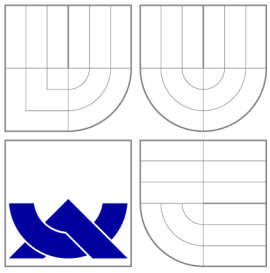
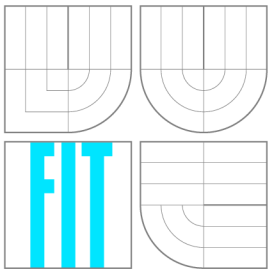


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

NÁSTROJ PRO VÝPOČET NASHOVA EKVILIBRIA V NEKOOPERATIVNÍCH HRÁCH S NENULOVÝM SOUČTEM

A TOOL FOR COMPUTING NASH EQUILIBRIA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR ŠEBEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN HRUBÝ, Ph.D.

BRNO 2013

Abstrakt

Tato práce se zabývá popisem vývoje nástroje pro výpočet Nashova ekvilibria v nekooperativních hrách s nenulovým součtem. Definuje základní pojmy v teorii nekooperativních her. Popisuje vhodné algoritmy pro výpočet ryzího a smíšeného Nashova ekvilibria podle počtu hráčů dané hry. Práce prezentuje implementaci výsledné aplikace a experimenty na ní provedené.

Abstract

This thesis deals with development of tool for computing Nash equilibrium in non-zero-sum non-cooperative games. It defines basic terms in non-cooperative game theory. It describes suitable algorithms for computation pure and mixed Nash equilibrium according to number of players. Thesis presents implementation of resulting application and experiments conducted on it.

Klíčová slova

Teorie her, výpočet Nashova ekvilibria, ryzí Nashovo ekvilibrium, smíšené Nashovo ekvilibrium, optimalizace Lyapunovy funkce, CMA-ES

Keywords

Game theory, Nash equilibrium computation, pure Nash equilibrium, mixed Nash equilibrium, Lyapunov function optimalization, CMA-ES

Citace

Petr Šebek: Nástroj pro výpočet Nashova ekvilibria v nekooperativních hrách s nenulovým součtem, bakalářská práce, Brno, FIT VUT v Brně, 2013

Nástroj pro výpočet Nashova ekvilibria v nekooperativních hrách s nenulovým součtem

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.

.....
Petr Šebek
13. května 2013

Poděkování

Rád bych poděkoval Ing. Martinu Hrubému, Ph.D. za odbornou pomoc a cenné rady při vypracovávání této práce.

Obsah

Kapitola 1

Úvod

V této práci se budu zabývat teorií her a hlavně její podkapitolou nekooperativní hry v normální formě. Pomocí teorie her lze matematickým aparátem popsat racionální chování lidí v situaci, kdy se navzájem ovlivňují svými rozhodnutími. Samotná teorie her je multidisciplinární vědou, nejvíce ovlivňuje vědecké odvětví jako matematika, ekonomie, politologie a sociologie. Vlastní teorie je aplikovatelná na každodenní rozhodování.

Za první dílo, jež se věnuje popisu teorie her je považována kniha *Theory of games and economic behavior* od Johna von Neumanna [19]. Pro tuto práci je však důležitější článek Johna Nashe *Non-Cooperative Games* [13]. Nash definoval hry nekooperativní pro více hráčů, zatímco ve von Neumannově práci byla rozebrána pouze teorie dvouhráčových her s nulovým součtem a vícehráčové kooperativní hry. Nash ve svém článku řeší hry pomocí ekvilibria neboli rovnováhy ve hře. Rovnováhu si vysvětlujeme tak, že žádný hráč hrající hru nebude mít tendenci měnit svou strategii, a tím měnit i strategie protihráčů, protože každý svou volbou odpovídá na akce protivníků. Na počest Nashova objevu byla tato rovnováha nazvána Nashovým ekvilibriem (angl. Nash equilibrium).

Tato práce se zabývá vývojem nástroje pro výpočet Nashova ekvilibria, který má nalézt všechna ryzí Nashova ekvilibria, všechna smíšená Nashova ekvilibria v dvouhráčových hrách a alespoň jedno smíšené Nashovo ekvilibrium ve hrách vícehráčových.

V kapitole *Teorie her* (2) bude popsána teorie her s ohledem na nekooperativní hry v rozsahu potřebném pro tuto práci. Kapitola *Algoritmy pro výpočet Nashova ekvilibria* (3) popisuje použité algoritmy pro nalezení tohoto rovnovážného bodu ve hře. Následující kapitola *Implementace* (4) nám umožní představu o implementaci nástroje pro výpočet Nashova ekvilibria. Poslední kapitola *Experimenty* (5) ukazuje použití vytvořeného nástroje v praxi při počítaní s hrami o různé náročnosti a srovnání algoritmu s konkurenčními prostředky.

Bakalářská práce navazuje na výsledky semestrálního projektu z předmětu ISP. V tomto projektu jsem splnil první dva body zadání, tzn. nastudoval jsem teorii her se zaměřením na nekooperativní hry, provedl první návrh programu a stanovil algoritmy pro výpočet Nashova ekvilibria.

Kapitola 2

Teorie her

V následujících sekcích zaměřených na samotnou teorii jsem vycházel především z publikací *Doprovodné texty ke kurzu Teorie her* od Martina Hrubého [6], *Course in game theory* od autorů Martin J. Osborne a Ariel Rubinstein [17] a *Algorithmic game theory*, kterou z editoval Noam Nisan [14]. Notace je plně převzata z [6].

2.1 Nekooperativní hra

Nekooperativní hru si lze představit jako interaktivní strategickou situaci dvou a více protihráčů, kdy každý hráč sleduje pouze své vlastní cíle a projevuje tak tzv. individuální racionalitu. Teoretický model u hráčů předpokládá následující vlastnosti[17, str. 1]:

- Jsou racionální, tedy chovají se jen podle vlastních zájmů.
- Očekávají racionální hru svých protihráčů.

Hráči mají možnost vybírat z určitých akcí nebo *strategií* a jsou si vědomi důsledků, pokud budou hrát tyto strategie. Tyto důsledky jsou nazývány *užitky*. Právě vědomí si svých strategií, strategií protihráčů a všech užitků je jedním ze základních kamenů her a nazývá se *společná znalost* (angl. common knowledge).

Hra je potom hrána jako jedna ojedinělá a neopakující se událost. Každý hráč se rozhodne pro určitou strategii bez znalosti zvolených strategií ostatních hráčů. Samozřejmě takový hráč očekává, že i ostatní hráči jsou racionální, a proto může předpokládat akce protihráčů. Hráčům je sdělen výsledek hry, a tedy i jejich užitek. Hra tímto končí.

Nyní můžeme přistoupit k matematické definici nekooperativní hry a všeho, co je k ní potřeba.

Definice 1. Strategická nekooperativní hra N hráčů je $(2N+1)$-tice

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

kde:

- $Q = 1, 2, \ldots, N$ je konečná množina hráčů ve hře.
- $S_i, i \in Q$ jsou (konečné) množiny ryzích strategií hráčů $i \in Q$.
- $U_i : S_1 \times S_2 \times \ldots \times S_N \to \mathbb{U}$ jsou funkce užitku hráčů.

Oborem hodnot užitkových funkcí je obecně univerzum $\mathbb{U}$. Prakticky se však většinou za toto univerzum berou reálné čísla $\mathbb{U} = \mathbb{R}$.

Kartézský součin množin strategií hráčů je nazýván *množinou strategických profilů hry*:

$$S = S_1 \times S_2 \times \ldots \times S_N = \prod_{i \in Q} S_i$$

Tato množina je tedy množinou všech možných výstupů, které mohou hráči hrát. Prvky se nazývají *strategické profily* $s \in S$.

Množina sub-profilů hráče i je kartézský součin všech množin strategií kromě množiny hráčovy:

$$S_{-i} = S_1 \times S_2 \times \ldots \times S_{i-1} \times S_{i+1} \times \ldots \times S_N = \prod_{j \in Q \setminus \{i\}} S_j$$

Složení strategie $s_i \in S_i$ a kontextu protihráčů $s_{-i} \in S_{-i}$ značíme následujícím způsobem: $(s_i, s_{-i}) \in S$.

Počet strategií hráče i budeme značit:

$$m_i = |S_i|$$

Počet všech strategií všech hráčů pak můžeme vyjádřit pomocí:

$$m = \sum_{i \in Q} m_i \tag{2.1}$$

2.1.1 Best response

Nyní si definujeme pojem nejlepší odpovědi na protihráčovi akce. Předpokládejme, že protihráči zvolí strategický sub-profil s_{-i}, potom se hráč i snaží vybrat ze svých strategií akci, která bude pro něho nejpřínosnější. Množina strategií, jejíž prvky jsou nejlepší možnou odpovědí na akce protihráčů, se nazývá *best response*.

Definice 2. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$, hráče $i \in Q$ a strategický subprofil $s_{-i} \in S_{-i}$. Best response hráče i potom je následující korespondence:

$$BR_i(s_{-i}) = arg \max_{s_i \in S_i} [U_i(s_i, s_{-i})]$$

Z definice (2) plyne, že oborem hodnot BR_i je $2^{S_i} \setminus \{\emptyset\}$.

Zde je důležité upozornit, že racionální hráč si vždy vybere strategii z množiny BR_i. To je samozřejmě nutné brát v úvahu jako protihráč. Zároveň je také indiferentní mezi všemi strategiemi v množině BR_i.

2.1.2 Dominance mezi strategiemi

Na druhou stranu, pokud má některá strategie vždy menší užitek než ostatní, nebude tuto strategii hráč nikdy hrát a nemusí ji tedy ani uvažovat. Protihráči jsou si vědomi toho, že tato strategie nikdy hrána nebude, a proto ji také neuvažují. Taková strategie je nazývána dominovaná (angl. dominated).

Definice 3. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$, strategie $s_i^1 \in S_i$ **striktně** dominuje nad strategií $s_i^2 \in S_i$, pokud:

$$\forall s_{-i} \in S_{-i} : U_i(s_i^1, s_{-i}) > U_i(s_i^2, s_{-i})$$

Existuje ještě druhý koncept dominance mezi strategiemi, a to slabá dominance.

Definice 4. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$, strategie $s_i^1 \in S_i$ **slabě** dominuje nad strategií $s_i^2 \in S_i$, pokud:

$$\forall s_{-i} \in S_{-i} : U_i(s_i^1, s_{-i}) \geq U_i(s_i^2, s_{-i})$$

A současně

$$\exists s'_{-i} \in S_{-i} : U_i(s_i^1, s'_{-i}) > U_i(s_i^2, s'_{-i})$$

Pokud budeme dominanci strategie zkoumat oproti všem ostatním hráčovým strategiím, zjistíme, zda je strategie celkově dominující/dominovaná.

Definice 5. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$. Strategie s_i^1 hráče i je ve hře striktně dominovaná, pokud existuje jedna strategie $s_i^2 \in S_i \setminus \{s_i^1\}$ taková, že s_i^2 striktně dominuje nad s_i^1.

Definice 6. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$. Strategie s_i^1 hráče i je ve hře striktně dominující, pokud pro všechny strategie $s_i^2 \in S_i \setminus \{s_i^1\}$ platí, že s_i^1 striktně dominuje nad s_i^2.

Pokud má každý hráč ze hry mezi svými strategiemi jednu strategii striktně dominující nade všemi, získáváme první možné řešení nekooperativních her: řešení v dominujících strategiích (angl. dominant strategy solution [14, str. 10] / dominant strategy equilibrium [17, str. 181]).

Definice 7. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$. Strategický profil $s^* \in S$ je řešením ve striktně dominantních strategiích, pokud platí:

$$\forall i \in Q,\ s_i \in S_i \setminus \{s^*\} : U_i(s_i^*, s_{-i}) \geq U_i(s_i, s_{-i})$$

Existuje ještě pojem dominance smíšené strategie nad ryzí strategií. Tuto však pro vysokou výpočetní náročnost nebudeme uvažovat.

