégiou, pričom musí platiť $U_{br}(s) > U_{normal}(s)$. V implementovaných testovacích príkladoch sme použili náhodné generovanie úžitkov v zvolených intervaloch, a preto sú tieto príklady použiteľné iba v kombinácii s cache úžitkov (bez použitia cache by pri opätovnom výpočte pre určitý profil dochádzalo k vráteniu rozdielnych výsledkov).

Algorithm 5 Výpočet úžitku hráča v umelo generovaných hrách

Require: $i \in Q$
Require: $s \in S$
  $k \leftarrow 0$
  for all $q \in Q$ **do**
    if $q \neq \imath$ **then**
      $k \leftarrow k * |S_q| + s_q$
    end if
  end for
  $k \leftarrow hash(k)$
  $br \leftarrow BRS[k \bmod |BRS_i|]$
  if $br = s_i$ **then**
    $u_i \leftarrow U_{br}(s)$
  else
    $u_i \leftarrow U_{normal}(s)$
  end if
  return u_i

5.3 Zložitosť algoritmu

Najvhodnejším spôsobom skúmania zložitosti algoritmu je vzhľadom na veľkosť prehľadaného stavového priestoru, čiže počet vyhodnotení *CellModelu*. V prípade zložitejších modelov býva táto funkcia typicky výpočtovo najnáročnejšia. Navyše ak je použitá cache vypočítaných úžitkov má rozhodujúci vplyv aj na priestorovú zložitosť, vzhľadom na to, že veľkosť preskúmaného stavového priestoru priamo ovplyvňuje veľkosť tejto cache. Tá je zároveň aj najnáročnejšou dátovou štruktúrou z hľadiska pamäťových požiadaviek.

Veľkosť preskúmaného stavového priestoru sme vyhodnocovali vzhľadom na

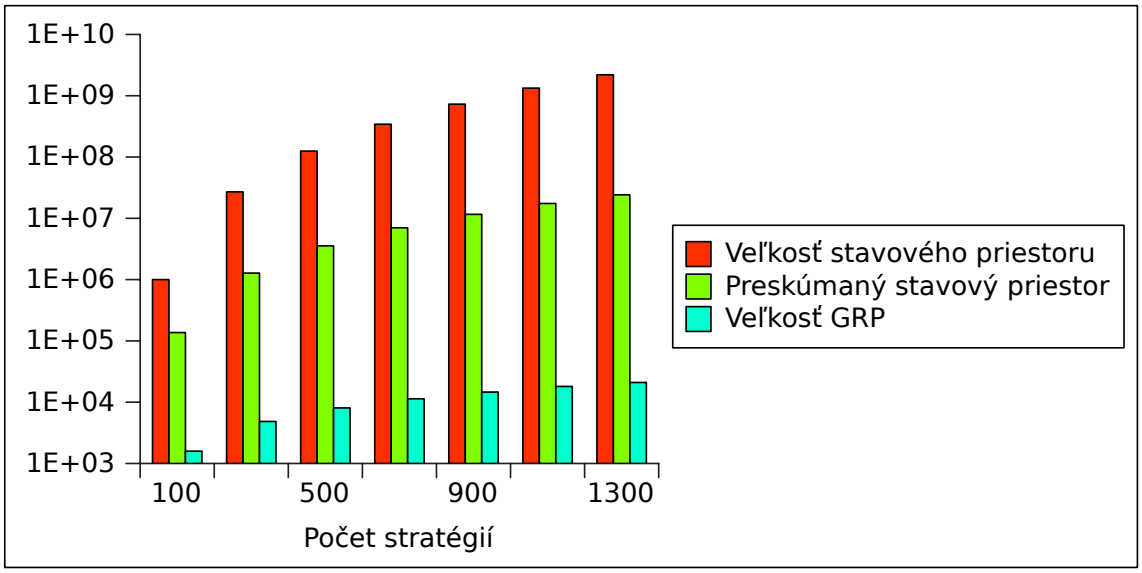
- počet stratégií hráčov (graf 5.1),
- počet hráčov (graf 5.2),
- veľkosť množiny best-response stratégií (graf 5.3),
- rozlíšiteľnosť stavového priestoru (tabuľka 5.2).

Pre každý z týchto parametrov bola vykonaná séria testov, v rámci ktorej boli všetky ostatné parametre hry okrem skúmaného nemenné.

$\|BR(s_{-i}\|)$	1	2	3	4	5
Preskúmaný stavový priestor	298608	488268	501683	502964	502843
Veľkosť GRP	3777	16461	26948	35587	42842

Tabuľka 5.2: Zložitosť algoritmu vzhľadom na rozlíšiteľnosť stavového priestoru. Testovaná hra je zložená z troch hráčov s 100 stratégiami ($|S| = 100^3$) a $BR_i = 20$.

