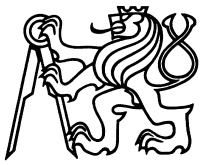
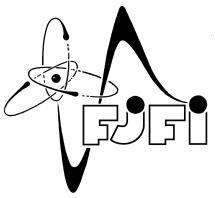


České vysoké učení technické v Praze
Fakulta jaderná a fyzikálně inženýrská



Implicitní spolupráce agentů ve více-účastnickém prostředí

Implicit Cooperation in Multi-agent Environment

Bakalářská práce

Autor: **Aleksej Gaj**

Vedoucí práce: **Ing. Miroslav Kárný, DrSc.**

Akademický rok: 2021/2022

České vysoké učení technické v Praze, Fakulta jaderná a fyzikálně inženýrská

Katedra: matematiky

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Aleksej Gaj
Studijní program:	Aplikace přírodních věd
Studijní obor:	Matematické inženýrství
Studijní zaměření:	Aplikované matematicko-stochastické metody
Název práce (česky):	Implicitní spolupráce agentů ve více-účastnickém prostředí
Název práce (anglicky):	Implicit cooperation in multi-agent environment

Pokyny pro vypracování:

1) Seznamte se s bayesovským odhadováním parametrů markovských modelů pro případ konečného počtu stavů a akcí [2].
2) Seznamte se s problémem rozhodování za neurčitosti [1] a s principy plně pravděpodobnostního návrhu rozhodovacích pravidel [3].
3) Navrhněte plně pravděpodobnostního učícího se agenta.
4) Navrhněte postup implicitní kooperace agentů založený na sdílení modelů či rozhodovacích pravidel, které používají.
5) Navržený postup simulačně otestujte a zhodnoťte výsledky.

Doporučená literatura:

1) M. Puterman, Markov decision processes. John Wiley and Sons, 1994.

2) V. Peterka, Bayesian System Identification. In 'P. Eykhoff, Trends and Progress in System Identification', Pergamon Press, Oxford, 1981, 239-304.

3) M. Kárný, Towards fully probabilistic control design. Automatica 32(12), 1996, 1719-1722.

4) M. Kárný, F. Hůla, Fusion of Probabilistic Unreliable Indirect Information into Estimation Serving to Decision Making. Int. Journal of Machine Learning and Cybernetics, 2021, doi = 10.1007/s13042-021-01359-9.

Jméno a pracoviště vedoucího bakalářské práce:

Ing. Miroslav Kárný, DrSc.
ÚTIA AVČR, v.v.i., Pod Vodárenskou věží 4, , 182 08 Praha 8

Jméno a pracoviště konzultanta:

Datum zadání bakalářské práce: 31.10.2021

Datum odevzdání bakalářské práce: 7.7.2022

Doba platnosti zadání je dva roky od data zadání.

V Praze dne 19. října 2021

.....
garant oboru

.....
vedoucí katedry

.....
děkan

Poděkování:
Chtěl bych zde poděkovat především svému školiteli Ing. Miroslavu Kárnému, DrSc. za pečlivost, ochotu, vstřícnost a odborné i lidské zázemí při vedení mé bakalářské práce. Tato práce byla podpořena projektem MŠMT LTC18075 *Distributed rational decision making: cooperation aspects* a EU-COST Action CA16228.
Dále děkuji své rodině (a zvlášť mé babičce) za podporu a zázemí během mého studia.

Čestné prohlášení:
Prohlašuji, že jsem tuto práci vypracoval samostatně a uvedl jsem všechnu použitou literaturu a software.

V Praze dne 5. ledna 2022 Aleksej Gaj

Název práce:

Implicitní spolupráce agentů ve více-účastnickém prostředí

Autor: Aleksej Gaj

Obor: Matematické inženýrství

Zaměření: Aplikované matematicko–stochastické metody

Druh práce: Bakalářská práce

Vedoucí práce: Ing. Miroslav Kárný, DrSc., Ústav Teorie informace a automatizace, AVČR, v.v.i.

Abstrakt: Tato bakalářská práce se zabývá problémem rozhodování za neurčitosti. Konkrétně zaměřuje se na implicitní kooperaci rozhodovacích agentů ve více–agentním prostředí. Metodologicky jsou to nezávislé bayesovské bytosti, které používají plně pravděpodobnostní návrh rozhodovacích strategií. Agenti kooperují přes sdílení svých rozhodovacích pravidel z předchozího kroku. Pravidlo, které poskytl soused, agent používá jako externí prediktor pro obohacení svých představ o sousedovi. Toto řešení bylo ilustrováno na Nashově obchodní hře, která je abstraktním modelem obchodování. Získané výsledky ukazují, že implicitní kooperace: i) vede ke zvýšení úspěšnosti hry, ii) přináší větší individuální zisky hráčům.

Klíčová slova: bayesovské učení, implicitní spolupráce, pravděpodobnostní návrh rozhodovacích pravidel, teorie rozhodování, více–agentní systémy

Title:

Implicit Cooperation in Multi-agent Settings

Author: Aleksej Gaj

Abstract: This bachelor's thesis addresses a problem of sequential decision making (DM) under uncertainty. In particular, it focuses on the implicit cooperation of DM agents in multi-agent environment. Methodologically, they are self-interested independent Bayesian agents that use fully probabilistic design to find optimal DM strategy. The agents cooperate via sharing their DM rules from the previous decision epoch. The neighbour's DM rule, gathered by the cooperating agent, is used as an external predictor to enrich the agent's knowledge about the neighbour. We illustrate the approach by applying the designed agent to the Nash demand game, which is an abstract model of bargaining. The results indicate that the established implicit negotiation: i) results in enhancing success rate of the game, and, ii) brings higher individual profits to the players.

Key words: implicit cooperation, multi–agent systems, Bayesian learning, fully probabilistic design, decision–making theory

Obsah

Úvod

Každý den řešíme tisíce rozhodovacích úloh, aniž bychom si to nějak uvědomovali. V podstatě v každém okamžiku čelíme jistému rozhodnutí. A jelikož není nijak běžné vracet se zpátky v čase (abychom napravili špatné rozhodnutí včetně důsledků), výsledné rozhodnutí musí být definitivní a optimální (resp. maximálně blízké k optimálnímu).

Existují různé přístupy k řešení tohoto problému v různých oblastech (a tedy i různé názvosloví). Pro příklad uveďme v teorii rozhodování přístup Markovova rozhodovacího procesu (viz sekce 1.3) [24], metodologii plně pravděpodobnostního návrhu (viz sekce 1.4) [9], zpětnovazební učení [28] a další.

Teorie rozhodování má široké spektrum aplikací, např. v robotice, umělé inteligenci, systémech člověk–stroj, zpracování dat, ekonomii, zdravotnictví, atd. Příklady viz [30], [2], [25].

Souvisejícím a mnohem složitějším problémem je návrh rozhodovacích algoritmů, uvažujících několik interagujících agentů ve stejném nebo částečně se překrývajícím prostředí. Tady vzniká celá řada úloh, např.:

- jak má agent uzpůsobit své chování tak, aby zohlednil nejen přítomnost ostatních agentů, ale i informace, které ostatní agenti s ním sdílí (své cíle, svá pozorovaní, ...). A zde hned vzniká podotázka, jaká kooperace je ve prospěch agenta a jak toto posoudit.

- jaké bude výsledné chování skupiny agentů jako celku, tzv. skupinové chování (*angl.* emergent behaviour). Často pro jejich řešení jsou často používány metody, které byly vyvinuty ve fyzice, např. Isingův model [7], nebo jiné přístupy [4], [14], [16].

Pokud je v soustavě vícero agentů, jsou dva způsoby, jak lze provádět rozhodování:

- *centralizované rozhodování* – existuje právě jeden agent-moderátor, zpravidla mající největší pravomoce a přístup k veškerým informacím ostatních agentů (modely, cíle). tento agent provádí optimalizaci a ostatní agenty koordinuje (je např. předmětem výzkumu sociodynamických systémů)

- *distribuované rozhodování* – každý z agentů provádí svou vlastní optimalizaci (agenti mohou být všichni strukturálně stejní, ale nemusí), přičemž má zohlednit výše zmíněné problémy plynoucí z interakcí s druhými agenty.

Předmětem této práce je *distribuované rozhodování* s cílem navržení algoritmu pro optimalizaci strategie chování agenta za podmínky neurčitosti a v přítomnosti jiných (s ním interagujících) agentů.

Co je rozhodovací úloha? Představme si dvojici: *systém* a *agent* (ať už lidská bytost, počítač nebo algoritmus, nebo jejich kombinace). Systém je část světa, kterou agent pozoruje a vzhledem ke kterému má určitý *cíl*. Cílem může být jak ovlivnit systém, tak se i něco o

něm naučit. Agent přitom může vykonávat akce, kterými ovlivní systém nebo svůj stav znalostí ohledně systému.

Pro dosažení svého cíle agent má na základě svých preferencí, pozorování a dalších dostupných znalostí navrhnout a zrealizovat takovou posloupnost dostupných akcí, která naplní jeho cíl.

Příklad. Uveďme názorný příklad, ilustrující výše zavedené pojmy.

Představme si fotbalistu na hřišti s míčem. Systémem je soubor objektů $\{\text{hřiště}, \text{brána}, \text{míč}\}$, agentem fotbalista. Jeho cíl je kopnout míč tak, aby letěl žádaným směrem. Tím vzniká rozhodovací úloha, jejímž předmětem je zvolit směr a velikost síly úderu do míče. Akcí pak bude kopnout do míče s vypočtenou silou a navrženým směrem. Ovšem i v případě optimální volby akce pohyb míče může být ovlivněn dalšími vlivy, např. kvalitou povrchu, větrem, nepřesností provedení akce (to vše je ztíženo neurčitostí, doprovázející každou úlohu).

Nyní si představme složitější úlohu: fotbalistu během zápasu. V tomto případě je systém tvořen nejen fotbalovým hřištěm s míčem a branami, ale i ostatními hráči (spoluhráči i protivníky). *Množina možných akcí* může vypadat například takto: pohyb po hřišti ve směru x s rychlostí v; a kopnutí do míče ve směru x se silou f, kde množiny možných hodnot x, v, f jsou dány fyzikálními možnostmi. *Množina možných stavů* systému je mnohem bohatší, výčtem prvků ji v podstatě[1] nelze zapsat. Zjednodušeně to může být soubor polohových vektorů míče a hráčů v čase, vlhkost trávníku, povětrnostní vlivy, apod. Cílem fotbalisty je dostat míč do brány a vyhnout se útokům protivníka, tj. hledá posloupnost takových akcí, která jeho cíl splní.

Úloha se značně zkomplikuje pohybem na hřišti protihráčů a spoluhráčů, kteří mají podstatný vliv na strategii agenta (fotbalisty). Pak agent musí zohlednit také záměry ostatních hráčů, se kterými bude kooperovat. Další komplikace vznikne, pokud agent má vlastní nepřesnost (tj. provádí své akce jen s určitou přesností – například méně zkušený hráč).

Výklad je veden následovně. Kapitola 1 obsahuje stručný přehled použité teorie. Kapitola 2 se zaměří na konstrukci agenta, který pro svá rozhodnutí využívá princip plně pravděpodobnostního návrhu a bayesovské odhadování parametrů. Kapitola 3 pojednává o systému více agentů, kteří kooperují mezi sebou. Kapitola 4 obsahuje použití navrženého adaptivního kooperujícího agenta v Nashově obchodní hře a analýzu obdržených výsledků. Práci uzavírají shrnutí získaných poznatků a nástin případných budoucích směrů zkoumání.

[1] Záleží, zda uvažujeme např. polohu míče jako spojitou funkci nebo diskrétní, pro kterou zavedeme síť souřadnic s určitým rozlišením.