2.2 Ryzí Nashovo ekvilibrium

Ekvilibrium neboli rovnováha značí ve hrách rovnovážný stav. Tento bod by měl popisovat situaci hry, která je únosná pro všechny hráče a především, že racionální hráči svým uvažováním do tohoto bodu sami dospějí. Tuto skutečnost také do jisté míry dokazují experimentální studie s chováním lidí v reálných situacích jako například chování tenistů na Wimbledonu od Marka Walkera [20] a chování fotbalistů a brankářů při kopání penalty, jak vysledoval Piere-André Chiappori [2].

Tato práce se bude zabývat výhradně Nashovým ekvilibriem. Ekvilibrium můžeme považovat za určité řešení hry. Je to informace, jejíž znalost může být velmi důležitá, ale na druhou stranu její získání patří mezi obtížné výpočetní úkony, jak je popsáno v článku *The Complexity of Computing a Nash Equilibrium* [3].

Definice 8. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$. Strategický profil $s^* \in S$ je **ryzím Nashovým ekvilibriem** (angl. pure Nash equilibrium), pokud platí:

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

V literatuře se tato definice interpretuje tak, že žádný z hráčů nemůže změnit svoji ryzí strategii, a tím si zvýšit užitek ze hry [17, 14]. Z definic (8) a (7) vyplývá, že pokud má hra řešení ve striktně dominantních strategiích je toto řešení zároveň unikátním Nashovým ekvilibriem.

2.3 Nekooperativní hra se smíšenými strategiemi

Jak nám ukazují hry *Matching pennies* (A.1) nebo *Kámen-nůžky-papír* (A.2), ne všechny hry mají ryzí Nashovo ekvilibrium. Znamená to, že hra nemá ryzí Nashovo ekvilibrium, ale že každý hráč hraje nedeterministicky, tedy s určitým vlivem pravděpodobnosti. Takové hráčovo chování se nazývá *smíšenou strategií*. Existence smíšených strategií u hry nazýváme *smíšeným rozšířením* [17, str. 32] nekooperativní hry.

Definice 9. Mějme hru $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$. Hru $\Gamma^m = (Q; \{\Delta_i\}_{i \in Q}; \{\pi_i\}_{i \in Q})$ nazveme smíšeným rozšířením hry Γ, pokud $\forall i \in Q$:

- Δ_i je množina smíšených strategií hráče i s prvky $\sigma_i \in \Delta_i$. Číslo $\sigma_i(s_i)$ označuje pravděpodobnost přiřazenou ryzí strategii $s_i \in S_i$ ve smíšené strategii σ_i. Množinu všech množin smíšených strategií značíme $\Delta = \prod_i \Delta_i$.

$$\Delta_i = \left\{ \sigma_i \in \langle 0, 1 \rangle^{m_i} | \sum_{s_j \in S_i} (\sigma_{ij} = \sigma_i(s_j)) = 1 \right\} \tag{2.2}$$

- Výplatní funkce hráčů / očekávaný užitek (angl. expected payoff) je v každém smíšeném profilu $\sigma \in \Delta$:

$$\pi_i(\sigma) = \sum_{s \in S} U_i(s) \cdot \left(\prod_{i \in Q} \sigma_i(s_i) \right)$$

V dalším textu budeme potřebovat složení ryzí strategie se strategií smíšenou. To budeme pro jednoduchost značit (s_i, σ_{-i}), kde $i \in Q$, $s_i \in S_i$ a $\sigma_{-i} \in \Delta_{-i}$. Tento zápis znamená, že smíšená strategie se skládá ze strategií protihráčů a jedné ryzí strategie hráče i.

2.3.1 Smíšené Nashovo ekvilibrium

Ve smíšeném rozšíření nekooperativní hry samozřejmě existuje také koncept Nashova ekvilibria. Ukazuje nám stabilní rozložení pravděpodobnosti přes hráčovy strategie po dosáhnutí stavu, kdy žádný z hráčů nemá zájem tuto smíšenou strategii měnit. Musíme však stále mít na paměti, že nekooperativní hra je neopakovatelnou a neopakovanou událostí a celý proces hledání ekvilibria probíhá před vlastním rozhodnutím o hrané strategii. Smíšené Nashovo ekvilibrium je možno vysledovat na velkém vzorku nezávislých pokusů [2, 20].

Definice 10. Mějme hru $\Gamma^m = (Q; \{\Delta_i\}_{i \in Q}; \{\pi_i\}_{i \in Q})$. Smíšeny profil $\sigma^* \in \Delta$ nazveme smíšené Nashovo ekvilibrium ve hře Γ^m, pokud platí:

$$\forall i \in Q,\ \forall \sigma_i \in \Delta_i : \pi_i(\sigma^*) \geq \pi_i(\sigma_i, \sigma_{-i}^*)$$

Podobně jako u ryzího Nashova ekvilibria si tento vzorec můžeme vyložit tak, že žádný hráč nemá tendenci změnit svoji smíšenou strategii.

Důležitý pojem, který vystává u otázky smíšeného Nashova ekvilibria a smíšených strategií obecně, je *indiference*. Hráč, který hraje smíšenou strategii, musí mít stejný užitek z každé jednotlivé ryzí strategie v doméně (2.3.3) této strategie, tedy z každé ryzí strategie hrané s nenulovou pravděpodobností [17, str. 33]. To je samozřejmé, kdyby měl z jedné strategie užitek větší než ze druhé, tak to je jeho best response a druhou strategii nebude vůbec hrát.

2.3.2 Existence smíšeného Nashova ekvilibria

John Nash ve svém článku *Non-cooperative games* [13] vyslovil následující větu:

Věta 1. *Každá konečná nekooperativní hra má Nashovo equilibrium.*

Důkaz této věty je možno nalézt v [13, str. 5]. Věta 1 nám ukazuje, že v každé hře, se kterou se budeme v této práci zabývat, bude minimálně jedno Nashovo ekvilibrium, ať už ryzí nebo smíšené. Tím máme vymezen teoretický základ pro hledání Nashových ekvilibrií v nekooperativních hrách.

Dalším zajímavou vlastností nekooperativních her je počet ekvilibrií.

Věta 2. *Každá konečná nedegenerovaná hra má vždy lichý počet ekvilibrií [14, str. 62].*

2.3.3 Doména smíšené strategie

Doména smíšené strategie (angl. support) je pojem, díky kterému se nám bude lépe pracovat se smíšenými strategiemi při hledání smíšeného Nashova ekvilibria. Také je díky ní definována degenerovanost hry (kapitola 2.3.4).

Definice 11. Mějme hru $\Gamma^m = (Q; \{\Delta_i\}_{i \in Q}; \{\pi_i\}_{i \in Q})$ a smíšenou strategii $\sigma_i \in \Delta_i$ hráče i. Doménu smíšené strategie hráče i definujeme takto:

$$supp_i(\sigma_i) = \{s_i \in S_i | \sigma_i(s_i) > 0\}$$

Doména smíšené strategie obsahuje všechny ryzí strategie hrané s nenulovou pravděpodobností ve smíšené strategii σ_i.

2.3.4 Degenerovaná hra

U her je zavedený i další pojem *degenerovanost*, který nám dává určitou informaci o smíšených Nashových ekvilibriích.

Definice 12. Dvouhráčová hra je *nedegenerovaná* (angl. nondegenerate), pokud žádná smíšená strategie s doménou velikosti k nemá více než k ryzích best response strategií.

Jednoduchým testem na degenerovanost hry je výpočet ryzích best-response na všechny ryzí strategie protihráče. Pokud hráč na jednu ryzí strategii protihráče nalezne více než jednu best response, je tato hra degenerovaná [14, str. 56]. Z této nutné rovnosti počtu best response nám vyplývá následující věta:

Věta 3. *V jakémkoli Nashovu ekvilibriu nedegenerované dvouhráčové hry mají smíšené strategie toho ekvilibria domény stejné délky. [14, str. 56]*

Důležitou vlastností degenerovaných her je, že mohou mít nekonečně mnoho řešení ve smíšených strategiích [14, str. 66].

Za příklad si můžeme vzít degenerovanou hru z přílohy (A.3). Zde je vidět, že na strategii 0 řádkového hráče má sloupcový hráč hned dvě ryzí best response: 0 a 1. Tyto strategie jsou zároveň doménami ve smíšeném Nashovu ekvilibriu. Tedy řádkový hráč hraje strategii 0 s pravděpodobností $1, 0$ a sloupcový hráč má libovolný výběr rozložení pravděpodobnosti přes jeho jediné dvě strategie 0 a 1.

Kapitola 3

Algoritmy pro výpočet Nashova ekvilibria

V této kapitole si definujeme a popíšeme algoritmy pro nalezení Nashova ekvilibria. Nejprve si představíme algoritmus pro hledání ryzích Nashových ekvilibrií hrubou silou, poté algoritmus pro nalezení smíšeného Nashova ekvilibria ve dvouhráčových hrách a nakonec algoritmus pro nalezení smíšeného Nashova ekvilibria ve hrách vícehráčových.

Přehled jednotlivých algoritmů a přístupů k řešení nekooperativních her lze nalézt v *Computation of equilibria in finite games* [11]. V této práci jsou popsány algoritmy pro řešení dvouhráčových a vícehráčových her. Za základní algoritmy se tu považují algoritmus Lemke-Howson [9] pro dvouhráčové hry a Simplical subdivision pro hry vícehráčové. Také je představena metoda minimalizace funce Lyapunovy funkce [10], která bude použita jako stěžejní algoritmus této práce spolu s metodou minimalizace funkce Covariance Matrix Adaptation - Evolution Strategy [5].

3.1 Nalezení ryzího Nashova ekvilibria hrubou silou

Pokud si za úkol stanovíme nalezení pouze ryzího Nashova ekvilibria v obecně n-hráčové hře, budeme vycházet z definic best response (2) a ryzího Nashova ekvilibria (8). Požadavkem pro existenci Nashova ekvilibria je, aby žádný hráč neměl nutkání měnit svou ryzí strategii na strategii jinou. Jinými slovy si každý hráč vybírá nejlepší možnou strategii s ohledem na akce soupeřů. To je však definice best response. Můžeme tedy říci, že Nashovo ekvilibrium je takovým strategickým profilem, kde všichni hráči hrají svoji best response [17, str. 15]. Zapsáno algoritmem:

Algoritmus 1 Nalezení Nashova ekvilibria hrubou silou

Input: $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$
  for all $i \in Q$ **do**
    for all $s_{-i} \in S_{-i}$ **do**
      $BRpne_i \leftarrow (BR_i(s_{-i}), s_{-i})$
    end for
  end for
  $PNE \leftarrow \bigcap_{i \in Q} BRpne_i$
Output: PNE ▷ ryzí Nashova ekvilibria

Algoritmus tedy zjistí best response všech hráčů na všechny možné sub-profily protihráčů. Každý hráč má potom množinu strategických profilů, kde jeho akce je jeho best response. Vytvořením průniku těchto množin dostaneme množinu profilů, které jsou ryzími Nashovými ekvilibrii.

3.2 Iterativní eliminace dominovaných strategií

V sekci (2.1.2) jsme si představili řešení v dominujících strategiích. Tato řešení však jsou vzácná. Co ale při řešení výpočtu Nashova ekvilibria použít můžeme je znalost, že racionální hráči nebudou nikdy hrát striktně dominované strategie. Pokud všechny takové strategie ze hry vyřadíme, hra bude ekvivalentní (všichni racionální hráči projevují stejné strategické chování) a zároveň zjednodušíme složitost dané hry. Také je možnost, že danou hru zjednodušíme až na stav, kdy každý hráč má jen jednu nedominovanou strategii a tím nalezneme i unikátní Nashovo ekvilibrium.

Algoritmus nejdříve zjistí, zda hra obsahuje striktně dominované strategie. Pokud ano, eliminuje je ze hry. Může se stát, že odstraněním strategie jednoho hráče se stane dominovanou strategií strategie hráče druhého, proto tento algoritmus stále testuje, zda hra neobsahuje striktně dominované strategie, a pokud ano, tak je odstraňuje.