Vo všetkých vyššie uvedených príkladoch bola výsledná zredukovaná hra best-response ekvivalentná pôvodnej hre, čiže boli zachované všetky best-response stratégie. Toho bolo dosiahnuté aj vďaka tomu, že vygenerované hry neobsahovali žiadne izolované best-response



Obr. 5.1: Zložitosť algoritmu vzhľadom na počet stratégií hráčov. Testovaná hra je zložená z troch hráčov, pričom množina best-response stratégií každého z nich obsahuje 5 prvkov. Stavový priestor je dobre rozlíšiteľný (na každý kontext s_{-i} hráča i) platí $|BR(s_{-i})| = 1$.

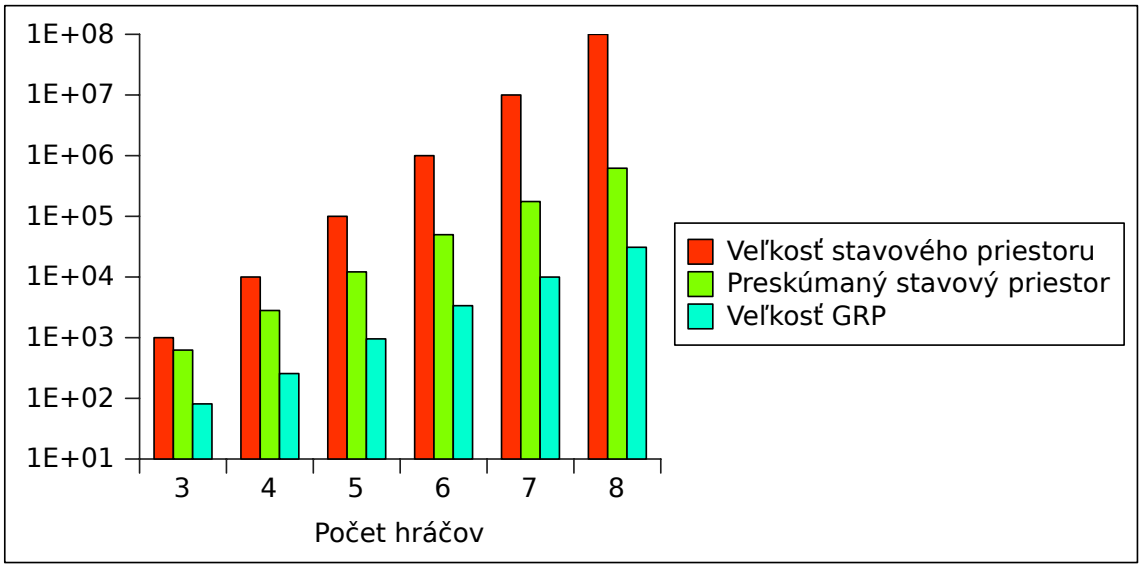
stratégie. Z hľadiska zložitosti algoritmu je vidieť, že na redukciu hry bolo potrebné preskúmanie len zlomku z celkových strategických priestorov hier. Zaujímavé môže byť porovnanie testov s ohľadom na prvé dve kritéria (vplyv počtu stratégií a počtu hráčov). Test so 100 stratégiami jednotlivých hráčov a test so šiestimi hráčmi boli vykonávané na hrách s podobnou veľkosťou stavového priestoru. Výsledný počet vyhodnocovaných strategických profilov je tiež porovnateľný. Treba si však uvedomiť, že druhý z týchto testov je výraznejšie časovo aj pamäťovo náročný, a to v dôsledku toho, že sa jedná o viacrozmerný rozhodovací problém. Každý z analyzovaných uzlov bolo nutné riešiť pre väčší počet hráčov. Tento počet sa navyše prejavil aj na fyzickej veľkosti dátových štruktúr, najmä na cache vypočítaných úžitkov (pre každý profil je nutné ukladať hodnoty pre všetkých hráčov).