Použité značení

Zkratka	*Význam*
n.v.	náhodná veličina
MRP	Markovův rozhodovací proces
PPN	Plně pravděpodobnostní návrh
KLD	Kullback–Leiblerova divergence
NOH	Nashova obchodní hra

Tabulka 1: Tabulka použitých zkratek

Symbol	*Stručný vysvětlivek*
$\mathbb{R}$	množina reálných čísel
$\mathbb{R}^+$	množina reálných kladných čísel
$\mathbb{N}$	množina přirozených čísel
$\mathbf{X}$, $\mathbf{A}$, $\mathbf{S}$, $\mathbf{\Omega}$, ...	množiny
$\mathbf{T}$	konečná množina časů, $\mathbf{T} \subset \mathbb{N}$
$\langle a, b \rangle$	uzavřený interval od $a \in \mathbb{R}$ do $b \in \mathbb{R}$, $a \leq b$
$a := b$	definice přiřazením: objekt a je definován jako b
$\mathrm{D}(f \Vert g)$	Kullback–Leiblerova divergence mezi pravděpodobnostními funkcemi f a g
$\mathcal{X}$, $\mathcal{A}$, ...	náhodné veličiny
A	σ–algebra
x_t	hodnota diskrétní veličiny x v diskrétním čase t, $x_t \in \mathbf{X}$
x^t	uspořádaná t–tice $x^t := (x_1, x_2, x_3, \ldots, x_\tau, \ldots, x_t)$
$\mathbb{E}[\mathcal{X}]$	střední hodnota náhodné veličiny $\mathcal{X}$
$p(x, y)$	sdružená hustota pravděpodobnosti n.v. $\mathcal{X}$ a $\mathcal{Y}$
$p(x \vert y)$	podmíněná hustota pravděpodobnosti n.v. $\mathcal{X}$ za podmínky $\mathcal{Y} = y$
b	chování uzavřené smyčky do času T, $b := b^T$
π	strategie agenta do času T, $\pi := \pi^T$

Tabulka 2: Tabulka použitého značení

Úmluva. Napříč celým textem *zelenou kurzívou* jsou vyznačeny pojmy uvnitř definic, které jsou zde definovány. Znění vět a předpokladů je naopak na zeleném podkladu.

Diskrétní a spojité veličiny. V této práci všechny veličiny jsou diskrétní. Výjimku tvoří pouze parametr systému (viz Sekce 1.5). V příslušném místě na to ještě jednou upozorníme.

Kapitola 1

Přehled použité teorie

V této kapitole zmíníme základní pojmy, vztahy a věty, které použijeme v dalších kapitolách.

1.1 Užívané pojmy

Definice 1 (Pravděpodobnost)**.** Mějme množinu elementárních jevů $\mathbf{\Omega} \neq \varnothing$ a σ–algebru podmnožin $\mathbf{\Omega}$ nad touto množinou, kterou označíme A. Pak libovolné zobrazení $P : \mathsf{A} \to \mathbb{R}$, pro které platí:

1. axiom nezápornosti: $\forall \mathbf{A} \subset \mathsf{A} : P(\mathbf{A}) \geqslant 0$
2. axiom aditivity: $(\mathbf{A}_i \cap \mathbf{A}_j = \varnothing, i \neq j) \Rightarrow P(\sum_i \mathbf{A}_i) = \sum_i P(\mathbf{A}_i)$
3. axiom finitnosti: $P(\mathsf{A}) = 1$

nazvu *pravděpodobnostní mírou* (pravděpodobností) na měřitelném prostoru $(\mathbf{\Omega}, \mathsf{A}, P)$. Trojici $(\mathbf{\Omega}, \mathsf{A}, P)$ nazveme *pravděpodobnostním prostorem.*

Exaktní definice, zavedení a vlastnosti viz [17]. V této práci nejčastěji budeme operovat s pravděpodobnostní funkcí.

Definice 2 (Marginální hustota pravděpodobnosti)**.** Mějme sdružený vektor náhodných veličin $(\mathcal{X}, \mathcal{Y}) \sim P^{\mathcal{X},\mathcal{Y}}$ s hustotou pravděpodobnosti $p_{\mathcal{X},\mathcal{Y}}$. Pak:

- $\mathcal{X}$ má hustotu pravděpodobnosti $p_{\mathcal{X}}(x) = \int_{y \in \mathbf{Y}} p_{\mathcal{X},\mathcal{Y}}(x, y) \mathrm{d}y$
- $\mathcal{Y}$ má hustotu pravděpodobnosti $p_{\mathcal{Y}}(y) = \int_{x \in \mathbf{X}} p_{\mathcal{X},\mathcal{Y}}(x, y) \mathrm{d}x$.

Poznámka. Definici 2 lze použít i pro vícerozměrné náhodné veličiny, viz [17].

Definice 3 (Podmíněná hustota pravděpodobnosti)**.** Mějme sdružený vektor náhodných veličin $(\mathcal{X}, \mathcal{Y}) \sim P^{\mathcal{X},\mathcal{Y}}$ s hustotou pravděpodobnosti $p_{\mathcal{X},\mathcal{Y}}$ a nechť platí $p_{\mathcal{Y}}(y) \neq 0$. Potom funkci

$$p_{\mathcal{X}|\mathcal{Y}}(x) = \frac{p_{\mathcal{X},\mathcal{Y}}(x, y)}{p_{\mathcal{Y}}(y)} \quad \forall x \in \mathbb{R} \tag{1.1}$$

nazveme *podmíněnou hustotou pravděpodobnosti $\mathcal{X}$ za podmínky $\mathcal{Y}$* .

Věta 1 (Bayesův vzorec pro hustoty pravděpodobnosti). Mějme dvě náhodné veličiny $\mathcal{X}$,$\mathcal{Y}$ se sdruženou hustotou pravděpodobnosti $p_{\mathcal{X},\mathcal{Y}}$. Pak platí:

$$p_{\mathcal{X}|\mathcal{Y}}(x|y) = \frac{p_{\mathcal{Y}|\mathcal{X}}(y|x)\, p_{\mathcal{X}}(x)}{\int_{\mathbf{X}} p_{\mathcal{Y}|\mathcal{X}}(y|x)\, p_{\mathcal{X}}(x)\mathrm{d}x} \tag{1.2}$$

Věta 2 (Řetězové pravidlo pro hustoty pravděpodobnosti). Nechť $(\mathcal{X},\mathcal{Y})$ je dvojice náhodných veličin, mající sdružené rozdělení $p_{\mathcal{X},\mathcal{Y}}$. Pak platí:

$$p_{\mathcal{X},\mathcal{Y}}(x,y) = p_{\mathcal{X}|\mathcal{Y}}(x|y)\, p_{\mathcal{Y}}(y). \tag{1.3}$$

Důkaz. Jedná se o prostý důsledek Definice 3. □

Poznámka (Značení hustot pravděpodobnosti). Doposud jsme používali značení $p_{\mathcal{X}}(x)$ pro hustotu pravděpodobnosti (pravděpodobnost) náhodné veličiny $\mathcal{X}$ a realizaci této veličiny x. Odteď a dále budeme dolní index, značící příslušnou náhodnou veličinu, vynechávat, a budeme tedy psát $p(x)$. To, o kterou náhodnou veličinu jde, budeme rozlišovat podle argumentů funkce p.

Poznámka (Zobecnění Věty 2). Větu 2 lze upravit i pro více náhodných veličin: $(\mathcal{X}_1, \ldots, \mathcal{X}_n)$. Případně celou rovnost (1.3) podmínit takto: $p(x,y|z) = p(x|y,z)p(y|z)$. Zároveň si lze povšimnout, že Definice 2, 3, Věty 1 a 2 platí i pro diskrétní případ (pravděpodobnostní funkce).

Definice 4 (Kullback–Leiblerova divergence)**.** Mějme dvě hustoty pravděpodobnosti stejného argumentu x, $p(x)$ a $q(x)$. Pak je *Kullback–Leiblerova divergence* (dále jen KLD) definována takto:

$$\mathrm{D}\left(p\|q\right) := \int_{x\in\mathbf{X}} p(x) \ln\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x \tag{1.4}$$

Poznámka (Tvar KLD pro diskrétní náhodné veličiny). V případě diskrétní náhodné veličiny vzorec (1.4) má tvar:

$$\mathrm{D}\left(p\|q\right) = \sum_{x\in\mathbf{X}} p(x) \ln\left(\frac{p(x)}{q(x)}\right). \tag{1.5}$$

Věta 3 (Vlastnosti KLD). Nechť $p(x)$ a $q(x)$ jsou dvě hustoty pravděpodobnosti stejného argumentu x. Pak platí:

1. $\mathrm{D}\left(p\|q\right) \geqq 0$
2. $\mathrm{D}\left(p\|q\right) = 0 \Leftrightarrow p = q$ skoro všude
3. $\mathrm{D}\left(p\|q\right) = +\infty \Leftrightarrow p \neq 0 \wedge q = 0$ na množině kladné míry
4. $\mathrm{D}\left(p\|q\right) \neq \mathrm{D}\left(q\|p\right)$

Dále pro KLD neplatí trojúhelníková nerovnost.

Důkaz. Zde bez důkazu, lze nalézt v [3]. □

Poznámka. Výhodou KLD je, že zohledňuje celý tvar pravděpodobnostní funkce a nejenom její momenty, viz výpočet KLD pro dvě různá normální rozdělení v A.2.

Poznámka. Někdy je KLD též nazývána *relativní entropie* nebo *divergenční míra*. Obecně je to způsob měření odlišnosti mezi dvěma pravděpodobnostními mírami. Podrobněji viz [3].

V následujících kapitolách budeme potřebovat pár speciálních funkcí. Jedná se o funkce Beta a Gamma, někdy též nazývané Eulerův integrál prvního a druhého druhu[1].

Definice 5 (Beta funkce)**.**

$$B(x,y) := \int_0^1 t^{x-1}\,(1-t)^{y-1}\,\mathrm{d}t \qquad \forall x,y \in \mathbb{R}^+ \tag{1.6}$$

Definice 6 (Gamma funkce)**.**

$$\Gamma(z) := \int_0^{+\infty} t^{z-1}e^t \mathrm{d}t \qquad \forall z \in \mathbb{R}^+ \tag{1.7}$$

Poznámka. S použitím (1.7) je možno Beta funkci zobecnit pro $k \in \mathbb{N}$ kladných argumentů vztahem

$$B(\alpha_1,\alpha_2,\ldots,\alpha_k) := \frac{\prod_{i=1}^k \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^k \alpha_i\right)} \tag{1.8}$$

Definice 7 (Kroneckerova delta funkce)**.** Kroneckerova delta funkce je definována jako 0 pro různé objekty na vstupu a jako 1, pokud objekty na vstupu jsou totožné, tedy:

$$\begin{array}{l} \delta(i,j) = 0 \Leftrightarrow i \neq j \\ \delta(i,j) = 1 \Leftrightarrow i = j \end{array} \quad \forall i,j \tag{1.9}$$

Definice 8 (Dirichletovo rozdělení)**.** Mějme k–tici nezáporných čísel $(x_i)_{i=1}^k$ pro kterou platí: $\sum_{i=1}^k x_i = 1$. Pak k–tice nezáporných čísel $(\alpha_i)_{i=1}^k$ zaručuje, že

$$\mathrm{Dir}(x_1,x_2,\ldots,x_k|\alpha_1,\alpha_2,\ldots,\alpha_k) := \frac{1}{B(\alpha_1,\alpha_2,\ldots,\alpha_k)} \prod_{i=1}^k x_i^{\alpha_i-1} \tag{1.10}$$

je hustotou pravděpodobnosti. Tuto funkci nazveme *Dirichletovo rozdělení* .