Algoritmus 2 Iterativní eliminace dominovaných strategií

Input: $\Gamma = (Q; \{S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$
  $dominatedStrategies \leftarrow getDominatedStrategies()$
  while $dominatedStrategies \neq \emptyset$ **do**
    Smaž dominované strategie ze hry
    $dominatedStrategies \leftarrow getDominatedStrategies()$
  end while
Output: $\Gamma = (Q; \{S'_i \subseteq S_i\}_{i \in Q}; \{U_i\}_{i \in Q})$ ▷ hra bez striktně dominovaných strategií

Funkce `getDominatedStrategies()` vrací striktně dominované strategie podle definice (5).

Tento algoritmus vede k nalezení Nashova ekvilibria například u hry *Vězňovo dilema* (A.4).

3.3 Algoritmus výpočtu smíšeného Nashova ekvilibria ve dvouhráčových hrách

Při hledání smíšeného Nashova ekvilibria už nevystačíme s algoritmem (1). Zde je potřeba vypočítat rozložení pravděpodobnosti přes hráčovy strategie. Pro hry o dvou hráčích však můžeme použít algoritmus *Vyčíslení domén* (angl. Support enumeration [14, str. 56]). Tento algoritmus počítá pouze s nedegenerovanými dvouhráčovými hrami.

V algoritmu vyčíslení domén se vychází z následujících předpokladů:

- Vstupní hra je nedegenerovaná.
- Hledané Nashovo ekvilibrium bude mít domény stejné délky (věta 3).
- Hráč musí být mezi všemi svými strategiemi v doméně indiferentní, tedy musí z nich mít stejný užitek.

- Každá smíšená strategie musí být best response na smíšený sub-profil protivníků. Pokud by toto neplatilo, mohlo by se stát, že nalezneme řešení, ale to bude platit jen pro právě počítané strategie a ne pro celou hru.

Algoritmus 3 Vyčíslení domén

Input: $\Gamma^m = (Q; \{\Delta_i\}_{i \in Q}; \{\pi_i\}_{i \in Q}), N = |Q| = 2, m_i = |S_i|$
  for $k = 1, ..., \min(m_1, m_2)$ **do**
    for all $(I, J) : I \subseteq S_1, J \subseteq S_2, |I| = |J| = k$ **do**
      $\sum_{i \in I} x_i U_2(s_i, s_j) = v$, for $j \in J$
      $\sum_{i \in I} x_i = 1$
      $\sum_{j \in J} y_j U_1(s_i, s_j) = u$ for $i \in I$
      $\sum_{j \in J} y_j = 1$
      if $x \geq \mathbf{0}$ and $y \geq \mathbf{0}$ and x je best response pro y and y je best reponse pro x **then**
        $MNE \leftarrow (x, y)$ ▷ x a y je v kontextu ostatních nulových strategií
      end if
    end for
  end for
Output: $MNE \subseteq \Delta$ ▷ smíšená Nashova ekvilibria

V případě, že lineární rovnice v tomto algoritmu nemají řešení, není hledané Nashovo ekvilibrium v podmnožině strategií, a musí se proto hledat dál. Výstupem algoritmu jsou všechna Nashova ekvilibria této hry.

Pro odvození výpočetní složitosti algoritmu Vyčíslení domén (3) vyjdeme z článku *Computing equilibria in bimatrix games by parallel support enumeration* [21]. Za předpokladu, že $m_1 > m_2$, tedy že počet ryzích strategií hráče prvního je větší než počet strategií hráče druhého, budeme muset vypočítat $\binom{m_1+m_2}{m_2} - 1$ soustav rovnic. Výpočetní složitost řešení systému lineárních rovnic je $O((m_1 + m_2)^3)$. Výpočetní složitost algoritmu Vyčíslení domén (3) je

$$O\left((m_1 + m_2)^3 \binom{m_1 + m_2}{m_2}\right) \tag{3.1}$$

3.4 Algoritmus pro nalezení NE ve vícehráčových hrách

Hlavním a nejsložitějším úkolem této práce je nalézt alespoň jedno Nashovo ekvilibrium ve vícehráčových nekooperativních hrách (t.j. $N > 2$). Tady už nelze jednoduše použít algoritmus Vyčíslení domén (3), protože by bylo nutno počítat soustavy nelineárních rovnic a algoritmus by tak byl mnohem složitější než pro dvouhráčové hry.

Jak už bylo zmíněno: za standard se při výpočtu Nashových ekvilibrií ve vícehráčových hrách považuje algoritmus *Simplicial subdivision*. V této práci byla však použita více experimentální metoda, a to minimalizace Lyapunovy funkce (kapitola 3.4.1). Pro lepší kontrolu běhu programu a pro experimentální účely je tato funkce minimalizována pomocí evolučního algoritmu *Adaptace kovarianční matice - evoluční strategie* (angl. Covariance Matrix Adaptation - Evolution Strategy, CMA-ES, kapitola 3.4.2).

Pokud budeme brát počet výpočtů Lyapunovy funkce za parametr rozhodující o náročnosti algoritmu, pak byla metoda CMA-ES mezi dalšími inteligentními metodami (optimalizace hejnem částic - angl. particle swarm optimization, diferenciální evoluce - angl.

differential evolution) vyhodnocena jako nejšetrnější v testu v článku *Computing Nash equilibria through computational intelligence methods* [18]. V této práci ji budeme porovnávat s jinými minimalizačními metodami.

Na CMA-ES také stále probíhá aktivní vývoj a její autor, Nikolaus Hansen, ji stále zpřesňuje a přidává nové vlastnosti. Je také autorem přepisů této metody do mnoha programovacích jazyků. Metoda CMA-ES se prokazuje být velice úspěšnou pro tzv. *black-box* optimalizaci, navíc přes svou relativní novost byla již mnohokrát aplikována ve vědeckém výzkumu [4].

3.4.1 Lyapunova funkce

Lyapunova funkce se používá pro prokázání stability ekvilibria obyčejných diferenciálních rovnic. V této práci ji použijeme jako objektivní funkci, jejíž optimalizací získáme smíšené Nashovo ekvilibrium. Při její definici budeme vycházet z článku *A Laipunov Function for Nash Equilibria*[1] od Richarda D. McKelveyho [10].

Definice 13. Mějme hru $\Gamma^m = (Q; \{\Delta_i\}_{i\in Q}; \{\pi_i\}_{i\in Q})$ a smíšenou strategii $\sigma \in \Delta$. Lyapunova funkce $v : \Delta \to \mathbb{R}$ je definována jako:

$$v(\sigma) = \sum_{i\in Q} \sum_{s_i \in S_i} \{\max\left[U_i(s_i, \sigma_{-i}) - U_i(\sigma), 0\right]\}^2 \tag{3.2}$$

Interpretovat tuto funkci lze takto: pokud existuje nějaká ryzí strategie, kterou hráč i dosáhne lepšího užitku než z vlastního strategického subprofilu σ, je funkce nenulová. Pokud pro všechny hráče žádná taková ryzí strategie neexistuje funkce se rovná nule. Lyapunovy funkce jsou třídou funkcí splňující určité podmínky. Pro účely zjednodušení budu nadále v textu označovat jako Lyapunovu funkci pouze funkci v z definice (13).

Lyapunova funkce není lineární, ale pro účely numerického výpočtu je postačující.

Věta 4. *Mějme hru $\Gamma^m = (Q; \{\Delta_i\}_{i\in Q}; \{\pi_i\}_{i\in Q})$ a smíšenou strategii $\sigma \in \Delta$. Pokud platí $v(\sigma) = 0$, je σ smíšeným Nashovým ekvilibriem [10, str. 2].*

Důkaz této věty můžeme najít v McKelveyho článku [10, str. 3].

Zde je třeba upozornit, že obecná Lyapunova funkce je definována na $\mathbb{R}^m$. My však hledáme bod z prostoru $\Delta \subset \mathbb{R}^m$. Mimo tento prostor nejsou operace se hrou definovány, například nemůžeme spočítat užitek ze hry hráče i, pokud strategický profil obsahuje zápornou pravděpodobnost hraní ryzí strategie $\sigma_i(s_j) < 0$ pro $\sigma_i \in \Delta_i$ a $s_j \in S_i$.

Pro větší explicitnost si tedy definujeme vektor $\mathbf{p}_i \in \mathbb{R}^{m_i}$, pro každého hráče i, jehož prvky můžeme zapsat jako $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{im_i})$. Vektor $\mathbf{p}$ je potom definován takto $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N)$.

Abychom zajistili, že $\mathbf{p}$ bude patřit do Δ, opatříme Lyapunovu funkci ještě penalizací (rovnice 3.3), pro body, které jsou mimo prostor Δ. Tento přístup je ostatně popsán i v tutoriálu pro metodu CMA-ES [5, str. 29].[2]

$$w(\mathbf{p}) = v(\mathbf{p}) + \sum_{i\in Q} \sum_{s_j \in S_i} \{\min[p_{ij}, 0] + \max[p_{ij}, 1] - 1\}^2 + \sum_{i\in Q} \left(1 - \sum_{s_j \in S_i} p_{ij}\right)^2 \tag{3.3}$$

[1] Všeobecně převládá název *Lyapunov function* namísto *Laipunov function*, proto budeme i my používat výrazu Lyapunova funkce.

[2] V článku *A Laipunov Function for Nash Equilibria* [10] je první člen roven pouze $\sum_{i\in Q} \sum_{j\in m_i} \{\min[\sigma_{ij}, 0]\}^2$, není tedy udávána penalizace za číslo, které překročí $1,0$, já jsem tuto penalizaci přidal pro větší explicitnost.

První člen po vlastní Lyapunově funkci zajišťuje penalizaci bodů mimo interval $\langle 0, 1\rangle$. Mějme hráče $i \in Q$ a strategii $s_j \in S_i$. Pokud je p_{ij} menší než 0 nebo větší než 1, je k výsledku Lyapunovy funkce přidána tato odchylka umocněná na druhou. Druhý člen penalizuje takové body, které odpovídají celému pravděpodobnostnímu rozložení přes smíšenou strategii hráče i, které se nerovnají 1, tyto odchylky jsou potom sečteny a umocněny na druhou. Touto úpravou zajistíme, že lokální minimum bude vždycky ležet jen v prostoru Δ a minimalizační metoda nebude mít tendenci tento prostor opouštět.

Jako alternativní řešení je ve článku stavícím na Lyapunově funkci *Computing Nash equilibria through computational intelligence methods* [18] použita jen Lyapunova funkce funkce v a pro zajištění rovnice (2.2) je použita normalizace (rovnice 3.4) přes hodnoty zkoumaného bodu.

$$\forall i \in Q : p_{ij}^{p} = \frac{\|p_{ij}\|}{\sum_{j=1}^{m_i} \|p_{ij}\|} \tag{3.4}$$

Přičemž normalizace probíhá pouze pro výpočet hodnoty funkce, původní zkoumaný strategický profil je zachován a nenahrazen opraveným profilem, jinak by byla použitá metoda minimalizace výrazně degradována [18, str. 123].

V mých experimentech se normalizace bodu ukázala být jako rychlejším a zároveň robustnějším řešením než penalizace, proto bude v programu použita jako výchozí. Zároveň je však ve výsledném programu možnost použít i metodu penalizace.