5.4 Best-response ekvivalencia zredukovaných hier

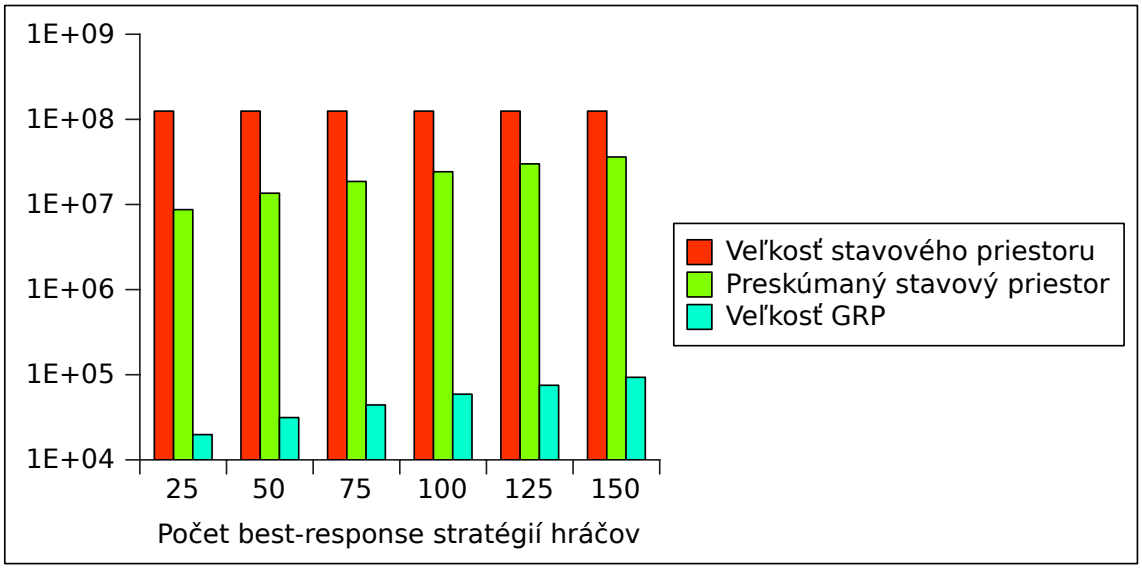
Ako bolo v jednej z predošlých kapitol 3.5 naznačené, algoritmus *FDDS* v kombinácií s použitím *ISB* nezaručuje vo všetkých prípadoch to, že zredukovaná hra bude best-response ekvivalentná s pôvodnou hrou (platí aj pre ekvivalenciu len s ohľadom na užitočné best-response stratégie). Príčinou tohoto fenoménu je možná existencia stratégií, ktoré síce patria do množiny BR, ale prejavujú sa len vo veľmi malom množstve strategických profilov. V tejto sekcii budú experimentálne skúmané hry, ktoré práve takéto stratégie obsahujú. Z dosiahnutých výsledkov sa pokúsime vyhodnotiť pravdepodobnosť odhalenia takýchto stratégií.

Odhalenie izolovanej best-response stratégie vo všeobecnosti závisí od viacerých faktorov.

- Veľkosť stavového priestoru.



Obr. 5.2: Zložitosť algoritmu vzhľadom na počet hráčov. Hra je tvorená hráčmi s 10 stratégiami a best-response množinou o veľkosti 2. Strategický priestor je opäť dobre rozlíšiteľný.



Obr. 5.3: Zložitosť algoritmu vzhľadom na počet best-response stratégií hráčov. Skúmaná hra je tvorená z troch hráčov s 500 stratégiami. Stavový priestor je dobre rozlíšiteľný.

- Počet kontextov, v ktorom sa daná izolovaná stratégie prejaví (v súvislosti s predchádzajúcim bodom).
- Užitočnosť danej best-response stratégie, ktorá znamená väčšiu šancu sa do daného profilu prepracovať prostredníctvom iného hráča. Navyše ak daná stratégia užitočnou nie je, jej neobjavenie nie je vôbec podstatné z pohľadu best-response ekvivalencie hier vzhľadom na užitočné stratégie.

- Celková štruktúra hry, ktorá zahrňuje veľkosti best-response množín jednotlivých hráčov, rozlíšiteľnosť a vetvenie strategického priestoru s ohľadom na *GRP*.

Úspešnosť odhalenia takejto izolovanej best-response bola experimentálne zisťovaná na nasledujúcich dvoch hrách líšiacich sa vo veľkosti množín BR ($Q = 3$ $|S_i| = 100$, $|BR_i| = 10$ a $|BR_i| = 5$). Jednému z hráčov bola následne dogenerovaná ďalšia best-response stratégia s možnosťou manipulácie počtu kontextov, v ktorých sa prejaví. Boli generované tri typy takýchto kontextov, kde izolovaná best-response stratégia bude

- iba neužitočnou,
- užitočnou alebo neužitočnou (pravdepodobnosť daná počtom hráčov, množstvom stratégií a best-response stratégií. V použitých hrách to je konkrétne 19%/81% respektíve 9.75%/90.25%),
- iba užitočnou.