Poznámka. S použitím vzorce (1.8) lze výraz (1.10) upravit:

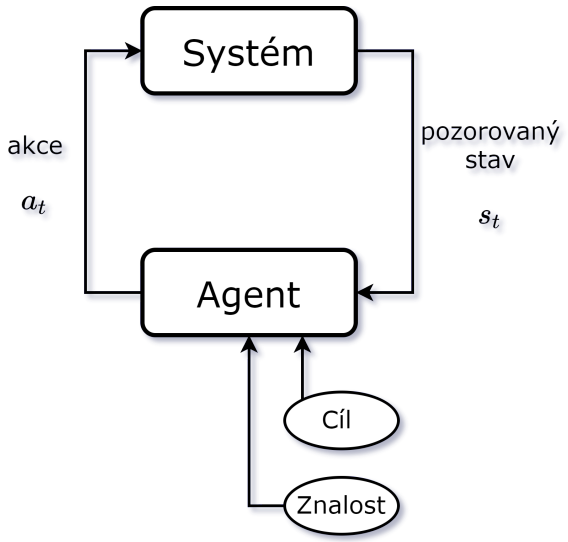
$$\mathrm{Dir}(x_1,x_2,\ldots,x_k|\alpha_1,\alpha_2,\ldots,\alpha_k) = \prod_{i=1}^k \frac{x_i^{\alpha_i-1}}{\Gamma(\alpha_i)}\,\Gamma\left(\sum_{i=1}^k \alpha_i\right). \tag{1.11}$$

[1] Historická poznámka o názvosloví v příloze, viz A.1.

1.2 Rozhodování v uzavřené smyčce

Uvažujeme rozhodovací úlohu (viz str. 13) v tzv. *uzavřené smyčce* (*angl.* closed loop) viz Obrázek 1.1. V každém čase $t \in \mathbf{T}$, $\mathbf{T} := \{0, 1, \ldots T-1\}$ systém generuje stav $s_t \in \mathbf{S}$, který agent pozoruje. Agent na základě pozorovaných dat, svého cíle a apriorních znalostí vygeneruje akci $a_t \in \mathbf{A}$, která ovlivní stav systému v dalším časovém okamžiku, tj. $s_{t+1} \in \mathbf{S}$.

Poznamenejme, že názvosloví se různí: systém je také někdy nazýván světem či prostředím; agent regulátorem, aj.



Obrázek 1.1: Uzavřená smyčka obecně

Poznámka (Agentovo pozorování a stavy systému). V obecném případě agent nemusí pozorovat přímo celý stav systému s_t. Proto se často využíván pojem *pozorovaná veličina* (*angl.* observation), která v obecném případě nese informaci o stavu systému. Pozorování tedy může:

- obsahovat pouze část skutečného stavu systému,
- vůbec neobsahovat skutečný stav, ale záviset na něm,
- být zatíženo chybou (např. chybou měřicího přístroje, šumem informačního kanálu aj.).

V této práci předpokládáme, že agent pozoruje skutečný stav systému.

Definice 9 (Chování uzavřené smyčky)**.** *Chováním uzavřené smyčky* (*angl.* closed–loop behaviour) se nazývá posloupnost b, která je definována následovně:

$$b^T := (s_T,\, a_{T-1}, s_{T-1},\, \ldots,\, a_t, s_t,\, \ldots,\, a_1, s_1)\,, \tag{1.12}$$

kde a_t jsou akce agenta v čase $t \in \mathbf{T}$ a s_t jsou stavy systému v čase $t \in \mathbf{T}$. Počáteční akce a_0 a počáteční stav s_0 jsou známy a přítomny v b^T implicitně.

Poznámka. Je tu jistá analogie s přístupem teoretické fyziky: dalo by se říct, že prostor všech možných chování uzavřené smyčky je fázovým prostorem pro danou úlohu. Chování uzavřené smyčky totiž představuje posloupnost stavů a akcí v čase pro všechny časy, což je analogií fázové trajektorie.

Pro popis chování uzavřené smyčky používáme pravděpodobnostní přístup. Budeme tedy modelovat chování uzavřené smyčky pomocí sdružené pravděpodobnostní funkce $p(b^T)$.

Poznámka. Pro lepší přehlednost zápisu budeme index T u chování uzavřené smyčky vynechávat: $b := b^T$. Známé počáteční podmínky uvažujeme pouze implicitně.

Pro $p(b)$ platí:

$$\begin{aligned} p(b) &\overset{(1.12)}{=} p\left(s_T, a_{T-1}, s_{T-1}, \ldots, a_1, s_1\right) \\ &\overset{(1.3)}{=} \prod_{t\in\mathbf{T}} p\left(s_{t+1}, a_t | a^{t-1}, s^t\right) \\ &\overset{(1.3)}{=} \prod_{t\in\mathbf{T}} p\left(s_{t+1} | a^t, s^t\right) p\left(a_t | a^{t-1}, s^t\right) \end{aligned} \tag{1.13}$$

Definice 10 (Rozhodovací pravidlo)**.** *Rozhodovacím pravidlem* agenta nazveme hustotu pravděpodobnosti, určující pravděpodobnost (zvolení) akce a_t za podmínky všech předchozích stavů a akcí, formálně:

$$p\left(a_t | s^t, a^{t-1}\right).$$

Definice 11 (Strategie)**.** *Strategií* [2] agenta nazveme posloupnost jeho rozhodovacích pravidel v čase a označíme ji π^T. Platí tedy:

$$\pi^T = \left(p\left(a_t | s^t, a^{t-1}\right)\right)_{t\in\mathbf{T}}.$$

Poznámka. Pro přehlednost zápisu budeme index T u strategie vynechávat: $\pi := \pi^T$.

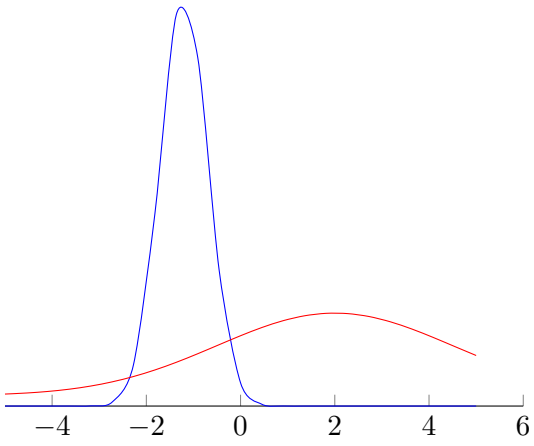
Pro členy na pravé straně v rovnosti (1.13) máme interpretaci:

$$p(b) = \prod_{t\in\mathbf{T}} \underbrace{p\left(s_{t+1} | a^t, s^t\right)}_{\text{model systému}} \quad \underbrace{p\left(a_t | s^t, a^{t-1}\right)}_{\text{rozhodovací pravidlo}}. \tag{1.14}$$

Řešení rozhodovací úlohy spočívá v návrhu algoritmu pro volbu strategie optimální z hlediska cílů agenta. Formulace cílů je samostatný a velmi obsáhlý problém (oblast preference elicitation, [11]). V této práci použijeme pravděpodobnostní popis cílů agenta.

Představme si rozhodovací úlohu v rovinném obrázku.

[2] Někdy též *politika*, *angl.* policy.



Obrázek 1.2: Dva možné modely uzavřené smyčky (1.14)

Zde jsou zobrazeny dvě možné pravděpodobnostní distribuce nad prostorem všech možných chování uzavřené smyčky: $p^{\pi_1}(b)$ (modrá) a $p^{\pi_2}(b)$ (červená)[3]. Předpokládejme, že $b_{\text{chci}=0}$ je chování, kterého chceme dosáhnout. Chceme tedy zvolit takovou hustotu pravděpodobnosti ze dvou možných, která zaručí co nejvyšší pravděpodobnost realizace b_{chci}. Kterou tedy?

Na to není jednoznačná odpověď. Možné způsoby volby:

1. zvolit $p^{\pi_i}(b)$ takovou ($i \in \{1, 2\}$), která minimalizuje vzdálenost střední hodnoty b od b_{chci}:

$$\min_{\pi_i \in \mathbf{\Pi}} |\mathbb{E}^{\pi}[b] - b_{\text{chci}}|$$

2. zvolit $p^{\pi_i}(b)$ takovou ($i \in \{1, 2\}$), která maximalizuje pravděpodobnost:

$$\max_{\pi_i \in \mathbf{\Pi}} p^{\pi_i}(b_{\text{chci}})$$

Obecnější přístup (mj. zahrnuje výše uvedené možnosti) je přístup minimalizující očekávanou ztrátu[4]:

$$\min_{\pi_i \in \mathbf{\Pi}} \int_{b \in \mathbf{B}} L(b) p^{\pi_i}(b) \mathrm{d}b. \tag{1.15}$$

Pro případ Markovova rozhodovacího procesu (viz Sekce 1.3) ztrátová funkce L nezávisí na strategii (rozhodovacích pravidlech). Nicméně i tak je tato minimalizace v obecném případě dost obtížná.

Uvažovaný alternativní přístup, tzv. *plně pravděpodobnostní návrh* (dále zkráceně jen PPN), volí:

$$\pi^{opt} = \arg\min_{\pi \in \mathbf{\Pi}} \mathrm{D}\left(p^{\pi}(b) \| {}^{I}p(b)\right),$$

kde pravděpodobnostní funkce ${}^{I}p(b)$ je z pohledu agenta ideální model chování systému a $\mathbf{\Pi}$ je množina přípustných strategií (v příkladě na Obrázku 1.2 je $\mathbf{\Pi} = \{\pi_1, \pi_2\}$).

[3] π_1 je strategie, zajišťující modré chování uzavřené smyčky a π_2 je strategie, zajišťující červené (Obrázek 1.2).

[4] tento přístup je jádrem Markovova rozhodovacího procesu, viz Definice 12.

1.3 Markovův rozhodovací proces

Markovův rozhodovací proces [24] (*angl.* Markov decision process), dále jen MRP, se hojně využívá k řešení rozhodovacích úloh. Užívá součtovou ztrátu nezávislou na strategii.

Definice 12 (Markovův rozhodovací proces)**.** Diskrétní Markovův rozhodovací proces je definován jako pětice $(\mathbf{T}, \mathbf{S}, \mathbf{A}, L, p)$, kde:

- $\mathbf{T}$ je konečná množina časových indexů, $\forall t \in \mathbf{T}$, $\mathbf{T} \subset \mathbb{N}$,
- $\mathbf{S}$ je konečná a diskrétní množina všech stavů systému, $s_t \in \mathbf{S} \quad (\forall t \in \mathbf{T} \cup \{T\})$,
- $\mathbf{A}$ je konečná a diskrétní množina všech akcí agenta, $a_t \in \mathbf{A} \quad (\forall t \in \mathbf{T})$,
- $p : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \langle 0; 1 \rangle$ je pravděpodobnostní model systému [5],
- $L : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to \mathbb{R}$ je ztrátová funkce (*angl.* loss function).

Poslední dvě zobrazení v Definici 12 lze intuitivně interpretovat takto: model systému, který je popsán tabulkou přechodových pravděpodobností $p\left(s_{t+1}|a_t, s_t\right)$, udává, s jakou pravděpodobností se systém ocitne ve stavu s_{t+1}, pokud v minulém časovém kroku byl ve stavu s_t a agent provedl akci a_t. *Ztrátová funkce* je vlastně kritérium nespokojenosti agenta s aktuální situací (tj. s_{t+1}, s_t, a_t) a vyjadřuje cíle agenta: najít strategii vedoucí k menší součtové ztrátě.

MRP stojí na tzv. *Markovově předpokladu*:

Předpoklad 1 (Markovův předpoklad). Stav systému v čase $t + 1$ závisí jen na stavu a akci předchozího času t. Formálně tedy

$$p\left(s_{t+1}|a_t, s_t\right) = p\left(s_{t+1}|a^t, s^t\right). \tag{1.16}$$

Jinými slovy, agent užívá model, který má *paměť rovnou jedné* - pamatuje si jen jeden krok do minulosti. S použitím Předpokladu 1 pak i optimální strategie má tvar: $\pi = \left(p\left(a_t|s_t\right)\right)_{t\in\mathbf{T}}$.

Poznámka. Pozor, nezaměňovat MRP s Markovovým procesem. Markovův *řetězec* (nebo také Markovův *proces*) je náhodný proces, popis je realizován pouze pomocí stavů systému. Může být jak diskrétní (*angl.* discrete–time Markov chain), tak i spojitý (*angl.* continuous–time Markov chain). Žádné rozhodování tu neprobíhá. Markovův *rozhodovací proces* je zobecnění Markovova procesu, je obohacen o možnost ovlivňovat stavy pomocí volených akcí (a tím vzniká prostor pro rozhodovací úlohu)[6].