3.4.2 Covariance Matrix Adaptation - Evolution Strategy

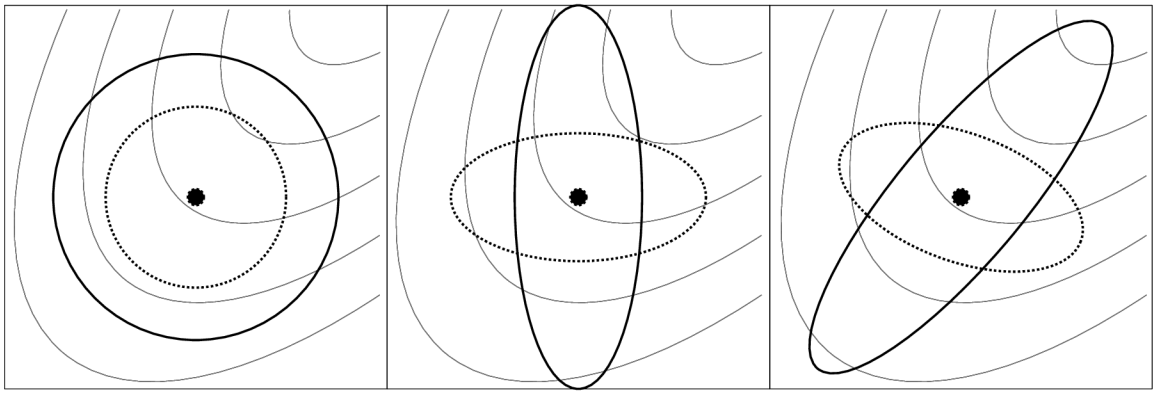
CMA-ES je stochastická metoda pro optimalizaci nelineárních, nekonvexních funkcí s reálnými parametry (se spojitou doménou). Pracuje na principu adaptace kovarianční matice pro použití s *vícerozměrným normálním rozdělením* (angl. mutlivariate normal distribution), kterým se vzorkují nové hledané body. Tato metoda zde bude použita pro *jednokriteteriální optimalizaci* (angl. single-objective optimization) Lyapunovy funkce 13. Pracuje pouze s objektivní funkcí, jedná se tedy o takzvanou *black-box* optimalizaci. Metoda CMA-ES zde bude představena jen v takové míře, jaká je důležitá pro tuto práci tzn. přehledově, pro hlubší pochopení souvislostí je třeba prostudovat citované materiály. V následujících sekcích budu vycházet z *CMA tutoriálu* [5].[3] Také bude použita notace z [5], vektory tedy budu značit malým tučným písmenem, matice velkým tučným písmenem.

Vícerozměrné normální rozložení

Vícerozměrné normální rozdělení je zobecněním normálního rozdělení pro vícerozměrnou náhodnou veličinu. Je značeno $\mathcal{N}(\mathbf{m}, \mathbf{C})$, kde $\mathbf{m} \in \mathbb{R}^n$ je střední hodnota rozložení, $\mathbf{C} \in \mathbb{R}^{n\times n}$ je symetrická pozitivně definitivní kovarianční matice, n je počet proměnných objektivní funkce, v případě minimalizace Lyapunovy funkce je $n = m$, kde m je suma strategií (2.1).

Kovarianční matice mají pro potřeby této metody příjemnou geometrickou interpretaci: mohou být považovány za hyperelipsoid $\left\{\mathbf{x} \in \mathbb{R}^n | \mathbf{x}^T\mathbf{C}^{-1}\mathbf{x} = 1\right\}$ (Obrázek 3.1). Hlavní osy tohoto elipsoidu odpovídají *vlastním vektorům* (angl. eigenvector) této matice. Odmocněné délky těchto os odpovídají *vlastním číslům* matice (angl. eigenvalue). *Rozklad na vlastní čísla a vektory* (angl. eigendecomposition) je možno zapsat jako $\mathbf{C} = \mathbf{B}(\mathbf{D})^2\mathbf{B}^T$.

[3] Některé vzorce odvozené v průběhu výkladu metody v CMA tutoriálu [5] nejsou shodné se vzorci uvedenými v závěrečném shrnutí, v této práci byly tyto vzorce opraveny podle závěrečného shrnutí, aby byla celá metoda konzistentní.



Obrázek 3.1: Přehled elipsoidů, kde tlustá čára značí stejnou hustotu dvourozměrného normálního rozložení. Vlevo je normální dvourozměrné rozložení $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, kde $\sigma \in \mathbb{R}_+$ je velikost kroku nebo také velikost prohledávané oblasti, $\mathbf{I}$ je jednotková matice. Uprostřed je dvourozměrné normální rozložení $\mathcal{N}(\mathbf{0}, \mathbf{D}^2)$, kde $\mathbf{D}$ je diagonální matice s odmocněnými vlastními čísly matice $\mathbf{C}$. Vpravo je dvourozměrné normální rozložení $\mathcal{N}(\mathbf{0}, \mathbf{C})$, kde $\mathbf{C}$ je symetrická pozitivně definitivní matice. Všechny pojmy budou vysvětleny dále v textu. Tenké linky značí vrstevnice objektivní funkce. [5, str. 6]

Kde $\mathbf{B} \in \mathbb{R}^{n\times n}$ je ortogonální matice se sloupci odpovídajícím vlastním vektorům matice $\mathbf{C}$. $\mathbf{D} \in \mathbb{R}^{n\times n}$ je diagonální matice, kde odmocněné diagonální prvky jsou vlastními čísly matice $\mathbf{C}$.

Vzorkování další generace

CMA-ES je metodou evoluční, budeme tedy pracovat s pojmy jako *jedinec*, *generace*, *populace*, *rodič* a *potomek*. Znalost těchto pojmů se předpokládá.

Nová generace je v CMA-ES vytvořena pomocí vzorkování vícerozměrného normálního rozložení.

$$\mathbf{x}_k^{(g+1)} \sim \mathcal{N}\left(\mathbf{m}^{(g)}, {\sigma^{(g)}}^2\mathbf{C}^{(g)}\right), \; k \; = 1, \ldots, \lambda$$

kde:

- $\sim$ je stejná distribuce
- g je číslo generace
- $\mathcal{N}\left(\mathbf{m}^{(g)}, {\sigma^{(g)}}^2\mathbf{C}^{(g)}\right)$ je vícerozměrné normální rozložení
- $\mathbf{x}_k^{(g+1)} \in \mathbb{R}^n$ je k-tý potomek nové generace
- $\mathbf{m}^{(g)} \in \mathbb{R}^n$ je střední hodnota normálního rozložení
- $\sigma^{(g)} \in \mathbb{R}_+$ je velikost kroku
- $\mathbf{C}^{(g)} \in \mathbb{R}^{n\times n}$ je kovarianční matice
- $\lambda > 2$ je velikost populace, počet potomků

V okamžiku, kdy máme populaci vzorků $\mathbf{x}_k, k = 1, \ldots, \lambda$, je každý tento jedinec použit jako argument objektivní funkce f. Výsledek této funkce bude ohodnocení každého jedince $\mathbf{x}_k$ a jejich seřazení do $\mathbf{x}_{i:\lambda}$, kde $i : \lambda$ značí i nejlepšího jedince. Pomocí této a dalších informací bude navzorkována nová generace. Jejich odvození si ukážeme v dalších kapitolách. Pokud některý jedinec splňuje ukončovací kritérium, tj. jedince lze označit za minimum funkce s určitou přesností, algoritmus je úspěšně dokončen.

Výpočet střední hodnoty nové generace

Každou generaci se přepočítává střední hodnota vícerozměrného rozložení. Tento bod by se měl každou generací blížit k hledanému minimu. Rovnice (3.5) vyjadřuje rekombinaci střední hodnoty a zároveň je v ní provedena selekce nejlepších bodů.

$$\mathbf{m}^{(g+1)} = \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}^{(g+1)} \tag{3.5}$$

Jednotlivé váhy w_i potom mají následující vlastnosti:

$$w_1 \geq w_2 \geq \cdots \geq w_\mu > 0, \quad \sum_{i=1}^{\mu} w_i = 1 \tag{3.6}$$

kde:

- $\mu \leq \lambda$ je velikost populace rodičů nové generace.
- $w_{i=1\ldots\mu} \in \mathbb{R}_+$ jsou kladné váhové koeficienty pro rekombinaci střední hodnoty normálního rozdělení
- $\mathbf{x}_{i:\lambda}^{(g+1)}$ je i nejlepší jedinec z $\left(\mathbf{x}_1^{g+1}, \ldots \mathbf{x}_\lambda^{(g+1)}\right)$. Pro index $i : \lambda$ platí $f(\mathbf{x}_{1:\lambda}^{(g+1)}) \geq f(\mathbf{x}_{2:\lambda}^{(g+1)}) \geq \cdots \geq f(\mathbf{x}_{\lambda:\lambda}^{(g+1)})$.
- f je funkce určená k optimalizaci, objektivní funkce.

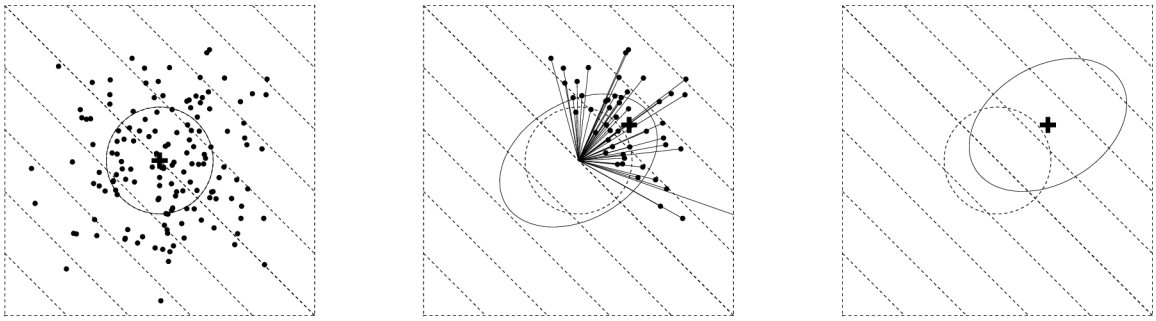
Funkce f je v algoritmu CMA-ES používána pouze ve výpočtu ohodnocení jedinců pro vektor vektorů $\mathbf{x}_{i:\lambda}$.

Koeficient kvality rozložení vah

Koeficient kvality rozložení vah $\mu_{eff} \in R_+$ (angl. variance effective selection mass). Tato míra bude později využívána pro výpočet kovarianční matice.

$$\mu_{eff} = \left(\sum_{i=1}^{\mu} w_i^2 \right)^{-1}$$

Ze znalosti rovnice 3.6 vyplývá, že $1 \leq \mu_{eff} \leq \mu$. Dále pak $\mu_{eff} = \mu$, pokud $w_i = \frac{1}{\mu}, i = 1, \ldots, \mu$. Podle *CMA tutoriálu* [5, str. 10] naznačuje $\mu_{eff} \approx \frac{\lambda}{4}$ dobré rozdělení vah.



Obrázek 3.2: Ukázka průběhu prvního kroku CMA-ES pro funkci s dvěma proměnnými. Minimum funkce je v pravém horním rohu. *Vzorkování*: na počátku jsou navzorkovány body vícerozměrným normálním rozložením s jednotkovou maticí. *Selekce*: Je vybráno μ rodičů, kteří budou vzorkovat další generaci. *Nové rozložení*: kovarianční matice se přizpůsobí výsledkům minulého vyhledávání (rovnice 3.8), střední hodnota se rekombinuje z nejlepších rodičů pomocí rovnice (3.5) [5, str. 12]
.

Evoluční cesta kovarianční matice

Evoluční cesta (angl. evolution path) je používána pro zachování úspěšně provedených kroků, a tím zajišťuje rychlejší učení, a tedy rychlejší konvergenci metody. Použitím kumulace úspěšných kroků nám dovoluje snížit populaci. Evoluční cesta slouží pro dále zmiňovaný *rank-one update*.

$$\mathbf{p}_c^{(g+1)} = (1 - c_c)\mathbf{p}_c^{(g)} + h_\sigma \sqrt{c_c(2 - c_c)\mu_{eff}}\langle \mathbf{y} \rangle_w \tag{3.7}$$

kde:

- $\mathbf{p}_c^{(g)} \in \mathbb{R}^n$ je evoluční cesta generace g
- $c_c \leq 1$ je míra učení (angl. learning rate) z předchozí evoluční cesty, pokud $c_c = 1$ vytváří se nová evoluční cesta pouze z rozdílu středních hodnot, pokud $c_c = 0$ je nová evoluční cesta rovna cestě z předchozí generace
- $\sqrt{c_c(2 - c_c)\mu_{eff}}$ je normalizační konstanta
- $h_\sigma = \begin{cases} 1 & pokud \frac{\|\mathbf{p}_\sigma\|}{\sqrt{1-(1-c_\sigma)^{2(g+1)}}} < (1.4 + \frac{2}{n+1})\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| \\ 0 & jinak \end{cases}$ jednotkový skok (angl. Heaviside step function) zpomalující růst $\mathbf{p}_c$ pokud je $\mathbf{p}_\sigma$ příliš velké.