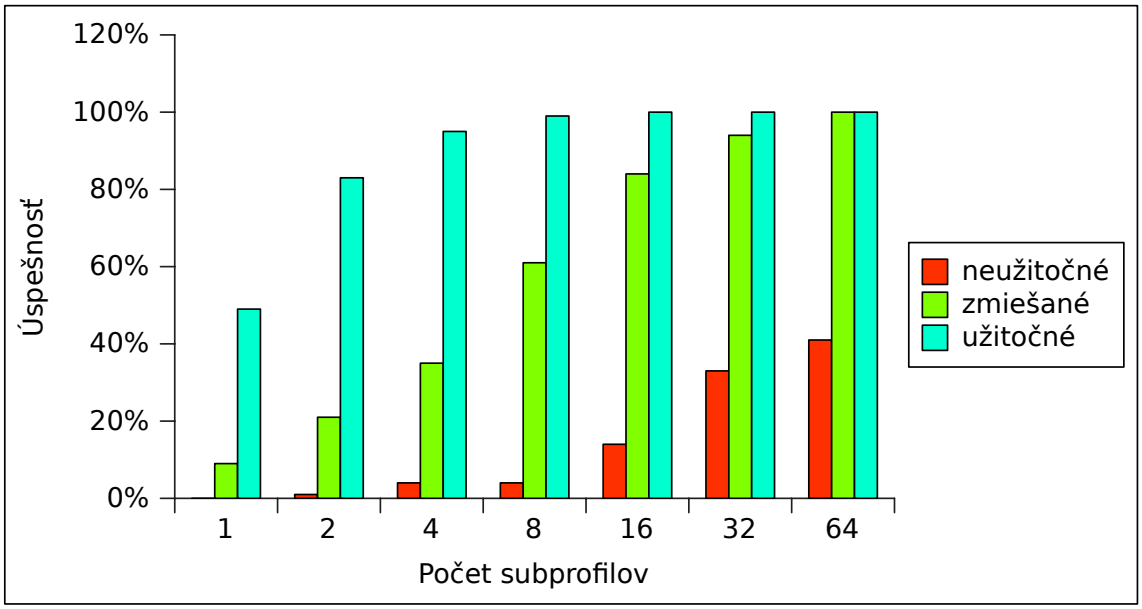
Pre každú kombináciu zloženú zo zvolených počtov kontextov a ich vyššie uvedených typov bolo vykonaných 100 simulácií a percentuálna úspešnosť odhalenia tejto stratégie v jednotlivých prípadoch je zobrazená v tabuľke 5.3 a na grafe 5.4.

p	1	2	4	8	16	32	64
neužitočné							
$\|BR_i\| = 5$	1%	2%	2%	10%	13%	28%	46%
$\|BR_i\| = 10$	0%	1%	4%	4%	14%	33%	41%
zmiešané							
$\|BR_i\| = 5$	8%	11%	23%	38%	62%	91%	97%
$\|BR_i\| = 10$	9%	21%	35%	61%	84%	94%	100%
užitočné							
$\|BR_i\| = 5$	46%	79%	95%	99%	100%	100%	100%
$\|BR_i\| = 10$	33%	65%	85%	99%	100%	100%	100%
$\|BR_i\| = 15$	35%	64%	85%	98%	100%	100%	100%

Tabuľka 5.3: Úspešnosť odhalenia izolovanej best-response stratégie.

Zo zaznamenaných výsledkov je patrné, že odhalenie užitočnej izolovanej best-response stratégie je výrazne pravdepodobnejšie ako neužitočnej. Dôvodom je to, že do daného strategického profilu sa je možné dostať prostredníctvom viacerých hráčov. Vo výraznej väčšine prípadov toto nastane prostredníctvom hráča i, práve vďaka ktorému sa izolovaná stratégia stáva užitočnou, keďže jeho best-response stratégia sa prejavuje vo výrazne väčšom počte profilov s_{-i}. V hrách generovaných v rámci tejto práce je tento počet približne daný pomerom celkového počtu kontextov s_{-i} k počtu best-response stratégií hráča i (závisí na použitej hašovacej funkcii).

Rozdielna veľkosť množín best-response stratégií sa na výsledkoch podpísala výrazne iba v druhej sérii testov, kde sa za príčinu dá považovať zväčšenie pravdepodobnosti generovania kontextov, na ktoré bude izolovaná stratégia užitočnou best-response. V tretej sérii testov je možné zachytiť akýsi trend znižovania úspešnosti vzhľadom na rastúci počet best-response stratégií. Stratégia, vďaka ktorej sa izolovaná best-response stala užitočnou, sa bude v dôsledku tohoto nárastu prejavovať v nižšom počte kontextov, čo nakoniec vedie k nižšej úspešnosti.



Obr. 5.4: Úspešnosť odhalenia izolovanej best-response stratégie.