Řešení MRP je hledáno jako minimum očekávané ztráty:

$$\pi^{opt} = \arg\min_{\pi\in\mathbf{\Pi}} \mathbb{E}^{\pi}\left[\sum_{t\in\mathbf{T}} L\left(s_{t+1}, a_t, s_t\right)\right], \tag{1.17}$$

kde $\mathbf{\Pi}$ je množina všech strategií (cf. (1.15)). Pro L nezávislé na $\pi \in \mathbf{\Pi}$ je optimální strategie deterministická [24].

[5] V uvažovaném diskrétním případě je ve tvaru tabulky přechodových pravděpodobností $p\left(s_{t+1}|a_t, s_t\right)$.

[6] V některých česky psaných zdrojích je pojem řetězce používán pro diskrétní čas a pojem procesu pro čas spojitý.

1.4 Plně pravděpodobnostní návrh

Vzorec (1.13) dává pravděpodobnostní popis uzavřené smyčky pro všechny časy $t \in \mathbf{T}$. V uzavřené smyčce má agent svůj cíl (preference) vzhledem k systému a svá omezení (viz Sekce 1.2). Agent pro popis svého cíle využije pravděpodobnostní model podobný (1.13), tj. ${}^{I}p(b)$. Vystihuje tak pro něj žádoucí chování uzavřené smyčky.

Použitím rozkladu jako ve vzorci (1.13) dostaneme:

$$ {}^{I}p(b) \overset{(1.3)}{=} \prod_{t\in\mathbf{T}} {}^{I}p\left(s_{t+1}|a^t, s^t\right) {}^{I}p\left(a_t|a^{t-1}, s^t\right) \tag{1.18}$$

Aplikací Předpokladu 1 se výsledný vztah výrazně zjednoduší:

$$ {}^{I}p(b) \overset{(1.3)}{=} \prod_{t\in\mathbf{T}} {}^{I}p\left(s_{t+1}|a_t, s_t\right) {}^{I}p\left(a_t|s_t\right). \tag{1.19}$$

V (1.19) výrazy na pravé straně mají následující význam:

- ${}^{I}p\left(s_{t+1}|a_t, s_t\right)$ – tzv. *ideální model*: model systému, který popisuje ideální chování systému z hlediska cílů agenta.
- ${}^{I}p\left(a_t|s_t\right)$ – tzv. *ideální rozhodovací pravidlo*: rozhodovací pravidlo, které agent preferuje (zohledňuje jeho cíle, omezení, ...).

Pravděpodobnostní funkce (1.19) popisuje *ideální chování uzavřené smyčky* do času T.

Definice 13 (Plně pravděpodobnostní návrh - PPN)**.** Mějme model systému $p\left(s_{t+1}|a_t, s_t\right)$ a pravděpodobnostní funkci, popisující ideální (z hlediska cílů agenta) chování uzavřené smyčky ${}^{I}p(b)$. *PPN–optimální strategií* nazveme takovou strategii, která zajišťuje minimální odlišnost mezi chováním uzavřené smyčky a ideálním chováním uzavřené smyčky, tj.

$$ \pi^{opt} \in \arg\min_{\pi_i\in\mathbf{\Pi}} \mathrm{D}\left(p^{\pi_i}(b) \| {}^{I}p(b)\right), \tag{1.20}$$

kde $\mathrm{D}(\cdot\|\cdot)$ je Kullback–Leiblerova divergence, viz Definice 4.

Poznámka. Axiomatické zdůvodnění této definice je v [12].

Intuitivně PPN je snaha najít akce, které zajistí, že realita bude maximálně blízká k přání.

Věta 4 (Řešení PPN v diskrétním případě). Nechť $\mathbf{A}$, $\mathbf{S}$ jsou diskrétní množiny a $a_t \in \mathbf{A}, s_t \in \mathbf{S}$ pro $\forall t \in \mathbf{T}$. Nechť máme model systému $p\left(s_{t+1}|a_t, s_t\right)$ a popis ideálního chování uzavřené smyčky ve tvaru pravé strany (1.19).

Pak optimální rozhodovací pravidlo pro čas $t \in \mathbf{T}$ najdeme takto:

$$ p^{opt}(a_t|s_t) = {}^{I}p\left(a_t|s_t\right) \frac{\exp\left(-d(a_t, s_t)\right)}{h(s_t)}, \tag{1.21}$$

kde

$$ d(a_t, s_t) = \sum_{s_{t+1}\in\mathbf{S}} p\left(s_{t+1}|a_t, s_t\right) \ln \frac{p\left(s_{t+1}|a_t, s_t\right)}{{}^{I}p\left(s_{t+1}|a_t, s_t\right) h(s_{t+1})}, \tag{1.22}$$

$$h(s_t) = \sum_{a_t \in \mathbf{A}} {}^{I}p\left(a_t|s_t\right) \exp\left(-d(a_t, s_t)\right) \tag{1.23}$$

Hledáme optimální rozhodovací pravidlo pro každý čas $t \in \mathbf{T}$, procházíme časy "pozpátku", tj. počítáme pomocí tzv. indukce pozpátku (*angl.* backward induction). Výpočet začíná na čase $t = T$ počínaje:

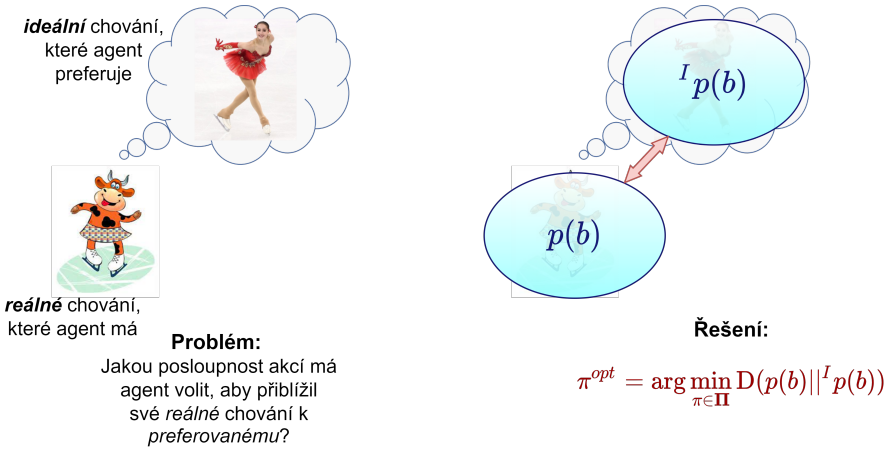
$$h(s_T) = 1 \quad \forall s_T \in \mathbf{S}. \tag{1.24}$$

Optimální strategie je pak posloupnost optimálních rozhodovacích pravidel:

$$\pi^{opt} = \left(p^{opt}(a_t|s_t)\right)_{t \in \mathbf{T}}. \tag{1.25}$$

Důkaz. Zde bez důkazu, lze najít v [27]. □

Základní princip PPN je schematicky zobrazen na Obrázku 1.3[7].



Obrázek 1.3: Ilustrace jak funguje PPN

Uveďme některé přednosti a nedostatky přístupu PPN ve srovnání s MRP.

Přednosti:

- MRP je speciální případ PPN (ukázáno v [12]).
- PPN díky znáhodněnému rozhodovacímu pravidlu (výstupem PPN je v každém čase pravděpodobnostní funkce nad prostorem akcí, (1.21)) řeší problém průzkumu (*exploration*) implicitně. MRP strategie je deterministická (výstupem je optimální akce) a proto MRP potřebuje dodatečné řešení *exploration*-problému.
- PPN je realizací snáze aproximovatelného zobrazení (má explicitní formu π^{opt}).

[7] Zdroje ilustrací: `https://kirill-yasko.com/wp-content/uploads/2010/01/cow_on_ice.jpg`, `https://cdnimg.rg.ru/i/gallery/76a441bd/1_9b58ae1c.jpg`)

Nedostatky:

- Konstrukce ideálního chování systému ${}^{I}p(b)$ je nejednoznačná a dosti technická.
- Není-li poznávání (*angl.* exploration) třeba, pak dosahovaná kvalita řešení PPN je horší, než je možné.

1.5 Bayesovské odhadování parametrů

Pro MRP i PPN je potřeba znát model systému $p\left(s_{t+1}|a_t, s_t\right)$. Ten ovšem k dispozici (v naprosté většině případů) nemáme. Proto budeme chtít ho zkusit vytvořit z toho, co máme - z dat (stavy a akce z předchozích časů) a apriorních znalostí. Ideální by bylo, aby vyhovoval kritériu, které před časem vyslovil S. Hawking ve své knize *"Velkolepý plán"* [5]:

Model je dobrý, když:

1. *Je elegantní.*
2. *Obsahuje jen málo libovolných nebo nastavitelných parametrů.*
3. *Poskytuje podrobné předpovědi o budoucích měřeních, která mohou model potvrdit nebo vyvrátit, podle toho, zda se předpovědi ukážou správné, nebo mylné.*

Při konstrukci takového modelu se opřeme o teorii odhadování (*angl.* estimation theory). Jejím cílem je odhadnout *nepozorované* parametry systému na základě *pozorovaných* veličin či signálů. Existující dva přístupy: *parametrický* a *neparametrický*. První přístup je používán v případě, že je znám matematický model systému a chceme určit jen neznámé parametry tohoto systému. Neparametrický přístup slouží je zkoumání objektů s neznámou strukturou. Tím se dále zabývat nebudeme.

Budeme uvažovat rozšířené chování uzavřené smyčky, které lze zapsat takto:

$$b^T = \left(s_T,\ a_{T-1}, s_{T-1},\ \ldots,\ a_t, s_t,\ \ldots,\ a_1, s_1, \theta\right), \tag{1.26}$$

kde θ je zatím blíže neurčený parametr systému. Jedná se o speciální případ vnitřní veličiny – nemění se totiž v čase.

Cílem agenta je s použitím pouze těchto informací naučit se předpovídat "co bude dál", tj. v jakém stavu $s_{t+1} \in \mathbf{S}$ bude systém v dalším časovém kroku, když jeho minulý stav byl s_t a akce agenta byla a_t. Máme tedy k dispozici data $\left(s^t, a^t\right)$, kde

$$s^t = \left(s_1, s_2, ..., s_t\right), \quad a^t = \left(a_1, a_2, ..., a_t\right), \quad t \in \mathbf{T}.$$

a potřebujeme získat předpověď

$$p(s_{t+1}|a^t, s^t),\ t \in \mathbf{T}. \tag{1.27}$$

Předpoklad 2 (Model systému a jeho parametr). Předpokládáme, že máme parametrizovaný Markovův model

$$p\left(s_{t+1}|a_t, s_t, \theta\right) = p\left(s_{t+1}|a^t, s^t, \theta\right), \tag{1.28}$$

kde parametr systému $\theta \in \mathbf{\Theta}$ je neznámý. V uvažovaném diskrétním případě parametr θ tvoří hodnoty přechodových pravděpodobností

$$p(s_{t+1}|a^t, s^t, \theta) = p(s_{t+1}|a_t, s_t, \theta) \stackrel{\text{def.}}{=} \theta_{s_{t+1}|a_t,s_t} \geq 0,\ \sum_{s_{t+1} \in \mathbf{S}} \theta_{s_{t+1}|a_t,s_t} = 1. \tag{1.29}$$

S modelem (1.28) a s použitím Věty 1 a Definice 3 dostaneme hledanou předpověď

$$p(s_{t+1}|a^t, s^t) = \int_{\theta\in\mathbf{\Theta}} p(s_{t+1}, \theta|a^t, s^t)\mathrm{d}\theta = \int_{\theta\in\mathbf{\Theta}} p(s_{t+1}|a_t, s_t, \theta)p(\theta|a^t, s^t)\mathrm{d}\theta. \tag{1.30}$$

K ní potřebujeme získat aposteriorní hustotu pravděpodobnosti $p(\theta|a^t, s^t)$ pro uvažovaný parametrický model (1.29). K tomu užijeme vyjádření pomocí Kroneckerovy delty (1.9):

$$\theta_{s_{t+1}|a_t,s_t} = \prod_{\tilde{s}\in\mathbf{S}}\prod_{a\in\mathbf{A}}\prod_{s\in\mathbf{S}} \theta_{\tilde{s}|a,s}^{\delta(\tilde{s},s_{t+1})\,\delta(a,a_t)\,\delta(s,s_t)}. \tag{1.31}$$