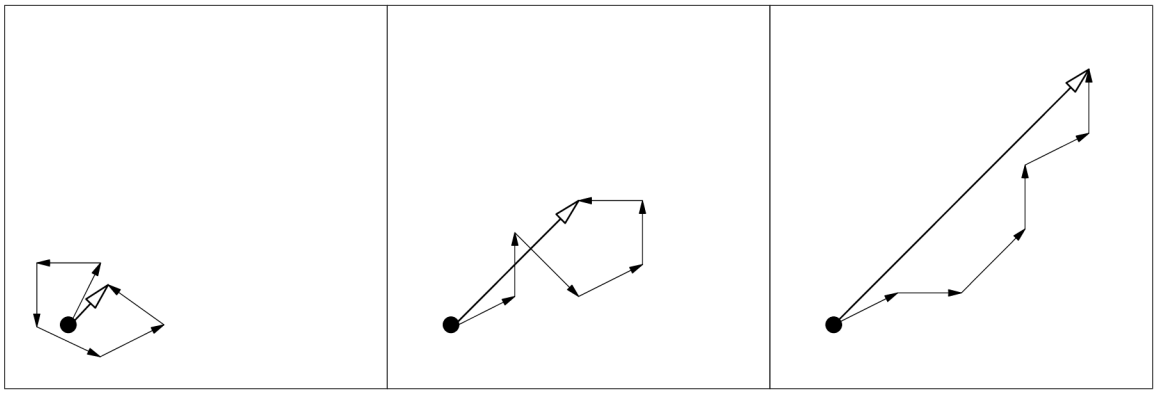
Adaptace kovarianční matice

Kovarianční matice určuje tvar vícerozměrného normálního rozložení. Jak můžeme vidět na Obrázku (3.2), matice mění tvar podle rodičů další populace. Dalo by se říct, že se "natahuje"po místu, kde očekává lepší výsledek, po místu s nejrychlejší změnou. Pro účely rychlejší konvergence k hledanému optimu jsou představeny dvě techniky: *rank-μ update* a *rank-one update*. Pomocí těchto technik možno snížit velikost populace, a tím zvětšit počet iterací, a tedy urychlit adaptaci kovarianční matice. Rank-μ update slouží pro zpřesnění výsledků malých populací. Rank-one update je použit pro limitní případ, kdy je vytvářen pouze jediný potomek a pomocí něho aktualizována kovarianční matice.

$$\begin{aligned}\mathbf{C}^{(g+1)} = &(1 - c_1 - c_\mu)\mathbf{C}^{(g)} \\ &+ c_1 \underbrace{\left(\mathbf{p}_c^{(g+1)}\mathbf{p}_c^{(g+1)^T} + \delta(h_\sigma)\mathbf{C}\right)}_{\text{rank-one update}} \\ &+ c_\mu \underbrace{\sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}^{(g+1)} \left(\mathbf{y}_{i:\lambda}^{(g+1)}\right)^T}_{\text{rank-}\mu\text{ update}}\end{aligned} \tag{3.8}$$

kde:

- $c_1 \approx 2/n^2$ je míra učení pro rank-one update
- $c_\mu \approx \min\left(\frac{\mu_{eff}}{n^2}, 1 - c_1\right)$ je míra učení pro rank-μ update
- $\mathbf{y}_{i:\lambda}^{(g+1)} = \frac{\mathbf{x}_{i:\lambda}^{(g+1)} - \mathbf{m}^{(g)}}{\sigma^{(g)}}$ je rozdíl nejlepších jedinců od střední hodnoty minulé generace
- $\delta(h_\sigma) = (1 - h_\sigma)c_c(2 - c_c)$, $\delta(h_\sigma) \leq 1$ nahrazuje druhý člen v rovnici (3.7) pokud $h_\sigma = 0$



Obrázek 3.3: Tři evoluční cesty pro různé problémy. Tyto cesty mají stejnou velikost kroku. Z obrázku je vidět, že délka evoluční cesty (suma kroků) je natolik odlišná, že potřeba kontroly velikosti kroku je nutná. [5, str. 18]

Evoluční cesta velikosti kroku

Tato evoluční cesta slouží pro dynamickou změnu délky kroku. Evoluční cesta je vlastně kumulací posledních úspěšných kroků. Jak si můžeme všimnout na obrázku (3.3), je možno rozdělit optimalizaci na 3 případy z hlediska velikosti kroku:

- Pokud je evoluční cesta krátká, kroky se vzájemně vyruší (Obrázek 3.3 vlevo). Délka kroku by tedy měla být snížena.
- Pokud jsou jednotlivé kroky na sebe přibližně kolmé, délka kroku by se měnit neměla (Obrázek 3.3 uprostřed).
- Pokud je evoluční cesta dlouhá, leží kroky stejným směrem.
- n je počet proměnných objektivní funkceDélka kroku by měla být prodloužena (Obrázek 3.3 vpravo).

Správná reakce na tyto situace je úkolem právě kontroly délky kroku. Evoluční cesta pro délku kroku je definována takto:

$$\mathbf{p}_\sigma^{(g+1)} = (1 - c_\sigma)\mathbf{p}_\sigma^{(g)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\mathbf{C}^{(g)^{-\frac{1}{2}}}\langle \mathbf{y} \rangle_w$$

kde:

- $\mathbf{p}_\sigma^{(g)} \in \mathbb{R}^n$ je evoluční cesta velikosti kroku generace g
- $c_\sigma < 1$ je míra učení z předchozí evoluční cesty
- $\sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}$ je normalizační konstanta
- $\mathbf{C}^{(g)^{-\frac{1}{2}}} = \mathbf{B}^{(g)}\mathbf{D}^{(g)^{-1}}\mathbf{B}^{(g)^T}$ zajišťuje, že očekávaná délka $p_\sigma^{(g+1)}$ je nezávislá na směru kroku
- $\langle \mathbf{y} \rangle_w = \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}$ je krok střední hodnoty rozložení, nezávislý na délce kroku σ

Kontrola délky kroku

Při výpočtu nového kroku je porovnána délka evoluční cesty $\|\mathbf{p}_\sigma^{(g+1)}\|$ s její očekávanou délkou $\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|$

$$\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\| = \frac{\sqrt{2}\Gamma\left(\frac{n+1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)} \approx \sqrt{n} + (1 - \frac{1}{4n} + \frac{1}{21n^2})$$

Γ je rozšíření funkce faktoriálu s argumentem sníženým o 1. Matematickým přepisem $\Gamma(n) = (n-1)!$.

Pokud je délka evoluční cesty delší než očekávaná délka, je krok prodloužen. Pokud byl odhad správný, krok se nemění. Pokud je délka evoluční cesty kratší než očekávaná délka, je krok zkrácen.

$$\sigma^{(g+1)} = \sigma^{(g)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma^{(g+1)}\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)$$

kde:

- $d_\sigma \approx 1$ damping parametr, určuje změnu $\sigma^{(g)}$

Kritéria ukončení

Algoritmus má několik kritérií, při kterých se ukončí. Pokud nebylo dosaženo hledaného minima, je možnost algoritmus restartovat, jak je popsáno v následující kapitole. Všechny podmínky jsou popsány v CMA tutoriálu [5, str. 28]. Kromě první značí všechny ukončovací podmínky neúspěch:

- Nalezení minima. Funkční hodnota nejlepšího bodu je menší než 10^{-10}. Algoritmus našel minimum s požadovanou přesností.
- Neefektivní osy. Přidání 0.1 směrodatné odchylky k jakékoliv hlavní ose $\mathbf{C}$ nezmění $\mathbf{m}$. Formálně $\mathbf{m} = \mathbf{m} + 0.1\sigma d_{ii}\mathbf{b}_i, i = (g \mod n) + 1$.
- Neefektivní souřadnice. Přidání 0.2 směrodatné odchylky k jakékoli souřadnici nezmění $\mathbf{m}$. $\forall i \in Q : m_i = m_i + 0.2\sigma c_{ii}$.
- Shodné funkční hodnoty. Pokud je rozsah funkčních hodnot nejlepších jedinců posledních $10 + \lceil \frac{30N}{\lambda} \rceil$ generací roven 0.
- Stagnace. Zkoumáme medián funkční hodnoty 20% posledních g generací $120+30\frac{n}{\lambda} \leq g \leq 20000$. Skončíme pokud není medián posledních 30% záznamů lepší než medián prvních 30% záznamů.[4]
- Podmíněnost matice. Pokud *podmíněnost matice* (angl. condition number) přesáhne 10^{14}.
- Horní tolerance (v originále *TolXUp*). Pokud $\sigma \times \max(diag(\mathbf{D}))$ přesáhne 10^4. Tato podmínka naznačuje příliš malé počáteční σ nebo divergentní chování metody.
- Tolerance funkčích hodnot (v originále *TolFun*). Pokud je rozsah funkčních hodnot nejlepších jedinců posledních $10+\lceil \frac{30N}{\lambda} \rceil$ menší než 10^{-12} a zároveň pokud jsou všechny funkční hodnoty poslední generace pod touto hodnotou.
- Tolerance kovarianční matice (v originále *TolX*). Pokud je $\sigma\mathbf{p}_c < TolX$ a zároveň $\sigma\sqrt{\mathbf{C}} < TolX$. $TolX = \sigma^{(0)}10^{-12}$
- Stejné funkční hodnoty (v originále *flat fitness*). Pokud je funkční hodnota několika jedinců poslední generace shodná.

Restart algoritmu se zvětšenou populací

Během testování se ukázalo, že algoritmus někdy uvázne v lokálním minimu. Problém lokálních minim Lyapunovy funkce je ostatně popsán i v McKelveyho článku Laipunov function for Nash Equilibria [10, str. 6]. Metoda samotná by měla do určité míry tyto lokální minima překonávat, avšak ve větších hrách se to nemusí podařit. Tento problém byl řešen na základě článku *A restart CMA evolution strategy with increasing population size* [1]. Vychází se z předpokladu, že větší populace povede ke globálnímu výsledku s větší pravděpodobností. Pokud se stane, že algoritmus uvázne v lokálním minimu, je zastaven ukončovacími kritérii z předchozí kapitoly a je proveden restart celé metody, tentokrát však s populací navýšenou dvojnásobně oproti minulému pokusu.

[4] Toto ukončovací kritérium jsem v programu nepoužil, často ukončovalo algoritmus v momentě, kdy už chybělo jen pár iterací k dosažení požadované přesnosti.

Přehled použitých parametrů

Metoda CMA-ES používá celou řadu parametrů, jejichž hodnot bylo experimentálně docíleno jejím autorem, Nikolausem Hansenem. Tyto hodnoty byly převzaty z CMA tutoriálu [5, str. 27].