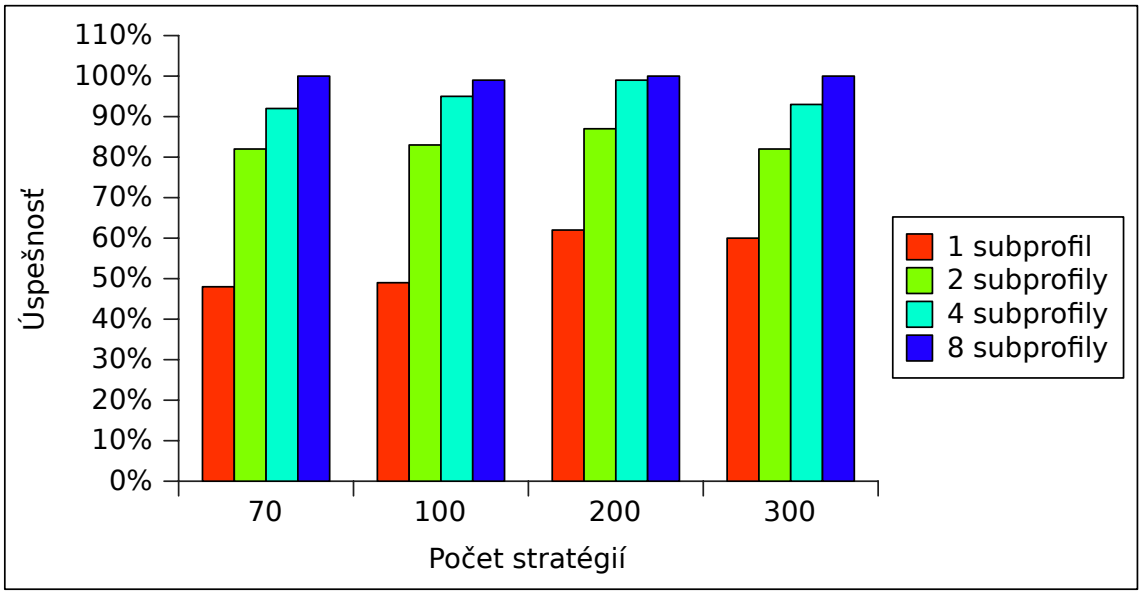
V ďalšom experimente bude naša pozornosť zameraná výlučne na užitočné izolované best-response stratégie, pričom predmetom skúmania bude vplyv počtu stratégií hráčov na úspešnosť detekcie izolovanej stratégie (graf 5.6). Zo zaznamenaných výsledkov vyplýva, že veľkosť množiny stratégií jednotlivých hráčov nemá na úspešnosť takmer alebo dokonca aj úplne žiadny vplyv. Zmena tohto parametra zapríčiní iba nárast strategického priestoru do šírky, ale pomer kontextov, v ktorých sa jednotlivé best-response stratégie prejavujú, k celkovému strategickému priestoru ostáva zachovaný, a teda ani neovplyvňuje výslednú pravdepodobnosť.

Pre popis správania v 4-hráčovej hre ($\forall i \in Q : |S_i| = 50 \wedge |BR_i| = 6$) sme zaviedli pojem stupeň užitočnosti best-response stratégie $b_i = BR_i(s_{-i})$ hráča i. V takejto hre stratégia môže b_i voči kontextu s_{-i} nadobúdať tri tieto stupne, podľa počtu hráčov $q \in Q : q \neq i$, pre ktorých platí:

$$s_{-i} = (s_{-i,-q}, s_q) : s_q \in BR_q((s_{-i,-q}, b_i))$$

V prípade ak je kontext s_{-i} tretieho stupňa užitočnosti, tvorí spolu so stratégiou b_i Nashovo ekvilibrium. Izolovaná best-response stratégia b_i na kontext s_{-i} druhého alebo tretieho stupňa je objaviteľná s pravdepodobnosťou blížiacou sa ku 100%, a to aj v prípade ak sa prejavuje iba v jednom kontexte s_{-i}. Úspešnosť detekcie užitočnej best-response stratégie prvého stupňa v 4-hráčovej bola degradovaná na úspešnosť nájdenia neužitočnej stratégie v 3-hráčovej hre. Z vyššie uvedeného je potom možné vyvodiť nasledovný záver. Izolovaná stratégia b_i hráča i, ktorá je best-response na kontext s_{-i}, je dobre odhaliteľná len v prípade, ak spolu s kontextom s_{-i} tvorí Nashovo ekvilibrium, alebo nanajvýš jeden hráč $q \in Q \wedge q \neq i$ s profilom (s_{-i}, b_i) nesúhlasí.

V 3-hráčovej hre 3.1 bola prvýkrát demonštrovaná možnosť neodhalenia izolovanej best-response stratégie algoritmom *FDDS* v kombinácii s *ISB*. V tomto konkrétnom prípade je táto stratégia zároveň aj súčasťou rýdzeho Nashovho ekvilibria, čo sa môže zdať v rozpore s predchádzajúcim tvrdením. Daná hra však bola skonštruovaná so zámerom, aby čo najmenej možných kombinácií uzlov ISB túto stratégiu detekovalo (menší počet už nie je možné



Obr. 5.5: Úspešnosť odhalenia izolovanej best-response vzhľadom na počet stratégií hráčov

dosiahnuť). Pre túto hru je možné vygenerovať celkovo 36 rôznych ISB, z toho 26 danú stratégiu a Nashovo ekvilibrium objaví. S nárastom počtu stratégií a vhodnou manipuláciou s preferenciami hráčov by mal tento pomer klesajúcú tendenciu.