Začneme tím, že použijeme tedy Bayesův vzorec (1) pro získání rekurze zahrnující do $p(\theta|a^t, s^t)$ nová data. Využijeme přitom řetězového pravidla (1.3)[8]:

$$\begin{aligned}
p(\theta|a^{t+1}, s^{t+1}) &= p(\theta|(a_{t+1}, a^t), (s_{t+1}, s^t)) \\
&\overset{\text{dle (1.2)}}{=} \frac{p(\theta, a_{t+1}, s_{t+1}|a^t, s^t)}{\int_{\theta\in\mathbf{\Theta}} p(\theta, a_{t+1}, s_{t+1}|a^t, s^t)\mathrm{d}\theta} \\
&\overset{\text{dle (1.3)}}{=} \frac{p(a_{t+1}, s_{t+1}|a^t, s^t, \theta)p(\theta|a^t, s^t)}{\int_{\theta\in\mathbf{\Theta}} p(a_{t+1}, s_{t+1}|a^t, s^t, \theta)p(\theta|a^t, s^t)\mathrm{d}\theta} \\
&\overset{\text{dle (1.3)}}{=} \frac{p(s_{t+1}|a^t, s^t, \theta)p(a_{t+1}|a^t, s^{t+1}, \theta)p(\theta|a^t, s^t)}{\int_{\theta\in\mathbf{\Theta}} p(s_{t+1}|a^t, s^t, \theta)p(a_{t+1}|a^t, s^{t+1}, \theta)p(\theta|a^t, s^t)\mathrm{d}\theta}.
\end{aligned} \tag{1.32}$$

Dle Předpokladu 2 první činitel v čitateli posledně získaného zlomku je $p(s_{t+1}|a_t, s_t, \theta)$. Poslední činitel téhož čitatele, $p(\theta|a^t, s^t)$, máme – je to korigovaná apriorní znalost o parametru (viz bayesovský přístup ve statistice). Prostřední činitel, $p(a_{t+1}|a^t, s^{t+1}, \theta)$, je ale pro nás problematický. Totiž agent pro generování svých akcí a_{t+1} nezná parametr systému θ (a tedy nezná ani, jak závisí hodnota parametru na datech). Proto přijmeme následující předpoklad:

Předpoklad 3 (Přirozené podmínky řízení, [23]). Předpokládáme, že

$$p(a_{t+1}|a^t, s^{t+1}, \theta) = p(a_{t+1}|a^t, s^{t+1}). \tag{1.33}$$

S využitím (1.28) a (1.33) lze (1.32) upravit takto:

$$p(\theta|a^{t+1}, s^{t+1}) = \frac{p(s_{t+1}|a_t, s_t, \theta)p(\theta|a^t, s^t)}{\int_{\theta\in\mathbf{\Theta}} p(s_{t+1}|a_t, s_t, \theta)p(\theta|a^t, s^t)\mathrm{d}\theta}. \tag{1.34}$$

Pro parametrický model (1.29) zvolíme apriorní pravděpodobnost ve tvaru Dirichletovy hustoty (rovnice (1.11)) dané $V_0(\tilde{s}, a, s) > 0, \quad \tilde{s}, s \in \mathbf{S}, a \in \mathbf{A}$

$$p(\theta) = p(\theta|s^0, a^0) = \mathrm{Dir}(\theta|V_0) = \prod_{s\in\mathbf{S}}\prod_{a\in\mathbf{A}} \left[\prod_{\tilde{s}\in\mathbf{S}} \frac{\theta_{\tilde{s}|s,a}^{V_0(\tilde{s},a,s)-1}}{\Gamma(V_0(\tilde{s}, a, s))}\right] \Gamma\left(\sum_{\tilde{s}\in\mathbf{S}} V_0(\tilde{s}, a, s)\right). \tag{1.35}$$

[8] Poznamenejme, že integrály ve jmenovatelích zlomků jsou pouze normalizace (integrand je pokaždé shodný s čitatelem). V literatuře jsou často odbyty formou konstanty, my jsme se ale rozhodli je pro názornost a úplnost úprav ponechat.

Pak s využitím (1.34) a (1.31) se tento tvar zachovává, tj. $p(\theta|a^t, s^t) = \mathrm{Dir}(\theta|V_t)$, přičemž

$$V_{t+1}(\tilde{s}, a, s) = V_t(\tilde{s}, a, s) + \delta(\tilde{s}, s_{t+1})\,\delta(a, a_t)\,\delta(s, s_t), \quad \forall t \in \mathbf{T}. \tag{1.36}$$

Odpovídající prediktivní pravděpodobnostní funkce (1.27) je

$$p(s_{t+1}|a^t, s^t) = p(s_{t+1}|a_t, s_t, V_t) = \frac{V_t(s_{t+1}, a_t, s_t)}{\sum_{\tilde{s}\in\mathbf{S}} V_t(\tilde{s}, a_t, s_t)}, \quad \forall t \in \mathbf{T}. \tag{1.37}$$

1.6 Zabudování externí znalosti do odhadování

Bayesovské odhadování parametru modelu (Sekce 1.5) odhaduje neznámý parametr parametrického modelu (1.29) s cílem zpřesnit předpověď budoucího stavu na základě dostupných dat, (1.37). V praxi je často k dispozici externí znalost (např. od experta nebo jiného pozorovatele, který pracuje se stejným systémem). Tato znalost může být vyjádřena ve tvaru marginálního modelu, predikujícím budoucí stav, tj. $p_e(s_{t+1})$. Vzniká tedy otázka jak tuto znalost zahrnout do algoritmu odhadování, abychom upravili výslednou aposteriorní hustotu pravděpodobnosti (1.35).

V práci [15] byl navržen postup, jak to udělat optimálním způsobem.

Věta 5 (Slučování znalostí). Nechť $p(s_{t+1}|a_t, s_t, \theta)$ je parametrický model pro parametr $\theta \in \boldsymbol{\Theta}$ a p_e externí znalost o s_{t+1} ve formě modelu $p_e = p(s_{t+1})$. Pak aposteriorní model parametru užívající p_e je

$$p(\theta|a^{t+1}, s^{t+1}, p_e) = \frac{p(\theta|a^t, s^t)\,\exp\left(w\sum_{s_{t+1}\in\mathbf{S}} p_e \ln p(s_{t+1}|a_t, s_t, \theta)\right)}{\int_{\theta\in\boldsymbol{\Theta}} p(\theta|a^t, s^t)\,\exp\left(w\sum_{s_{t+1}\in\mathbf{S}} p_e \ln p(s_{t+1}|a_t, s_t, \theta)\right)\,\mathrm{d}\theta}, \tag{1.38}$$

kde $w \in [0, 1]$ je koeficient vyjadřující důvěryhodnost zdroje informace, poskytujícího p_e, tj. $w = 0$ znamená absolutně nedůvěryhodný zdroj, hodnoty w blízké k 1 znamenají vysokou důvěru ke zdroji informace.

Důkaz. lze najít v [15]. □

Poznámka. Upozorněme, že v (1.38) ve jmenovateli je integrál z čitatele, zaručující normalizaci.

Poznámka. Zdůrazníme, že v (1.38) externí znalost p_e, model systému, hustota popisující parametr i rozhodovací pravidlo se jako funkce mění v čase. Kvůli přehlednějšímu zápisu indexy času k těmto funkcím ale nepíšeme.

Jinými slovy, Věta 5 navrhuje rozšíření Věty 1 tak, že upraví apriorní informaci o parametru $p(\theta|a^t, s^t)$ pomocí externí znalosti p_e o budoucím stavu s_{t+1}.

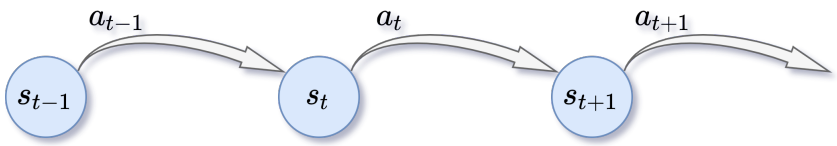
Kapitola 2

Plně pravděpodobnostní učící se agent

V této kapitole zkonstruujeme agenta, který funguje v uzavřené smyčce, viz Obrázek 1.1. Tento agent bude pro generování svých akcí využívat PPN (Sekce 1.4) a bude se učit pomocí bayesovského odhadování (Sekce 1.5).

2.1 Konstrukce agenta

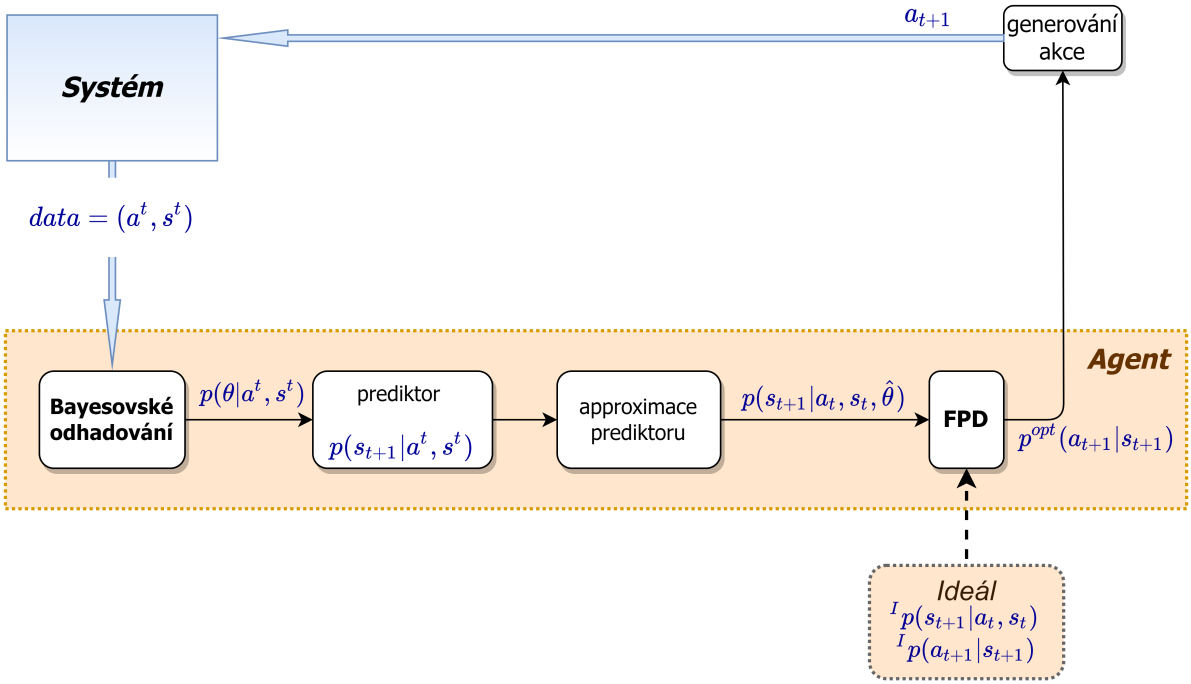
Nejdříve okomentujme časovou návaznost uzavřené smyčky, která je znázorněna na Obrázku 2.1. Agent v čase t napozoruje stav systému s_t, na základě tohoto pozorování zvolí optimální[1] akci a_t. Po realizaci této akce systém přejde do stavu s_{t+1}, který agent zpozoruje v čase $t+1$. Poté se celý proces opakuje znovu. Při tomto přirozeně vzniká tzv. *dopravní zpoždění*, kdy s_{t+1} je pozorován teprve *po* realizaci a_t, tj. v čase t, kdy agent navrhuje akci a_t ještě nezná s_{t+1}. V tomto časování prediktor je ve tvaru $p(s_{t+1}|a_t, s_t)$ (a tak to používáme v celé práci).