Selekce a rekombinace:

$$\lambda = 4 + \lfloor 3 + \ln n \rfloor, \quad \mu = \left\lfloor \frac{\lambda}{2} \right\rfloor$$

$$w_i = \frac{w_i'}{\sum_{j=1}^{\mu} w_j'}, \quad w_i' = \ln(\mu' + 0.5) - \ln i \quad \text{for } i = 1, \ldots, \mu$$

Kontrola délky kroku:

$$c_\sigma = \frac{\mu_{eff} + 2}{n + \mu_{eff} + 5}, \quad d_\sigma = 1 + 2max\left(0, \sqrt{\frac{\mu_{eff} - 1}{n + 1}} - 1\right) + c_\sigma$$

Adaptace kovarianční matice:

$$c_c = \frac{4 + \frac{\mu_{eff}}{n}}{n + 4 + \frac{2\mu_{eff}}{n}}$$

$$c_1 = \frac{2}{(n + 1.3)^2 + \mu_{eff}}$$

$$c_\mu = \min\left(1 - c_1, 2\frac{\mu_{eff} - 2 + \frac{1}{\mu_{eff}}}{(n + 2)^2 + \mu_{eff}}\right)$$

Celkový algoritmus

Zde je představen celý algoritmus metody CMA-ES. Je v něm zahrnuto i restartování se zvětšenou populací. Význam proměnných a hodnoty parametrů byly objasněny v předcházejících kapitolách.

Algoritmus 4 Covariance Matrix Adaptation - Evolution Strategies

Input: Objektivní funkce f

while nenalezeno minimum **do**

Nastav parametry $\lambda, \mu, w_{i=1,\dots,\mu}, c_\sigma, d_\sigma, c_c, c_1, c_\mu$

Inicializuj $\mathbf{p}_\sigma = \mathbf{p}_c = \mathbf{0}, \mathbf{C} = \mathbf{I}, g = 0$, vyber $\mathbf{m} \in \mathbb{R}^n$ a $\sigma \in \mathbb{R}_+$ podle řešené úlohy

while není splněna ukončovací podmínka **do**

Vzorkování potomků

$$\begin{aligned}\mathbf{z}_k &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{y}_k &= \mathbf{BDz}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{C}) \\ \mathbf{x}_k &= \mathbf{m} + \sigma \mathbf{y}_k \sim \mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{C})\end{aligned}$$

Selekce a rekombinace

$$\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \mathbf{x}_{i:\lambda}$$

Kontrola délky kroku

$$\begin{aligned}\mathbf{p}_\sigma &\leftarrow (1 - c_\sigma)\mathbf{p}_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_{eff}}\mathbf{C}^{-\frac{1}{2}}\langle \mathbf{y} \rangle_w \\ \sigma &\leftarrow \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\mathbf{p}_\sigma\|}{\mathsf{E}\|\mathcal{N}(\mathbf{0}, \mathbf{I})\|} - 1\right)\right)\end{aligned}$$

Adaptace kovarianční matice

$$\begin{aligned}\mathbf{p}_c &\leftarrow (1 - c_c)\mathbf{p}_c + h_\sigma \sqrt{c_c(2 - c_c)\mu_{eff}}\langle \mathbf{y} \rangle_w \\ \mathbf{C} &\leftarrow (1 - c_1 - c_\mu)\mathbf{C} + c_1\left(\mathbf{p}_c\mathbf{p}_c^T + \delta(h_\sigma)\mathbf{C}\right) + c_\mu \sum_{i=1}^{\mu} w_i \mathbf{y}_{i:\lambda}\mathbf{y}_{i:\lambda}^T\end{aligned}$$

end while

if nenalezeno minimum **then**

$\lambda \leftarrow 2\lambda$

end if

end while

Output: $\mathbf{x}_{1:\lambda}, f(\mathbf{x}_{1:\lambda}) < 10^{-10}$

Kapitola 4

Implementace

V této kapitole budou popsány implementační detaily nástroje pro výpočet Nashova ekvilibria, který byl pracovně pojmenován `NenG` (Nash Equilibria Non-cooperative Games). Výstupem této práce je aplikace pracující v příkazovém řádku, která na vstupu očekává hru ve formátu `.nfg`, který je standardem pro program Gambit [12], který řeší stejnou problematiku jako tato práce. Podle zvolených dalších přepínačů popsaných v uživatelském manuálu podá na výstup zprávu o výsledku výpočtu.

Jako implementační jazyk byl zvolen Python ve verzi 2.7. Pro volbu tohoto jazyka byly následující důvody:

- Díky kvalitním knihovnám `NumPy` [16] a `SciPy` [8] jsou v Pythonu všechny potřebné matematické operace snadno dostupné a efektivní. V Pythonu se také nabízí také rozsáhlou knihovnu pro tvorbu grafů `Matplotlib` [7].
- Jazyk Python je přehledný, kód tak může lépe sloužit jako učební pomůcka.
- Nabízí nástroje pro snadnou správu projektu jako je debugger pdb, interaktivní odchytávání výjimek pomocí programu IPython, profiler `profile`, `line-profiler`.

4.1 Implementační detaily

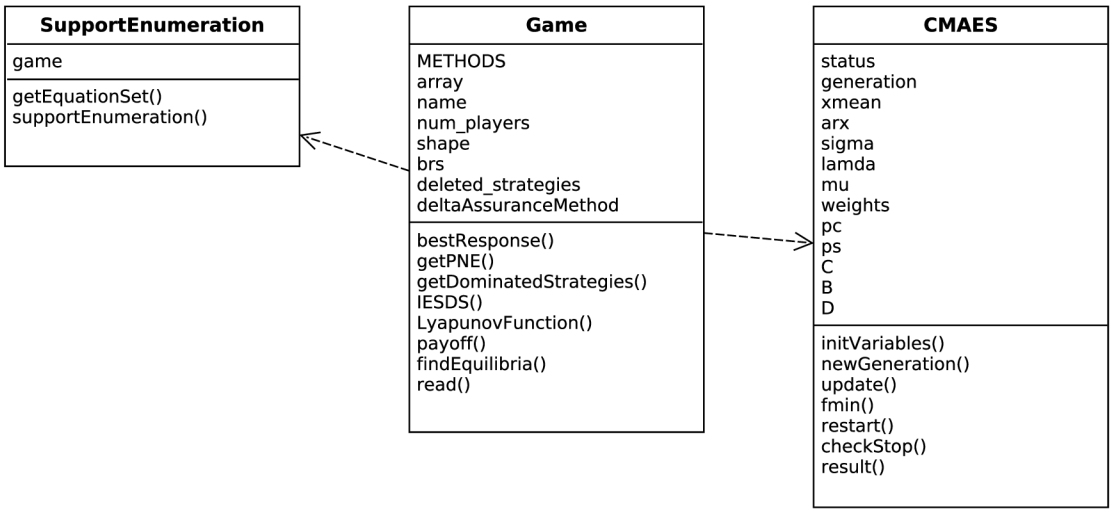
Program se skládá ze tří tříd, které zajišťují veškeré funkce potřebné pro splnění zadání. Implementace metod je programovým přepisem matematických postupů a definic z kapitoly (3).

Třída `Game` obaluje celou hru, zpracovává vstup, vypracovává výstup, provádí základní analýzu hry (např. zda je hra degenerovaná), provádí iterativní eliminaci striktně dominovaných strategií `Game.IESDS()` (algoritmus 2), počítá ryzí Nashovo ekvilibrium hrubou silou `Game.getPNE()` (algoritmus 1), provádí test, zda nalezený strategický profil je opravdu Nashovo ekvilibrium `Game.checkNE()` (kapitola 4.1.1), nabízí také rozcestník pro všechny metody nalezení Nashova ekvilibria `Game.findEquilibria()`.

Ve třídě `SupportEnumeration` je implementován algoritmus Vyčíslení domén (3). Obsahuje funkci `SupportEnumeration.getEquationSet()` pro vygenerování soustavy rovnic, která se používá v hlavní funkci `SupportEnumeration.supportEnumeration()`, která provede výpočet všech smíšených Nashových ekvilibrií ve dvouhráčové hře.

Třída `CMAES` implementuje celý algoritmus 4. Implementace tohoto algoritmu vychází z *CMA tutoriálu* [5], kde je základní algoritmus představen v jazyce Matlab. Třída `Game`

pracuje po jednotlivých iteracích. Na počátku jsou inicializovány proměnné třídy v konstruktoru, pomocí funkce `CMAES.init_variables()` jsou inicializovány proměnné závislé na parametru λ (velikost populace), to nám dává možnost provádět restart celé metody. Dále už algoritmus probíhá v iteracích, kdy je nejdříve volána metoda `CMAES.new_generation()` pro navzorkování nových jedinců. Ti jsou posléze ohodnoceni objektivní funkcí a je volána metoda `CMAES.update()` pro aktualizaci všech proměnných na základě nové generace. V každé iteraci se pomocí `CMAES.check_stop()` kontroluje, zda byla splněna některá z ukončovacích podmínek. Pokud byl algoritmus ukončen úspěšně, je vrácen výsledek a algoritmus končí. Pokud však skončil chybou, je metoda restartována se zdvojnásobenou populací (kapitola 3.4.2) a celý algoritmus začíná znovu.



Obrázek 4.1: UML diagram tříd programu `NenG`

4.1.1 Test Nashova ekvilibria

Pro provedení kontroly, zda nalezený strategický profil je opravdu Nashovo ekvilibrium, byl implementován test zkoumající, jestli jeden z hráčů bude mít tendence změnit svou strategii, a tak si zvýšit užitek ze hry. Tento test probíhá následujícím způsobem: nejdříve je prozkoumáno, zda má hráč lepší užitek z jakékoliv ryzí strategie, kterou může hrát. Pokud žádná taková strategie není, začne hráč i náhodně generovat strategie z množiny Δ_i. Tuto strategii si pak zvolí za vlastní a zkoumá, zda má s náhodnou strategií užitek větší než se zkoumaným potenciálním Nashovým ekvilibriem. Jako maximální hodnota, o kterou se může užitek zvýšit, byla experimentálně zvolena 10^{-4}. Testů s náhodným generováním je provedeno 1000 pro každého hráče. Pokud zkoumaný strategický profil obstojí v každém testu, je test Nashova ekvilibria úspěšný úspěšný a strategický profil prohlášen za Nashovo ekvilibrium.

Algoritmus 5 Test Nashova ekvilibria

Input: $\sigma^* \subset \Delta$ ▷ zkoumaný strategický profil
  for all $i \in Q$ **do**
    for all $s_j \in S_i$ **do**
      if $U_i(s_j, \sigma^*_{-i}) - U_i(\sigma^*) > 0,0001$ **then**
        return False ▷ strategický profil neobstál test
      end if
    end for
    for $i = 1 \rightarrow 1000$ **do**
      $randomStrategy \leftarrow random(\Delta_i)$
      if $U_i(randomStrategy) - U_i(\sigma^*) > 0,0001$ **then**
        return False ▷ strategický profil neobstál test
      end if
    end for
  end for
  return True ▷ strategický profil je Nashovo ekvilibrium
Output: True, pokud σ^* je smíšené Nashovo ekvilibrium, False jinak

Funkce $random(\Delta_i)$ generuje náhodné vektory z prostoru Δ_i.

4.2 Optimalizace

Profilace a optimalizace proběhla v závěrečné fázi vývoje `NenG`. K funkční profilaci byl použit profiler v programu `IPython`. Profilováním standardního průběhu při výpočtu smíšeného Nashova ekvilibria metodou CMA-ES se ukázala úzká hrdla aplikace. Jak lze předpokládat je to Lyapunova funkce `Game.LyapunovFunction()` a z ní volaná funkce pro zjištění užitku daného strategické profilu `Game.payoff()`. Na tyto funkce byl použit `line_profiler`, díky kterému byly nejnáročnější operace odhaleny a přepsány na méně náročnější variantu. Díky tomu byl například čas výpočtu smíšeného Nashova ekvilibria metodou CMA-ES ve hře `2x2x2x2x2.nfg` snížen na polovinu.

Kapitola 5

Experimenty

V této části se budeme zabývat experimenty nad programem `NenG`. Bude prozkoumána hlavně doba běhu programu, a správnost výpočtu. Výsledky budou porovnány s vhodnými konkurenty. Nejdříve bude prozkoumána náročnost implementace algoritmu pro výpočet ryzího Nashova ekvilibria hrubou silou (1) a bude porovnána s programem `gambit-enumpure`. Dále bude prozkoumán algoritmus Vyčíslení domén (3), za konkurenta jsem v této úloze zvolil program `gambit-enummixed`. Poslední experiment se zabývá minimalizací Lyapunovy funkce (definice 13) pomocí metody CMA-ES (kapitola 3.4.2), která je implementována v programu `NenG`, a v porovnání s dalšími metodami L-BFGS-B a SLSQP, které pochází z balíku `scipy.optimize`.