U hier modelujúcich problémy reálneho sveta nie je podobná deformácia ako v nami generovaných hrách častou záležitosťou a naopak sa dá predpokladať určitá spojitosť stavového priestoru (napríklad model oligopolu). Takéto vlastnosti sú pre správnosť výstupu algoritmu *FDDS* iba prospešné a zvyšujú jeho význam v praktickom použití.

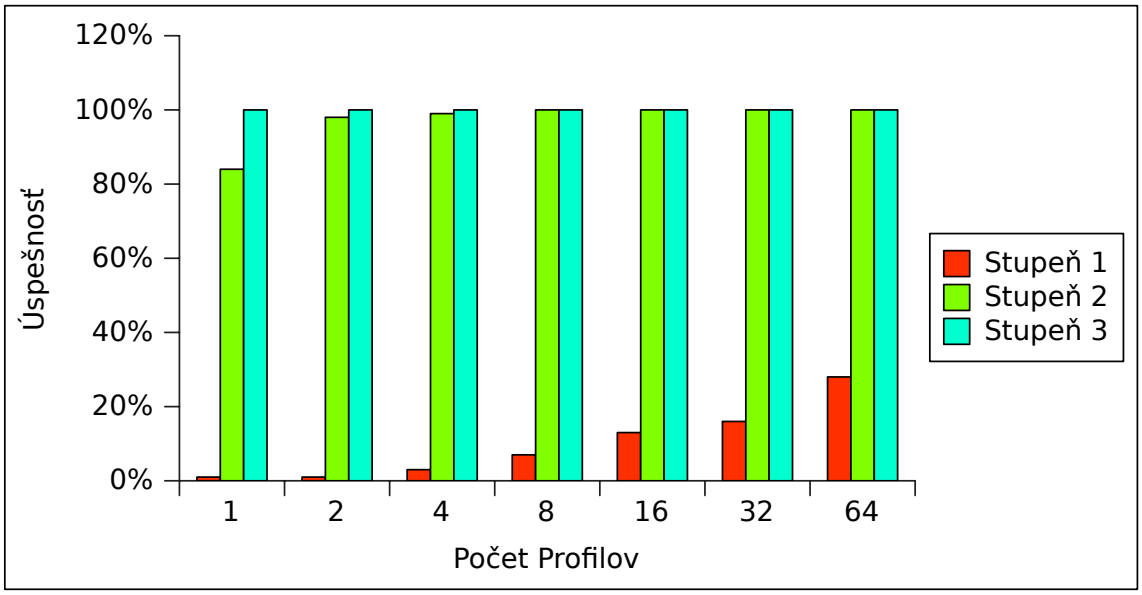
V poslednom teste z tejto série sa zameriame na rýdze Nashové ekvilibria, ktoré budu zložené len z izolovaných stratégií $s^* = (s^*_1, s^*_2, \ldots, s^*_{|Q|})$ takých, že $\forall i \in Q$:

$$\begin{aligned} s^*_i &\in BR_i(s^*_{-i}) \\ s^*_i &\notin BR_i(s_{-i}), \quad s_{-i} \neq s^*_{-i} \end{aligned}$$

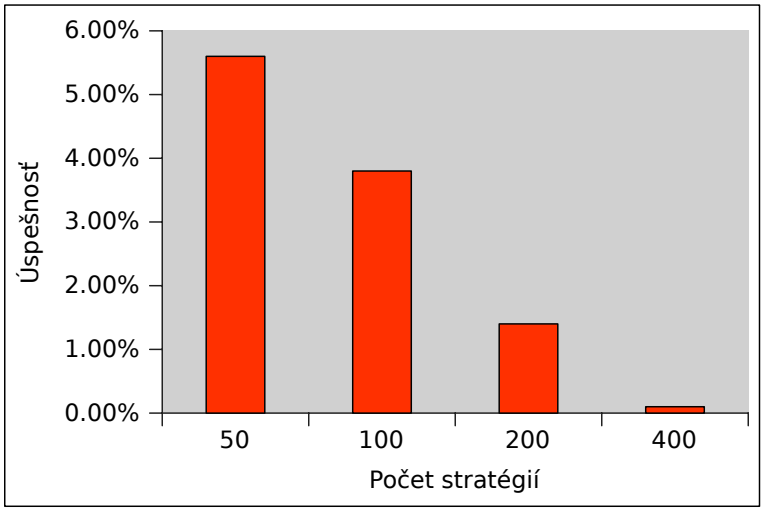
Určitý hráč i bude potom v kontexte svojho rozhodovania s^*_{-i} vždy hrať stratégiu s^*_i. Uvažujeme strategický profil $s = (s^*_{-i,-j}, s_i, s_j)$, kde $s_i \neq s^*_i \wedge s_j \neq s^*_j$. Dosiahnutie ekvilibria s^* z tohto strategického profilu by mohlo byť možné iba cez strategické profily (s^*_{-i}, s_i) respektíve (s^*_{-j}, s_j). Toto by sa dalo dosiahnuť iba ak by platilo $s^*_i \in BR_i((s^*_{-i,-j}, s_j))$ alebo $s^*_j \in BR_j((s^*_{-i,-j}, s_i))$, čo je v rozpore s vyššie uvedenou definíciou ekvilibria s^*. V dôsledku toho by k objaveniu takéhoto ekvilibria s^* algoritmom *FDDS* došlo iba v tedy ak by ISB obsahovala strategický profil s, ktorý by pre určitého hráča i a ľubovoľnú jeho stratégiu $s_i \in S_i$ splňoval:

$$s = (s^*_{-i}, s_i)$$

Pravdepodobnosť odhalenia takého ekvilibria by potom bola závislá len od pomeru počtu ISB, ktoré danú podmienku splňujú, k celkovému počtu kombinácii ISB. Následne bude táto pravdepodobnosť s rastúcou veľkosťou strategického priestoru klesať, čo podporujú aj naše dosiahnuté experimentálne výsledky (graf 5.7, $Q = 3$, $|BR_i| = 10$).



Obr. 5.6: Úspešnosť odhalenia izolovanej best-response v 4-hráčovej hre



Obr. 5.7: Úspešnosť odhalenia ekvilibria zložené z izolovaných best-response stratégií

5.5 Význam algoritmu *FDDS*

K množine ISB sme sa pokúsili vytvoriť inú alternatívu zaručujúcu odhalenie všetkých užitočných best-response stratégií. S ohľadom na tento cieľ, sa jedinou možnosťou javilo vygenerovanie množiny, ktorá by pre určitého hráča i obsahovala všetky kontexty jeho rozhodovania s_{-i}. Dôsledkom toho, by však bolo následné prehľadávanie celého strategického priestoru, čo je jednoznačne nežiadúcim javom popierajúcim samotný zmysel existencie metódy *FDDS*. Domnievame sa, že bez preskúmania celého strategického priestoru nie je možné v žiadnom prípade zaručiť zachovanie všetkých užitočných best-response stratégií vo výslednej zredukovanej hre. Pre nejakú množinu vstupných hier splňujúcich určité ob-

medzujúce podmienky, by to však reálne byť mohlo.

Aplikovaním algoritmu *FDDS* v kombinácii s ISB bude určitá stratégia $b_i \in BR_i(S)$ hráča i vždy objavená, ak pre nejakého hráča $j \in Q$, $i \neq j$ platí:

$$\exists s_j \in S_j : \forall s_{-i,-j} \in S_{-i,-j} \quad b_i \in BR_i(s_{-i,-j}, s_j)$$

Inak povedané, stratégia b_i bude zaručene objavená, ak je best-response stratégiou na každý profil, v ktorom hrá nejaký iný hráč j určitú svoju stratégiu s_j. Dalo by sa povedať, že b_i je best-response na s_j. Stratégiu splňujúcú definíciu s_j môžme nájsť napríklad v ukážke modelu oligopolu. Ak ľubovoľný z hráčov zvolí stratégiu väčšiu alebo rovnú dopytu M, best-response stratégiou ostatných hráčov bude vždy produkcia 0 jednotiek.

Platnosť vyššie uvedenej vety je zrejmá. ISB zaručuje existenciu každej stratégie každého hráča aspoň v jednom počiatočnom uzle. Následne pri riešení takéhoto uzla pre hráča i, bude jeho stratégia b_i patriť do množiny BR na tento uzol.

Kapitola 6

Záver

Matematická teória hier umožňuje modelovanie a následnú analýzu rôznych rozhodovacích problémov z reálneho sveta vo forme strategických hier v normálnej forme. Ich riešením je následne možné predikovať správanie modelovaného systému. Problémom naďalej ostáva aj napriek neustálemu technologickému pokroku príliš veľká časová a pamäťová zložitosť riešenia komplexnejších problémov konvenčnými metódami. K dispozícii sú rôzne teoretické metódy umožňujúce zjednodušenie rozsiahlych strategických hier, avšak existencia reálnych nástrojov použiteľných v praxi značne zaostáva. Práve metóda *FDDS*, ktorá bola hlavným objektom záujmu tejto diplomovej práce, má potenciál tento prázdny priestor vyplniť.