Obrázek 2.1: Přijatá časová konvence v uzavřené smyčce

Jedná se však pouze o konvenci a časové indexy lze libovolně posunout, např. můžeme pracovat s prediktorem ve tvaru $p(s_t|a_t, s_{t-1})$. Ovšem pak musíme posunout časové indexy odpovídajícím způsobem i ve všech definicích, větách a předpokladech v Kapitole 1.

[1] Optimální z hlediska naplnění cíle agenta.



Obrázek 2.2: Struktura kooperujícího agenta.

Struktura učícího se agenta, který používá PPN a interaguje se systémem v uzavřené smyčce, je zobrazena na Obrázku 2.2. Hlavními bloky takového agenta jsou: odhadování a návrh optimální strategie. Pomocí odhadování na základě napozorovaných dat agent nalezne prediktor budoucího stavu, který pak použije v návrhu. Aproximace umožňuje získat výpočetně zvládnutelný návrh strategie.

Pro návrh učícího se agenta, používajícího PPN potřebujeme umět vyjádřit cíle agenta pomocí ideálního modelu systému a ideálního rozhodovacího pravidla, které dohromady tvoří model ideálního chování uzavřené smyčky $^I p(b)$, (2.3). Vyjádření cíle pomocí ideálního chování uzavřené smyčky $^I p(b)$ je netriviální problém, nazývaný *kvantitativní popis preferencí* (*angl.* preference elicitation), [13], [1]. Velmi často volba $^I p(b)$ je nejednoznačná a její konstrukce z dostupných znalostí a požadavků agenta je dosti technická. V této práci použijeme způsob konstrukce $^I p(b)$, který je navržen v článku [10] a rozebrán v Sekci 2.2.

2.2 Volba ideálu pro PPN

Předpokládáme, že agent poskytuje popis svého cíle pomocí funkce $L : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \mapsto \mathbb{R}$. Tato funkce vyjadřuje ztrátu, kterou utrpí agent při konkrétní realizaci trojice (s_{t+1}, a_t, s_t)[2]. Pak lze použít *Proposition 6* ze článku [10], která pro náš případ může být přeformulována následujícím způsobem:

[2]Pokud ztrátová funkce není k dispozici, předpokládáme, že ji lze snadno zkonstruovat z preferencí agenta.

Věta 6 (Volba ideálu do PPN). Nechť $p(b)$ je chování uzavřené smyčky a $L(b) = \sum_{t\in\mathbf{T}} L(s_{t+1}, a_t, s_t)$ je celková ztrátová funkce, daná dílčími ztrátovými funkcemi agenta $L(s_{t+1}, a_t, s_t)$.

Pak popis ideálního chování uzavřené smyčky najdeme takto:

$$ {}^{I}p(b) := \frac{p(b)\exp\left(-\frac{L(b)}{\lambda}\right)}{\sum_{b\in\mathbf{B}} p(b)\exp\left(-\frac{L(b)}{\lambda}\right)} \tag{2.1}$$

pro $\lambda > 0$. Z toho vyplývá, že

$$ {}^{I}p(s_{t+1}, a_t|s_t) := \frac{p(s_{t+1}, a_t|s_t)\exp\left(-\frac{L(s_{t+1},a_t,s_t)}{\lambda}\right)}{\sum_{s_{t+1}\in\mathbf{S}}\sum_{a_t\in\mathbf{A}} p(s_{t+1}, a_t|s_t)\exp\left(-\frac{L(s_{t+1},a_t,s_t)}{\lambda}\right)}. \tag{2.2}$$

Získané ${}^{I}p(s_{t+1}, a_t|s_t)$ pak lze za pomocí (1.2) rozložit na ideální model systému a ideální rozhodovací pravidlo:

$$ {}^{I}p(s_{t+1}, a_t|s_t) = {}^{I}p(s_{t+1}|a_t, s_t)\,{}^{I}p(a_t|s_t). \tag{2.3}$$

Dále platí, že pro $\lambda \to 0^+$ optimální strategie, nalezená pomocí PPN s ideálem voleným podle (2.2) konverguje k optimální strategii podle MRP (1.17).

$$ \lim_{\lambda\to 0^+} \arg\min_{\pi\in\mathbf{\Pi}} \mathrm{D}\left(p(b)\|{}^{I}p(b)\right) = \arg\min_{\pi\in\mathbf{\Pi}} \mathbb{E}^{\pi}\left[\sum_{t\in\mathbf{T}} L(s_{t+1}, a_t, s_t)\right]. \tag{2.4}$$

Důkaz. Odvození této věty je v [10]. □

Z praktického hlediska to znamená, že pro konstrukci ${}^{I}p(b)$ při zadané ztrátové funkci L potřebujeme už jen vhodným způsobem zvolit parametr λ. Tento parametr, jak je patrno z (2.2), vyjadřuje míru, do jaké agent trvá na svém cíli. Čím více trvá agent na svých cílech, vyjádřených ztrátou L, tím menší musí být λ a naopak (čím méně trvá agent na svých cílech, vyjádřených ztrátou L, tím větší musí být λ).

Kapitola 3

Spolupráce agentů

Tato kapitola se zabývá úlohou rozhodování ve více–agentním prostředí. Vzhledem k tomu, že implementace a testování (experimenty) budou probíhat na Nashově obchodní hře, úloha spolupráce agentů bude popsána a zformulována pro tuto hru.

3.1 Více–agentní rozhodovací úloha

V předchozí kapitole jsme popsali agenta, který řeší vlastní rozhodovací úlohu vůči systému.Tento systém ale může zahrnovat i další agenty - tzv. sousedy (viz příklad s fotbalistou v Úvodu). Sousedé mají vlastní cíle na prostoru chování $b \in \mathbf{B}$, které mohou být jak podporující vůči našemu agentovi, tak i nepřátelské. Je zajímavé, že i v případě protichůdných cílů (nejsou–li vyloženě antagonistické[1]), spolupráce se agentům vyplatí. Např. představíme–li si dvě auta na úzkém mostě, v případě, že řidiči budou spolupracovat, rychleji projedou. Na druhou stranu i spolupráci podporující soused může ublížit, pokud nebude respektovat agenta a naopak.

V této kapitole se budeme řešit *implicitní kooperaci* agentů. Implicitní znamená, že agenti si vyměňují nikoliv konkrétní akce, ale jinou formu znalostí, např. model. Přitom se předpokládá, že agent je schopen i ochoten spolupracovat, tj.:

- sousedé poskytují agentovi část svých znalostí, které agent může využít při řešení své rozhodovací úlohy,
- agent poskytuje část svých znalostí sousedům, kteří ji případně (mohou a nemusí) využijí ve svých rozhodovacích úlohách).

3.2 Nashova obchodní hra

Nashova obchodní hra (*angl.* Nash demand game), dále jen NOH, byla navržena Johnem F. Nashem [22] jako příklad nekooperativního modelu sekvenčního obchodování mezi dvěma hráči[2]. Dva agenti si rozdělují určité množství peněz. Pokud se nedohodnou na způsobu rozdělení, nikdo nedostává nic. Takto formulovaná úloha může např. ilustrovat smlouvání prodavače a kupujícího o ceně zboží. Automatické smlouvání a jednání je nezbytnou součástí moderních kybernetické–fyzikálně–sociálních systémů [18] a má širokou škálu aplikací: automatické vyjednávání v síti [29], řízení dopravního provozu a dodávání energie [19], systémy více robotů [8] a další.

[1] Příkladem jsou válečné úlohy, kdy cílem je zničit soupeře.

[2] Je také známa pod názvy Dividing the Dollar, Dividing the Cake, Buyer and Seller a jiné.

Pravidla Nashovy obchodní hry (NOH) Hra se skládá z předem stanoveného počtu kol (označíme T). V každém kole $t < T$:

- Hráči si mají rozdělit fixovanou, předem známou částku $q \in \mathbb{N}$.
- Oba hráči současně navrhnou, jaký podíl částky q chtějí. Částka navržená jedním hráčem je pro jeho protihráče neznáma (a naopak).
- Hráči prozrazují navržené částky, a pokud jejich součet je menší nebo roven q, každý dostane to, co chtěl (a zbytek do částky q nedostává nikdo). Pokud součet bude větší než q, nikdo nedostává nic.

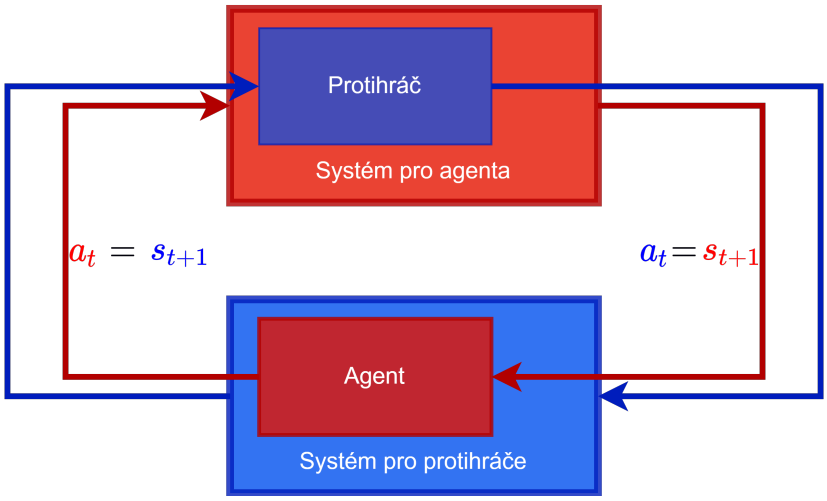
Je tu vidět problém spjatý s neoptimálním rozhodováním agentů, který může vést k neefektivnímu rozdělení prostředků. Konkrétně jde o situace, kdy z důvodu přehnaných požadavků agentů celá částka propadne, anebo když požadavky budou příliš skromné a část q zůstane nevyužita. V praktických úlohách, které jsou modelovány a řešeny pomocí NOH, to může vést k vysokým ztrátám.

Nehledě na zdánlivou jednoduchost NOH, je to dobrý model dynamické alokace zdrojů [6], [26] protože dochází ke kooperaci agentů i bez jejich explicitního smlouvání (hra samotná je k tomu nutí svojí strukturou). Také slouží k výzkumu povahy lidského smlouvání.

3.3 Formulace NOH jako rozhodovací úlohy

Ještě jednou zdůrazníme, že tato práce se zabývá distribuovaným rozhodováním, což znamená, že bude formulovat a řešit úlohu návrhu optimální částky z pozice jednoho z hráčů. V tomto případě jeho *protihráč* bude mít roli systému, viz popis uzavřené smyčky v Sekci 1.2. Je při tom zřejmé, že akce, kterou zvolí protihráč v předchozím kole bude pozorovaným stavem pro agenta v aktuálním kole hry, viz pravidla NOH v Sekci 3.2.

Pokud budeme formulovat rozhodovací úlohu *z pohledu protihráče*, symetrie pravidel NOH implikuje, že v tomto případě systémem protihráče bude agent, což je schematicky zobrazeno na Obrázku 3.1.



Obrázek 3.1: Uzavřená smyčka pro dva agenty: akce agenta je budoucím stavem pro protihráče a naopak. Časové indexy odpovídají přijaté konvenci podle Obrázku 2.1.

Pravidla hry nepředpokládají žádnou koordinaci agentů při návrhu požadovaných částek. Pokud by šlo o centralizované rozhodování (viz str. 13), byl by speciální agent–moderátor, který by určil optimální akci každého hráče. Ovšem představíme–li si příklad s fotbalistou z Úvodu, centralizovaný přístup není možný (rozměrnost úlohy bude příliš vysoká). Právě proto se zaměřujeme na distribuované řešení. Předpokládáme, že hráči pro návrh svých požadovaných částek používají PPN (viz Sekce 1.4). Připomeňme, že agent používající PPN operuje s několika pravděpodobnostními modely:

- model systému, $p(s_{t+1}|s_t, a_t)$ získaný učením,
- model svého cíle (ideál), ${}^I p(s_{t+1}|s_t, a_t)\ {}^I p(a_t|s_t)$ získaný dle Věty 6 pro ztrátu plynoucí z NOH,
- model rozhodovacího pravidla, $p(a_t|s_t)$, optimalizovaný PPN.