Experimenty jsou k nalezení na přiloženém CD v souboru `performance_tests.py`[1]. Pro každý experiment je připravena sada her s různou obtížností. Nashovo ekvilibrium v každé hře je vypočítáno daným programem a metodou, je zapsáno trvání výpočtu. Pokud nebylo Nashovo ekvilibrium nalezeno, není tento čas použit (výjimka je u výpočtu ryzího Nashova ekvilibria, zde nenalezení ryzího ekvilibria neznamená chybu programu, ale vlastnost hry). Dále je použit test Nashova ekvilibria, který je předveden v kapitole (4.1.1). Pokud ekvilibrium tímto testem neprojde, je trvání tohoto výpočtu také zahozeno. Po dokončení zvolených experimentů jsou soubory s časy zálohovány a vytvořeny grafy pomocí knihovny `matplotlib`.

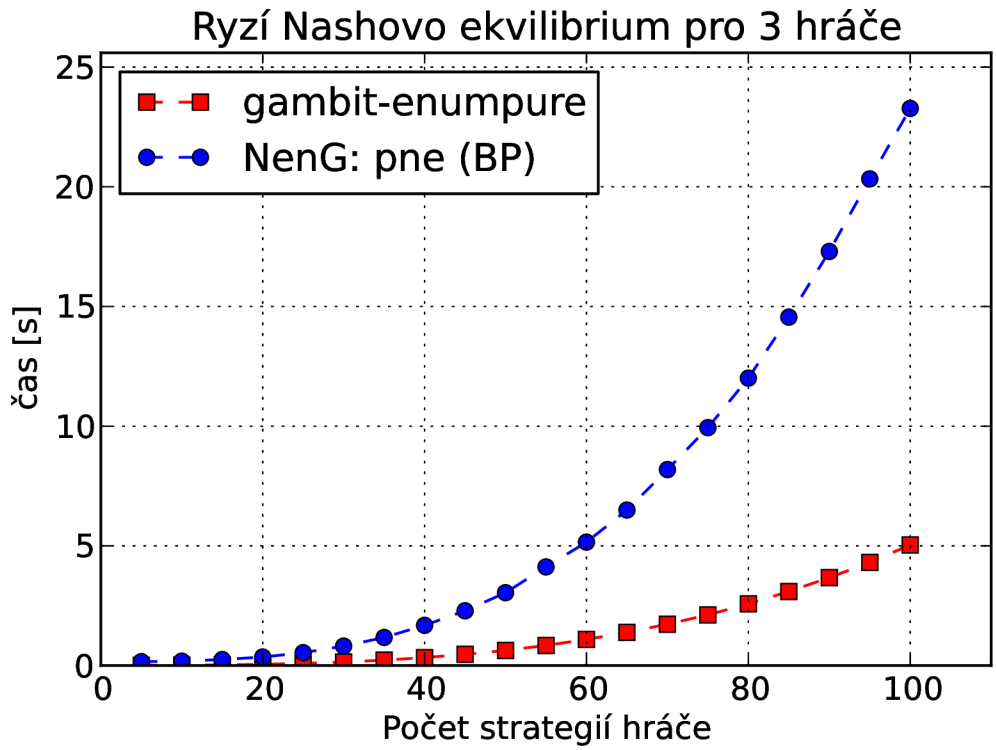
5.1 Ryzí Nashovo ekvilibrium

Pro účely testování funkčnosti výpočtu všech ryzích Nashových ekvilibrií hrubou silou (algoritmus 1) byly vygenerovány hry pomocí programu `Gamut` [15], který generuje různé třídy nekooperativních her. Pro generování byl použit náhodný výběr, do které třídy má hra spadat, pro účely experimentování byly vygenerovány o 2 až 7 hráčích a různých počtech strategií.

Jako protivníka mého algoritmu jsem vybral program `gambit-enumpure` z balíku `gambit`. Tento program provádí výpočet všech ryzích Nashových ekvilibrií ve vícehráčových hrách, tedy provádí přesně to, co metoda `Game.getPNE()` z mého programu.

V těchto experimentech byla zjištěna 100 % správnost nalezených výsledků. V každé zkoumané hře, v každém opakování, byly nalezeny všechna ryzí Nashova ekvilibria dané hry, ať už mým programem `NenG` nebo programem `gambit-enumpure`.

[1] Tento skript slouží čistě jen pro účely provedení experimentů. Nebyl nijak odladěn ani při jeho psaní nebyl brán zřetel na výkon. Je přiložen pouze k předvedení metodiky experimentování.



Obrázek 5.1: Ukázka jednoho experimentu. Závislost doby běhu programu na počtu strategií pro hry o třech hráčích.

Z experimentu 5.1 můžeme pozorovat, že časový vývoj výpočtu je exponenciální, což splnilo očekávání. Dále si lze všimnout, že program `gambit-enumpure` je rychlejší než můj program `NenG` s metodou výpočtu `pne`. To by se dalo vysvětlit tím, že program `gambit-enumpure` je napsán v jazyce C++, a tudíž tedy kompilovaný, naproti tomu `NenG` je napsán ve skriptovací jazyce Python, který je z podstaty pomalejší. Také byl balík `Gambit` vyvíjen mnohem delší dobu, jeho počátky se datují do 80. let 20. století.

Dalším důležitým činitelem v době trvání celého skriptu je parsování vstupního souboru do vnitřní struktury `Game`. V následující tabulce máme vypsánu časovou náročnost čtení vstupní hry vzhledem k výpočtu celého algoritmu:

Hra	Velikost souboru [MB]	Čas čtení [s]	Čas výpočtu [s]	Celkový čas [s]	Čas čtení / celkový čas
p4a5.nfg	0,012	0,024	0,845	0,869	0,027
p4a10.nfg	0,172	0,293	1,019	1,312	0,223
p4a15.nfg	0,872	1,494	1,152	2,646	0.565
p4a20.nfg	2,7	4,774	1,573	6,347	0,752
p4a25.nfg	6,6	11,709	2,229	13,938	0,840
p4a30.nfg	14	24,409	3,270	27,679	0,881
p4a35.nfg	26	45,502	5,000	50,502	0,901

Tabulka 5.1: Tabulka ukazuje vzrůstající poměr mezi časem pro čtení hry a celkovým časem výpočtu se vzrůstající velikostí hry. Pro velké hry dosahuje parsování vstupního souboru 90% celkového času.

Z tabulky (5.1) vyplývá, že u velkých her se dostává doba výpočtu ryzího ekvilibria do pozadí a velkou roli na celkovém čase hraje čtení vstupního souboru.

Pro zmírnění tohoto problému byl vytvořen parser pomocí knihovny `pyparsing`, ten lze nalézt na přiloženém CD v souboru `nfgparser.py`. Tento postup jsem prozkoumal a ukázal se být ještě pomalejší než původní řešení, proto jsem zůstal u ručního parsování vstupního souboru.

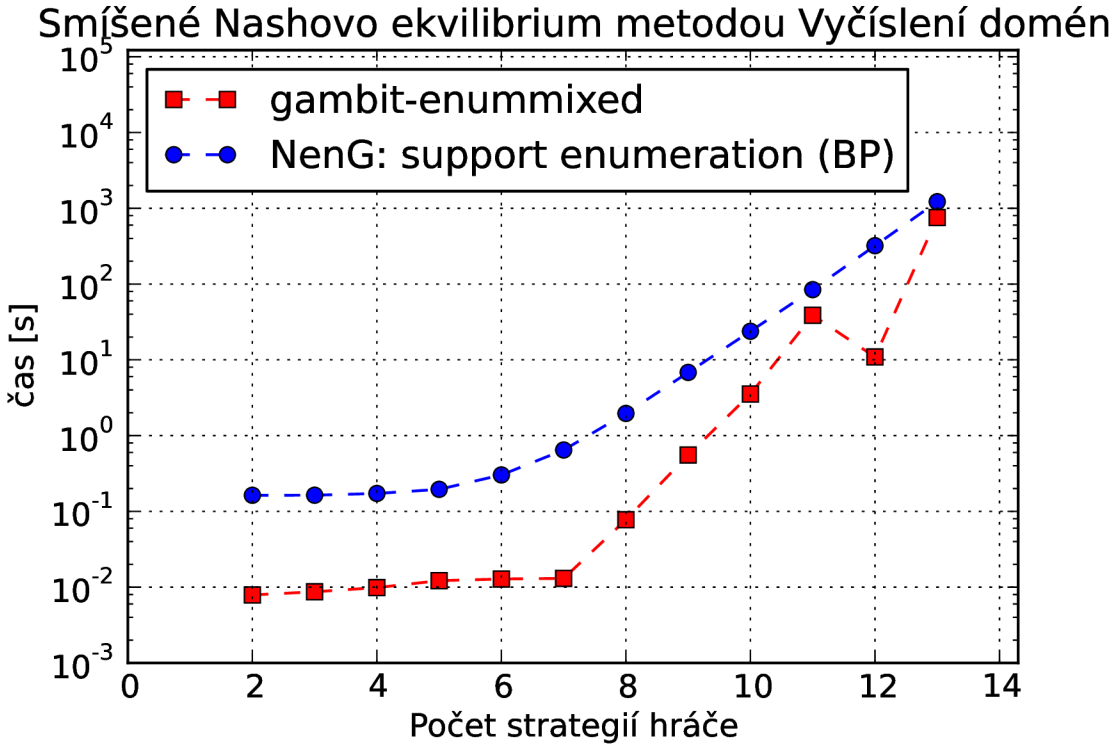
5.2 Smíšené Nashovo ekvilibrium ve dvouhráčových hrách

V tomto experimentu budeme zkoumat dobu běhu programu a správnost výpočtu všech smíšených Nashových ekvilibrií ve dvouhráčových hrách pomocí algoritmu Vyčíslení domén (3).

Pro tento experiment byl použit také program `Gamut` [15] generující nekooperativní hry. Stejně jako v minulém experimentu byly generovány hry z náhodné třídy s náhodnými užitky. Byly vygenerovány hry v rozsahu strategií 2 až 13 pro každého hráče.

Jako protivník mé aplikaci byl zvolen program `gambit-enummixed`. Ten hledá všechny smíšené Nashovy ekvilibria ve dvouhráčových hrách metodou Vyčíslení extrémního bodu. To je stejná úloha jako řeší můj program pomocí metody Vyčíslení domén. Algoritmus Vyčíslení domén je implementován v programu `gambit-enumpoly` ten však nemohl být použít, protože na některých hrách nebyl schopen dokončit výpočet.

Následující graf je graf závislosti době trvání výpočtu na počtu strategií pro oba hráče.



Obrázek 5.2: Graf časové náročnosti (v logaritmickém měřítku) výpočtu smíšeného Nashova ekvilibria pomocí metody vyčíslení domén v porovnání s programem `gambit-enummixed`

V tomto experimentu byly v každé hře nalezena všechna smíšená Nashova ekvilibria.

Bylo toho dosaženo i mou aplikací `NenG` i programem `gambit-enummixed`.

Z grafu lze vyčíst, že je program `gambit-enummixed` rychlejší pro zobrazený počet strategií. Za předpokladu, že by byla zachována stejná tendence růstu obou funkcí, je možno předpovídat, že by můj program `NenG` program `gambit-enummixed` předčil v ještě náročnějších hrách. Zůstává ovšem otázkou, zda je tak dlouho trvající výpočet ještě rentabilní.