Samotným cieľom práce bola implementácia knižnice, umožňujúcej redukciu strategických hier v normálnej forme na ich best-response ekvivalenty. Tento cieľ sa podarilo naplniť vytvorením nástroja *FddsSolver*, ktorý pochopiteľným spôsobom demonštruje činnosť algoritmu *FDDS* v kombinácii s *ISB*. Tento nástroj bol následne experimentálne odskúšaný na niekoľkých sériách vhodne zvolených hier. Naša pozornosť bola venovaná najmä výpočtovej zložitosti, pričom sme skúmali jej závislosť na viacerých parametroch, konkrétne počtu hráčov, veľkosti množín stratégií a best-response stratégií, a rozlíšiteľnosti strategického priestoru. Z dosiahnutých výsledkov je možné konštatovať, že algoritmus *FDDS* potrebuje pre dosiahnutie korektného výstupu preskúmať len zlomok z celkového strategického priestoru skúmanej hry. Tento pomer sa navyše zvýrazňuje pri hrách s vyšším počtom hráčom.

V ďalších experimentoch bola vyvinutá snaha skúmať správnosť výstupu algoritmu. Domnievaná zaručená best-response ekvivalencia (aj vo variante užitočných best-response stratégií) bola vyvrátená na konkrétnom prípade hry. Na základe toho bol zavedený pojem izolovanej best-response stratégie, ktorej existencia je hlavnou príčinou tohto javu. Následne boli umelo generovaných hier zakomponované práve takéto stratégie a predmetom záujmu bolo ich správanie vzhľadom na rôzne parametre hry. Aj v dôsledku týchto experimentov sme formulovali vetu (aj s odôvodnením), ktorá hovorí, že algoritmus *FDDS* je schopný odhaliť Nashové ekvilibria tvorené výlučne z izolovaných stratégií, len ak sú priamo obsiahnuté v *ISB*. Pre takéto ekvilibria by bola táto metóda prakticky degradovaná na náhodné prehľadávanie strategického priestoru. V závere bola formulovaná definícia stratégií, ktoré sú použitím tohoto algoritmu zaručene odhalitelné.

Výsledkom tejto diplomovej práce je nástroj *FddsSolver*, ktorý zrozumiteľným spôsobom implementuje metódu *FDDS*. Ďalšími výstupmi sú sada testovacích príkladov a generátor hier umožňujúci parametrizáciu niektorých ich vlastností. Z hľadiska možných rozšírení a vývoja do budúcnosti poskytuje skúmaná metóda rôzne príležitosti. Niektorými z nich sú hlbšia analýza *GRP* a s ňou spojené viaceré výstupy, alebo návrh inej počiatočnej množiny uzlov algoritmu. Aj keď metóda *FDDS* nezaručuje ekvivalenciu medzi obecnou vstupnou

hrou v normálnej forme a zredukovaným výstupom, formulovaním istých obmedzujúcich podmienok pre vstupnú hru by zaručenie ekvivalencie mohlo byť dosiahnuté. Táto oblasť predstavuje ďalšie možnosti budúceho vývoja, rozhodne sa však nejedná o triviálny problém.

Literatúra

[1] Brandenburger, A.: Nash Equilibrium: Definition. April 2007, http://pages.stern.nyu.edu/~abranden/nash-01-04-07.pdf.

[2] Hrubý, M.: Algorithmic Approaches to Game-theoretical Modeling and Simulation. *AUCO Czech Economic Review*, ročník 2, č. 3, December 2008: s. 268–300.

[3] Myerson, R. B.: *Game Theory: Analysis of Conflict*. First Harvard University Press, 1997, iSBN 0-674-34116-5.

[4] Möbius, M. M.: Lecture III: Normal Form Games, Rationality and Iterated Deletion of Dominated Strategies. February 2007, http://isites.harvard.edu/fs/docs/icb.topic138342.files/lecture3.pdf.

[5] Nash, J.: Non-cooperative games. *The Annals of Mathematics, Second Series*, ročník 54, č. 2, September 1951: s. 286–295, http://www.jstor.org/stable/1969529.

[6] Nissan, N.: *Algoritmic game theory*. Cambridge University Press, 2007, iSBN 978-0-521-87282-9.

[7] Osborne, M. J.: *Selected chapters from draft of An Introduction to Game Theory*. Oxford University Press, 2000, http://home.cerge-ei.cz/kalovcova/files/VSE_GT_S2010/Osborne.pdf.

[8] Papadimitriou, C. H.; Roughgarden, T.: Computing Correlated Equilibria in Multi-Player Games. January 2008, http://theory.stanford.edu/~tim/papers/cor.pdf.

[9] Rasmusen, E.: *Games and information: An introduction to game theory*. Blackwell Publishing, 2007, iSBN 10: 1-4051-3666-9.

[10] Židek, S.: *Implementace algoritmů Teorie her*. diplomová práce, FIT VUT v Brně, 2009.