Přirozeně, v případě agentů používajících PPN, může implicitní kooperace probíhat přes sdílení jednoho nebo několika těchto objektů. V této práci předpokládáme, že kooperující protihráč sdílí své posledně navržené rozhodovací pravidlo. Tuto informaci bude agent zpracovávat spolu s vlastními pozorováními, čímž zpřesní svůj model protihráče a dokáže navrhnout akci, která lépe zohlední protihráče.

3.4 Kooperace hračů v NOH: detaily algoritmu

Úlohu řešíme z hlediska agenta. Jak vlastně probíhá spolupráce z hlediska algoritmu? V Sekci 1.5 jsme dostali vzorec pro rekurentní opravu statistiky V_t (1.36) pozorovanými daty,

$$V_{t+1}(\tilde{s}, a, s) = V_t(\tilde{s}, a, s) + \delta(\tilde{s}, s_{t+1})\, \delta(a, a_t)\, \delta(s, s_t).$$

Protihráč poskytuje svoje rozhodovací pravidlo (z minulého časového kroku), což je pro agenta *externí prediktor*, popisující stav, tj. $p_e = p_e(s_{t+1})$. Agent tuto informaci[3] použije při odhadování parametru v čase t, viz Věta 5, která pro předpokládaný diskrétní stav má tvar

$$p(\theta|a^t, s^t, p_e) \propto p(\theta|a^t, s^t) \exp\left\{ w \sum_{s_{t+1}\in\mathbf{S}} p_e(s_{t+1}) \ln\left(p(s_{t+1}|s_t, a_t, \theta)\right) \right\}. \tag{3.1}$$

Zdůrazníme, že v (3.1) $p_e(s_{t+1})$ je funkce na množině $\mathbf{S}$. S využitím předpokladu (1.29), (3.1) lze zapsat

$$p(\theta|a^t, s^t, p_e) \propto p(\theta|a^t, s^t) \exp\left\{ w \sum_{s_{t+1}\in\mathbf{S}} p_e(s_{t+1}) \ln\left(\theta_{s_{t+1}|a_t,s_t}\right) \right\}, \tag{3.2}$$

kde parametrizovaná pravděpodobnostní funkce přechodu mezi stavy $\theta_{s_{t+1}|a_t,s_t}$ je dána vztahem (1.31). Po substituci a úpravách dostaneme aposteriorní hustota pravděpodobnosti ve tvaru

$$p(\theta|a^t, s^t, p_e) \propto \prod_{\tilde{s}\in\mathbf{S}} \prod_{a\in\mathbf{A}} \prod_{s\in\mathbf{S}} \theta_{\tilde{s}|a,s}^{V_t(\tilde{s},a,s)-1} \left[\prod_{s_{t+1}\in\mathbf{S}} \theta_{\tilde{s}|a,s}^{w\, p_e(s_{t+1})} \right], \quad \forall a_t \in \mathbf{A}, s_t \in \mathbf{S}. \tag{3.3}$$

Vyraz v hranatých závorkách v (3.3) může byt přepsán následujícím způsobem

$$[\,\bullet\,] = \prod_{\tilde{s}\in\mathbf{S}} \prod_{a\in\mathbf{A}} \prod_{s\in\mathbf{S}} \theta_{\tilde{s}|a,s}^{p_e(\tilde{s}),\delta(a,a_t)\,\delta(s,s_t)}. \tag{3.4}$$

[3]Informací je myšlen předpis funkce p_e, nikoliv její hodnota v bodě.

S označením (3.4), (3.3) dostává tvar

$$p(\theta|a^t, s^t, p_e) \propto \prod_{\tilde{s}\in\mathbf{S}} \prod_{a\in\mathbf{A}} \prod_{s\in\mathbf{S}} \theta_{\tilde{s}|a,s}^{V_t(\tilde{s},a,s)+w\, p_e(\tilde{s})\, \delta(a,a_t)\, \delta(s,s_t)-1}, \tag{3.5}$$

kde hodnota statistiky zohledňující externí informaci je (porovnejme se vzorcem (1.36))

$$V_t(\tilde{s}, a, s, p_e) = V_t(\tilde{s}, a, s) + w\, p_e(\tilde{s})\, \delta(a, a_t)\, \delta(s, s_t). \tag{3.6}$$

Kapitola 4

Experimentální ověření

Tato kapitola popisuje experimenty pro testování chování hráče v Nashově obchodní hře (Sekce 3.2) a ověření přínosu kooperace. Na začátku kapitoly jsou uvedeny parametry simulace. Dále je pak popsána struktura programu a provedené experimenty. V závěru kapitoly je uveden popis získaných výsledků a jejich analýza. Implementace algoritmického řešení byla provedena v prostředí MATLAB©, [20]. Zdrojové kódy jsou přiloženy, viz Příloha B.

4.1 Podstata experimentů

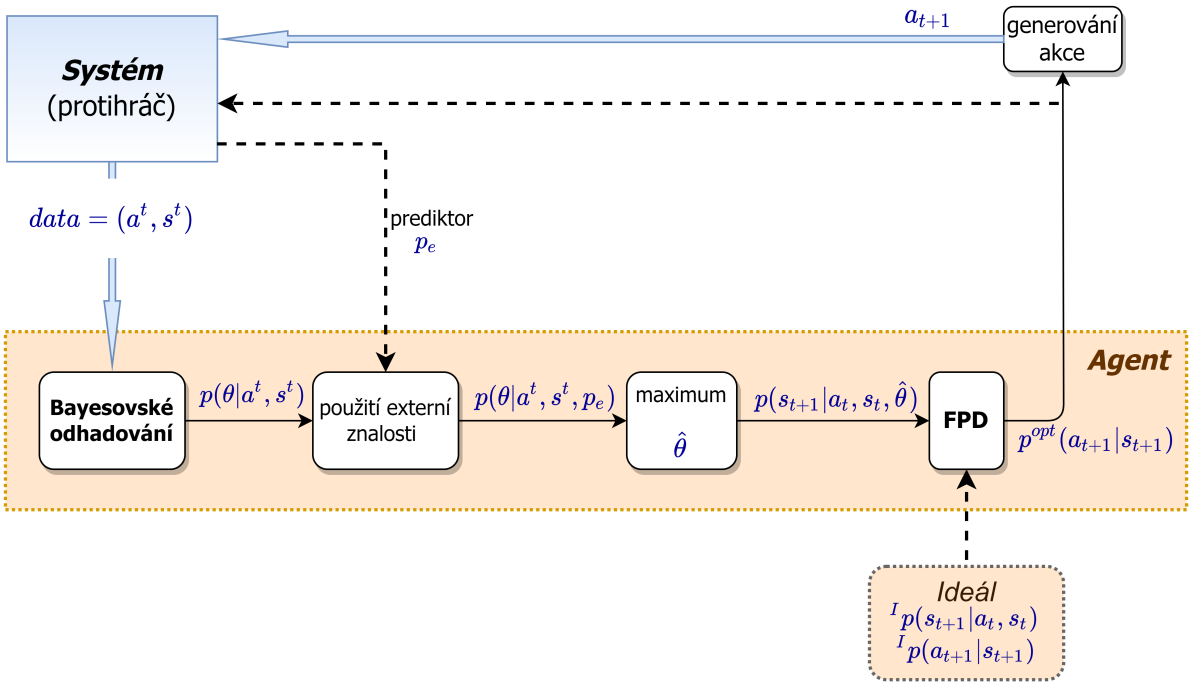
Navržený způsob kooperace (viz Kapitola 3) byl ověřen pomocí simulace NOH, viz (Sekce 3.2). Byli vytvořeni dva agenti strukturálně shodní dle Kapitoly 2: každý používal PPN (viz Sekce 1.4), Bayesovské odhadování parametrů (viz Sekce 1.5) a zpracování informace, poskytnuté protihráčem (viz Sekce 1.6).

Struktura kooperujícího agenta je ilustrována na Obrázku 4.1, kde agent interaguje se systémem (protihráčem) při rozdělování fixované peněžní částky nebo předem daného množství nějaké komodity. Tato částka je stejná v každém kole hry. Množství kol hry je předem známo. Cílem každého hráče je získat maximální celkovou částku.

Jak probíhá interakce hráčů? Agent s použitím svého modelu protihráče a svého ideálu pomocí PPN vytvoří optimální rozhodovací pravidlo $p^{opt}(a_{t+1}|s_{t+1})$ určující pravděpodobnost akcí v kole $t+1$ za podmínky stavu. Akce agenta je generována z tohoto rozhodovacího pravidla.

Jak probíhá kooperace? Protihráč, který funguje stejně jako agent[1] a operuje se stejnými objekty, poskytne agentovi svoje poslední rozhodovací pravidlo (Definice 10). Vzhledem k tomu, ze v NOH akce protihráče tvoří stavy, které agent pozoruje, viz Sekce 3.3, rozhodovací pravidlo protihráče není nic jiného než *externí prediktor budoucího stavu* $p_e = p(s_{t+1})$. Znalost tohoto prediktoru umožní agentovi vylepšit svůj odhad parametru modelu, který byl udělán na základě zpozorovaných dat a apriorna. Obdobně, agent sdílí své poslední rozhodovací pravidlo s protihráčem, který použije tuto znalost pro vylepšení své predikce v příštím kole.

[1] Tj. má stejnou strukturu.



Obrázek 4.1: Struktura kooperujícího agenta. Přerušovanou čárou je označeno sdílení rozhodovacích pravidel mezi agentem a protihráčem.

4.2 Parametry simulace

Hra byla simulována pro následující parametry:

- Celková částka k rozdělení v každém kole $q \in \{6, 10\}$. Pod jednotkou rozdělované částky, celkové částky a zisků si lze představovat peněžní měnu (CZK, USD, EUR,...) nebo množství jakékoliv komodity (např. počet dílků koláče, když koláč je dělen na q rovných dílků).
- Prostor stavů tedy v případě $q = 6$ je $\mathbf{S} = \{1, 2, \ldots, 5\}$ a v případě $q = 10$ je $\mathbf{S} = \{1, 2, \ldots, 9\}$.
- Vzhledem k tomu, že ve formalizaci NOH (Sekce 3.3) je prostor akcí totožný s prostorem stavů, platí: pro $q = 6$ je $\mathbf{A} = \{1, 2, \ldots, 5\}$ a pro $q = 10$ je $\mathbf{A} = \{1, 2, \ldots, 9\}$.
- Počet kol hry (délka simulace) je $T \in \{50, 100, 200\}$.
- Celková částka k rozdělení v jedné hře je určena počtem kol a částkou k rozdělení v jednom kole tj $Q = q\,T$, tedy např. pro $q = 6$ a $T = 50$ je to $Q = 300$.
- Váha důvěry informaci poskytnuté protihráčem v případě kooperace byla $w = 1$ (veškerá informace je zohledněna), v opačném případě $w = 0$.
- Optimální rozhodovací pravidlo (1.21) bylo pokaždé počítáno pro horizont $h = 10$.
- Apriorní pravděpodobnostní funkce pro Bayesovské odhadování parametru je volena jako rovnoměrná, v (1.36) volíme všechny prvky V_0 rovny jedné.

- Cíle jednotlivého agenta (ideální model a ideální rozhodovací pravidlo) byly nastaveny podle Věty 6 s parametry $\lambda \in \{0.4, 0.7\}$, přičemž hráč mající $\lambda = 0.4$ poměrně přísně trvá na svých cílech, zatímco hráč, mající $\lambda = 0.7$ má volnější požadavky.