5.3 Smíšené Nashovo ekvilibrium ve vícehráčových hrách

Experiment zkoumající nejdůležitější část mého programu, tedy minimalizaci Lyapunovy funkce (13), byl prováděn hned se třemi metodami pro minimalizaci funkce: CMA-ES (4), která je implementována v `NenG`, L-BFGS-B a SLSQP, které byly přejaty z balíku `scipy.optimize`. Původním záměrem bylo použít více metod z tohoto balíku [2] , žádná další z nich však nezvládla alespoň částečně spolehlivě dokonvergovat k výsledku s dostatečnou přesností. Podobný problém nastal i při testování programu `gambit-liap`, který také implementuje hledání Nashova algoritmu pomocí minimalizaci Lyapunovy funkce. Tento program skončí bez výsledku s jakýmkoli nastavením, proto nebyl do experimentu zahrnut.

Testovací skupina her byla převzata z nástroje `Gambit`, jedná se o stejné hry, které byly použity v článku [18] a byly také vygenerovány dvě hry větší. Všechny tyto hry jsou k nalezení na přiloženém CD:

coord333 tříhráčová hra s třemi strategiemi pro každého hráče. Tato hra má celkem 13 Nashových ekvilibrií.

coord4 dvouhráčová hra se čtyřmi strategiemi pro oba hráče. Tato hra má celkem 15 Nashových ekvilibrií.

2x2x2 tříhráčová hra se dvěma strategiemi pro každého hráče. Má celkem 9 Nashových ekvilibrií.

2x2x2x2 čtyřhráčová hra se dvěma strategiemi pro každého hráče. Tato hra je řešitelná 3 Nashovými ekvilibrii.

2x2x2x2x2 pětihráčová hra se dvěma strategiemi pro každého hráče. Tato hra je charakterizována 5 Nashovými ekvilibrii.

5x5x5 tříhráčová hra s pěti strategiemi pro každého hráče. Tuto hru lze vyřešit 5 Nashovými ekvilibrii.

7x7x7 tříhráčová hra se sedmi strategiemi pro každého hráče. Tato hra má 3 řešení v Nashových ekvilibriích.

Tabulka (5.2) nám ukazuje, že hledání smíšených Nashových ekvilibrií funguje spolehlivě především u malých her. U her, převzatých z nástroje `Gambit`, bylo metodou CMA-ES nalezeno smíšené Nashovo ekvilibrium pokaždé. Pro mnou generované větší hry už úspěšnost klesá.

Dalším zajímavým výsledkem tohoto experimentu je charakter Nashových ekvilibrií. Metoda převážně konverguje k malé podmnožině ekvilibrií, u některých her dokonce opakovaně vrací jediné ekvilibrium, i když jich hra má několik. Tato vlastnost se projevuje

[2] Pro přehled těchto metod navštivte http://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

Metoda	L-BFGS-B		SLSQP		CMA-ES	
Hra	úspěšnost [%]	čas [s]	úspěšnost [%]	čas [s]	úspěšnost [%]	čas [s]
coord333	100	0.289	100	0.417	100	1.168
coord4	100	0.357	80	0.353	100	0.880
2x2x2	100	0.258	100	0.256	100	0.859
2x2x2x2	100	0.599	100	0.457	100	1.915
2x2x2x2x2	60	1.403	80	0.908	100	4.227
5x5x5	20	5.135	80	3.046	40	27.162
7x7x7	20	39.353	20	11.441	20	154.264

Tabulka 5.2: Tabulka úspěšnosti a průměrné doby nalezení smíšeného Nashova ekvilibria použitých metod u každé z testovaných her.

v různé míře v závislosti na počítané hře. Lze si to vysvětlit tím, že tvar Lyapunovy funkce určitým způsobem zvýhodňuje některé body při minimalizaci.

V tabulce 5.2 můžeme vidět časovou náročnost jednotlivých metod, které řeší popsané hry. Metoda CMA-ES je pomalejší než metody SLSQP a L-BFGS-B, ale zvlášť u menších her není rozdíl nijak markantní a můžeme tak o ní prohlásit, že je srovnatelně náročná. Její horší výsledky si můžeme vysvětlit různou implementací těchto metod. Metody SLSQP a L-BFGS-B jsou součástí knihovny `SciPy`, ta je celá napsána v jazyce C a Fortran pro větší efektivitu a v jazyce Python je jen abstraktní vrstva pro volání těchto funkcí.

Také je třeba podotknout, že metoda CMA-ES je 100 % úspěšná ve více hrách než metody L-BFGS-B a SLSQP, jak můžeme vidět v tabulce (5.2).

5.3.1 Ukázka výpočtu smíšeného Nashova ekvilibria metodou CMA-ES

V této kapitole si představíme průběh výpočtu smíšeného Nashova ekvilibria pomocí minimalizace Lyapunovy funkce metodou CMA-ES. Za příklad nám poslouží hra `2x2x2` představená v minulé kapitole. Pro zajištění charakteru rozložení pravděpodobnosti přes jednotlivé body, použijeme metodu penalizace, díky které uvidíme skutečný bod, ke kterému algoritmus konverguje, lépe než při použití metody normalizace.

Generace	σ	Ohodnocení nejlepšího	Nejlepší jedinec
1	2.70e-01	3.08e+00	[-0.0866531, 0.22215836, 1.04637156, 0.05893053, -0.14015331, 0.63243666]
5	2.28e-01	1.56e+00	[0.06754021, 0.20152466, 0.29852439, 0.39390428, -0.17319681, 0.38299942]
10	1.42e-01	9.95e-01	[0.26475969, 0.27146635, 0.24076801, 0.3769357, -0.06037462, 0.71678507]
20	1.39e-01	6.96e-02	[0.37722237, 0.58319589, 0.58449225, 0.62847485, 0.03075299, 0.97571681]
60	2.24e-02	3.10e-03	[0.43387958, 0.56860606, 0.50901505, 0.52251524, 0.21664792, 0.81257705]
100	6.09e-03	2.79e-05	[0.40503743, 0.59800664, 0.49408558, 0.50928612, 0.33934125, 0.66004809]
140	1.18e-03	8.54e-08	[0.39959055, 0.60046442, 0.49993324, 0.50018775, 0.33363872, 0.66660178]
180	2.29e-04	1.38e-09	[0.39997612, 0.60003517, 0.50004305, 0.49998747, 0.33332453, 0.66669253]
196	8.67e-05	5.35e-11	[0.39999035, 0.60001066, 0.50000507, 0.49999827, 0.33333369, 0.66667255]

Tabulka 5.3: Průběžné výsledky metody CMA-ES při řešení hry `2x2x2.nfg` s metodou penalizace.

V tabulce (5.3) můžeme vidět že po úvodním bouřlivém začátku, kdy některé hodnoty jsou ještě dokonce záporné, nastává zpřesňování nalezeného bodu až bod v generaci 196 překročí hranici pro úspěšné zakončení algoritmu (10^{-10}). V tabulce si taky můžeme všimnout jak se zmenšuje délka kroku σ a tím se zmenšuje možný prostor, ze kterého se mohou vzorkovat další jedinci.

Kapitola 6

Závěr

V této práci jsem popsal teorii her pro nekooperativní hry. Byly vymezeny základní definice jednotlivých entit a definovány vztahy mezi nimi. Provedl jsem průzkum možných řešení výpočtu Nashova ekvilibria a pro každý jednotlivý případ jsem vybral vhodný algoritmus. Prostředkem pro vyřešení nejtěžšího problému, tedy nalezení smíšeného Nashova ekvilibria ve vícehráčových hrách, je relativně nová metoda pro minimalizaci funkce Covariance Matrix Adaptation - Evolution Strategy (CMA-ES). Představil jsem svou implementaci vytvořeného nástroje pro výpočet Nashova ekvilibria - `NenG` (Nash Equilibria Noncooperative Games). Sesbíral a vygeneroval jsem množství her, které jsem použil pro otestování mého programu. `NenG` prošel testy úspěšně a nalezl Nashova ekvilibria v rozsahu stanoveném v zadání. Ukázal se být konkurenceschopný mnohem staršímu programu `Gambit`. Během psaní programu jsem také bral ohledy na srozumitelnost kódu, aby mohl být použit jako učební pomůcka.

Literatura

[1] Auger, A.; Hansen, N.: A restart CMA evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, ročník 2, IEEE, 2005, s. 1769–1776.

[2] Chiappori, P.-A.; Levitt, S.; Groseclose, T.: Testing mixed-strategy equilibria when players are heterogeneous: the case of penalty kicks in soccer. *American Economic Review*, 2002: s. 1138–1151.

[3] Daskalakis, C.; Goldberg, P. W.; Papadimitriou, C. H.: The complexity of computing a Nash equilibrium. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, ACM, 2006, s. 71–78.

[4] Hansen, N.: References to CMA-ES applications. [online], 2005.
URL https://www.lri.fr/~hansen/cmaapplications.pdf

[5] Hansen, N.: The CMA Evolution Strategy: A Tutorial. 2011.
URL https://www.lri.fr/~hansen/cmatutorial.pdf

[6] Hrubý, M.: *Doprovodné texty ke kurzu Teorie her*. FIT VUT v Brně, 2010/2011.

[7] Hunter, J. D.: Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, ročník 9, č. 3, 2007: s. 90–95.
URL http://www.matplotlib.org

[8] Jones, E.; Oliphant, T.; Peterson, P.: SciPy: Open source scientific tools for Python. 2001.
URL http://www.scipy.org/

[9] Lemke, C. E.; Howson, J. T., Jr: Equilibrium points of bimatrix games. *Journal of the Society for Industrial & Applied Mathematics*, ročník 12, č. 2, 1964: s. 413–423.

[10] McKelvey, R. D.: A Laipunov Function for Nash Equilibria. Technická zpráva, 1998.

[11] McKelvey, R. D.; McLennan, A.: Computation of equilibria in finite games. *Handbook of computational economics*, ročník 1, 1996: s. 87–142.

[12] McKelvey, R. D.; McLennan, A. M.; Turocy, T. L.: Gambit: Software tools for game theory. 2006.

[13] Nash, J.: Non-cooperative games. *The Annals of Mathematics*, ročník 54, č. 2, 1951: s. 286–295.

[14] Nisan, N.; Roughgarden, T.; Tardos, E.; aj.: *Algorithmic game theory*. Cambridge University Press, 2007.

[15] Nudelman, E.; Wortman, J.; Shoham, Y.; aj.: Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, IEEE Computer Society, 2004, s. 880–887.

[16] Oliphant, T.; aj.: NumPy. 2007.
URL http://www.numpy.org/

[17] Osborne, M. J.; Rubinstein, A.: *Course in game theory*. The MIT press, 1994.

[18] Pavlidis, N.; Parsopoulos, K. E.; Vrahatis, M. N.: Computing Nash equilibria through computational intelligence methods. *Journal of Computational and Applied Mathematics*, ročník 175, č. 1, 2005: s. 113–136.

[19] Von Neumann, J.; Morgenstern, O.: *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[20] Walker, M.; Wooders, J.: Minimax play at Wimbledon. *The American Economic Review*, ročník 91, č. 5, 2001: s. 1521–1538.

[21] Widger, J.; Grosu, D.: Computing equilibria in bimatrix games by parallel support enumeration. In *Parallel and Distributed Computing, 2008. ISPDC'08. International Symposium on*, IEEE, 2008, s. 250–256.

Příloha A

Hry v normální formě

Tabulka A.1: Matching pennies

A / B	heads	tails
heads	1, -1	-1, 1
tails	-1, 1	1, -1

Tabulka A.2: Kámen-nůžky-papír

A / B	kámen	nůžky	papír
kámen	0, 0	1, -1	-1, 1
nůžky	-1, 1	0, 0	1, -1
papír	1, -1	-1, 1	0, 0

Tabulka A.3: Degenerovaná hra

A / B	0	1
0	1, 1	1, 1
1	0, 2	0, 3
2	0, 2	2, 0

Tabulka A.4: Vězňovo dilema

A / B	přiznat se	zatloukat
přiznat se	-10, -10	0, -20
zatloukat	-20, 0	-1, -1