4.3 Implementace

Jaké funkce jsou používány? Program operuje na několika strukturách a užívá několik funkcí odpovídajících teorii. Vodítkem je Obrázek 4.1. Program pracuje se strukturou `agent=[m,i,r,..]`, kde `agent.m` je model systému (protihráče), `agent.m_i` je ideální model systému, `agent.r` je rozhodovací pravidlo agenta, `agent.r_i` je ideální rozhodovací pravidlo. Vše ve tvaru pravděpodobnostních tabulek. Hlavní funkce v programu jsou:

- funkce `agent_con()` vytváří strukturu `agent`, obsahující `agent=[m, V_t, r, m_i, r_i, horizon, lambda, p_ext...]`, kde `lambda` odpovídá λ v (2.2), `V_t` je statistika v (1.36), `p_ext` je externí prediktor, viz p_e v (3.2).
- funkce `ideal_con()` součást funkce `agent_con()`, vytváří ideální model `agent.m_i` a ideální rozhodovací pravidlo `agent.r_i` (2.3) se ztrátou plynoucí z pravidel NOH.
- funkce `data_con()` vytváří strukturu `data`, obsahující `data=[t, seed, ndat,` s^t, a^t`,...]`, kde `seed` je parametr pro realizaci pseudonáhodných čísel[2] `ndat` je délka simulace (počet kol hry).
- funkce `FPD()` operuje na struktuře agent, na základě ideálů agenta zvolí nové rozhodovací pravidlo `agent.r`, viz Věta 4.
- funkce `generate_data()` operuje na strukturách obou agentů a strukturách obou dat každého z nich, z rozhodovacích pravidel navzorkuje akci a přidá k datům protihráče jako stav.
- funkce `cooperate()` pomocí vzorce (3.6) zohlední v modelu agenta poslední rozhodovací pravidlo protihráče.
- funkce `learn()` realizuje Bayesovské odhadování, vztah (1.36), a operuje na struktuře `agent` tj. na základě nových dat upraví `agent.V_t` a tím i `agent.m`.

Jak probíhá simulace hry? Na začátku se nastaví veškeré konstanty a parametry simulace. Poté proběhne inicializace a v cyklu pro všechny kola hry probíhají následující operace:

krok 1 oba hráči pomocí PPN vygenerují svá optimální rozhodovací pravidla,

krok 2 z rozhodovacích pravidel se navzorkují optimální akce, které protihráče zpozorují jako stavy,

krok 3 hráči si navzájem předají svá rozhodovací pravidla,

krok 4 každý na základě výsledků hry a rozhodovacího pravidla (poskytnutého v kroku 3) poopraví svůj model protihráče pro příští kolo.

Na konci programu proběhne vyhodnocení celé hry a každému hráči se přičte příslušný zisk.

[2] v programu $seed = 10$.

4.4 Experimenty a získané výsledky

Uvádíme 8 experimentů, pokaždé pro všechny případy kooperace (oba hráči kooperují; kooperuje jen jeden hráč; hra bez kooperace).

Experiment 1 Částka v jednom kole $q = 6$, počet kol $T = 50$, horizont $h = 10$. Parametr λ, (2.2), určující míru trvání hráče na svém ideálu[3] $\lambda_1 = \lambda_2 = 0.4$.
Cíl Experimentů 1–6 je zjistit jak se vliv kooperace projevuje na různě dlouhých hrách za podmínky, že oba hráči mají stejně nastavené ideály.

Experiment 2 má stejná nastavení jako Experiment 1, jen délka hry je $T = 100$ kol, čímž celková částka za hru je $Q = 600$.

Experiment 3 má stejná nastavení jako Experiment 1, jen délka hry je $T = 200$ kol, čímž celková částka za hru je $Q = 1200$.

Experiment 4 Částka v jednom kole $q = 6$, počet kol $T = 50$, horizont $h = 10$. Parametr $\lambda_1 = \lambda_2 = 0.7$, tzn. agenti shodně a poměrně volně trvají na svých cílech, vyjádřených ztrátou L (2.2). Přitom větší hodnoty λ dovolují vetší rozptyl akcí.

Experiment 5 má stejná nastavení jako Experiment 4, jen délka hry je $T = 100$ kol, čímž celková částka za hru je $Q = 600$.

Experiment 6 má stejná nastavení jako Experiment 4, jen délka hry je $T = 200$ kol, čímž celková částka za hru je $Q = 1200$.

Experiment 7 Částka v jednom kole je $q = 6$, počet kol $T = 100$, horizont $h = 10$. První hráč poměrně přísně trvá na svých cílech $\lambda_1 = 0.4$, zatímco druhý hráč má poměrně volné požadavky, $\lambda_2 = 0.7$, čímž dovoluje větší rozptyl svých akcí.
Cíl Experimentu 7 je zjistit vliv kooperace, když hráči mají různé míry trvání na svých přáních.

Experiment 8 Částka v jednom kole $q = 10$, počet kol $T = 10000$, horizont $h = 10$, čímž celková částka za hru je $Q = 10000$. Parametr $\lambda_1 = \lambda_2 = 0.7$, tzn. agenti shodně a poměrně volně trvají na svých cílech čímž dovolují větší rozptyl akcí.
Cíl Experimentu 8 je zjistit vliv kooperace ve hře s dlouhým během, větším počtem stavů a akcí a méně přesnou představu o ideálu (což přibližuje např. podmínky reálného obchodování).

Vyhodnocování výsledků Pro vyhodnocení experimentů byly použity následující hodnoty:

- **celkový zisk** hráče za hru (individuální, tj. pro každého hráče svůj). Je to součet všech požadavků hráče (tj. jeho akcí) v kolech, kdy obchodování bylo úspěšné, tj. součet požadavků obou hráčů byl menší nebo roven celkové částce rozdělované v daném kole.
- **nevyužitá částka**. Nevyužitou částkou je myšlen součet zbylých částek přes všechna kola, tj. jsou to prostředky (komodity), které žádný z agentů nedostal. Přičemž jsou tam zahrnuty situace, kdy požadavky agentů byly nekompatibilní (součet akcí obou agentů byl vetší než q a proto celá částka k rozdělení propadla) i situace, kdy rozdělení proběhlo neefektivně (požadavky hráčů byly nízké a část prostředků zůstala nevyužita, např. jeden

[3] Čím více trvá agent na svých cílech, vyjádřených ztrátou L (2.2), tím menší je λ a naopak.

z hráčů požádal o částku 2, druhý o částku 1, takže z částky k rozdělení $q = 6$ zůstalo nevyužito 3).

- **úspěšnost hry** vyjadřuje procento úspěšných kol, tj. všech kol, kdy požadavky hráčů byli kompatibilní[4].

Získané výsledky jsou zaznamenány v tabulkách, kde v levém sloupci jsou pokaždé uvedeny parametry, dále režim kooperace každého agenta, zisk každého agenta, nevyužitá částka a procentuální úspěšnost. Vyjma Experimentu 8, částka k rozdělení v každém kole je $q = 6$. Celková částka za hru závisí na počtu kol T.

Parametry experimentu	**Kooperace** (1 - ano, 0 - ne)		**Zisk**		**Nevyužitá částka**	**Úspěšnost** %
	agent 1	agent 2	agent 1	agent 2		
$T = 50$	1	1	98	110	92	69.3
$\lambda_1 = \lambda_2 = 0.4$	1	0	90	93	117	61.0
$h = 10$	0	1	93	104	103	65.7
$Q = 300$	0	0	78	88	134	55.3
$T = 100$	1	1	235	255	110	87.1
$\lambda_1 = \lambda_2 = 0.4$	1	0	191	208	201	66.5
$h = 10$	0	1	222	237	141	76.5
$Q = 600$	0	0	166	183	251	58.1
$T = 200$	1	1	505	536	159	86.7
$\lambda_1 = \lambda_2 = 0.4$	1	0	441	464	295	75.4
$h = 10$	0	1	472	494	234	80.5
$Q = 1200$	0	0	403	427	370	69.2

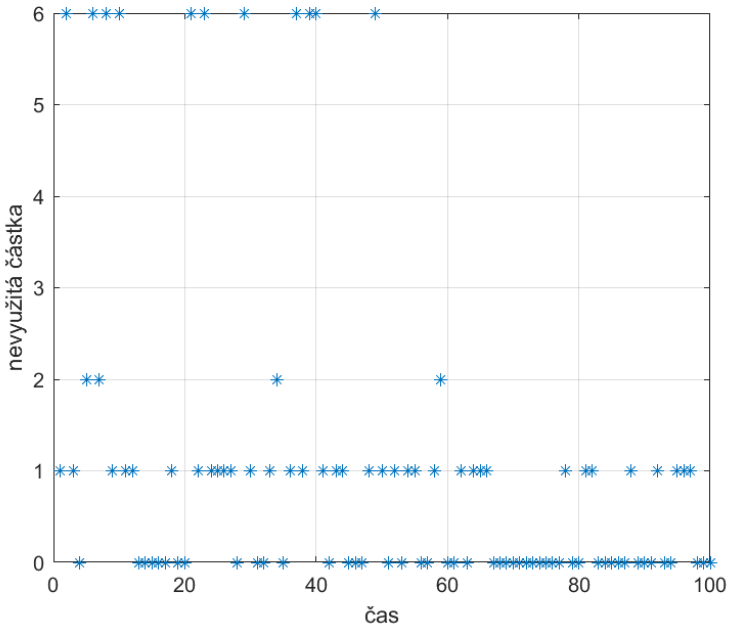
Tabulka 4.1: Výsledky Experimentů 1-3. částka k rozdělení v každém kole je $q = 6$

Získané hodnoty v Tabulce 4.1 ukazují, že kooperace evidentně pomáhá, tj nejvyšší zisk mají oba hráči v situaci, kdy oba kooperují (hodnoty vah jsou $(w_1, w_2) = (1, 1)$). Zároveň v tomto případě je úspěšnost hry nejvyšší a nevyužitá částka je nejmenší. V případě, kdy žádný z hráčů nekooperuje, jsou výsledky nejhorší (zisky hráčů nejmenší).

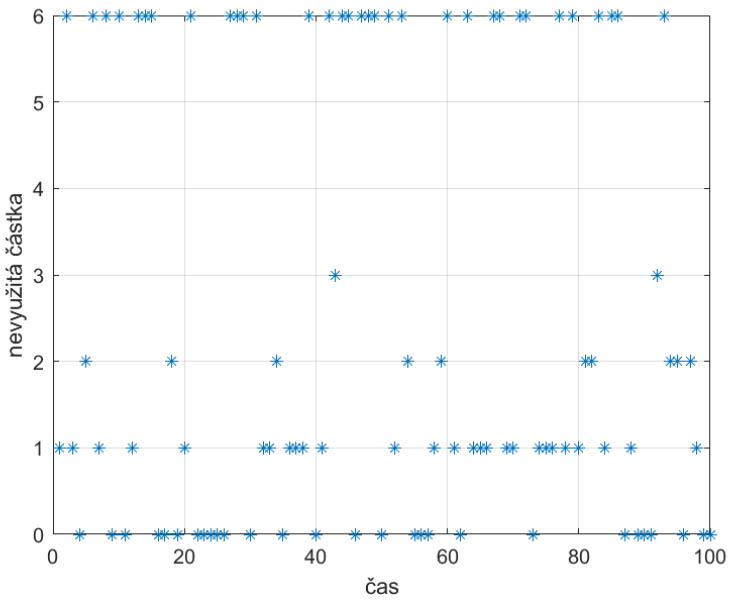
Z porovnání vývoje hodnot nevyužité částky pro Experiment 2 a plně kooperační hru (viz Obrázek 4.2) a pro hru bez kooperace (viz Obrázek 4.3) je vidět, že v případě kooperace velikost nevyužité částky v každém dalším kole postupně klesá, ostatně jako počet neúspěšných kol (kol, kdy rozdělení částky neproběhlo a nevyužitá částka je rovna q). Na Obrázku 4.2 je vidět, ze v druhé polovině hry, cca po $t = 50$, už žádné neúspěšné kolo nebylo, což je dáno tím, že agenti se již naučili model protihráče a došlo k výraznému zmenšení počtu neúspěšných kol a částek, které zůstaly nevyužity. Naopak v případě, kdy agenti nekooperují, nevyužitá částka má vysoké hodnoty po celou dobu hraní, viz Obrázek 4.3. Časový průběh akcí jednotlivých hráčů pro Experiment 2 jsou uvedené na Obrázku 4.4 (bez kooperace) a Obrázku 4.5 (oba hráči kooperují).

V ostatních scénářích (kooperuje pouze jeden z hráčů) hodnoty úspěšnosti hry a velikosti nevyužité částky jsou mezi hodnotami získanými v plně kooperační hře a hře bez kooperace (pro odpovídající experiment).

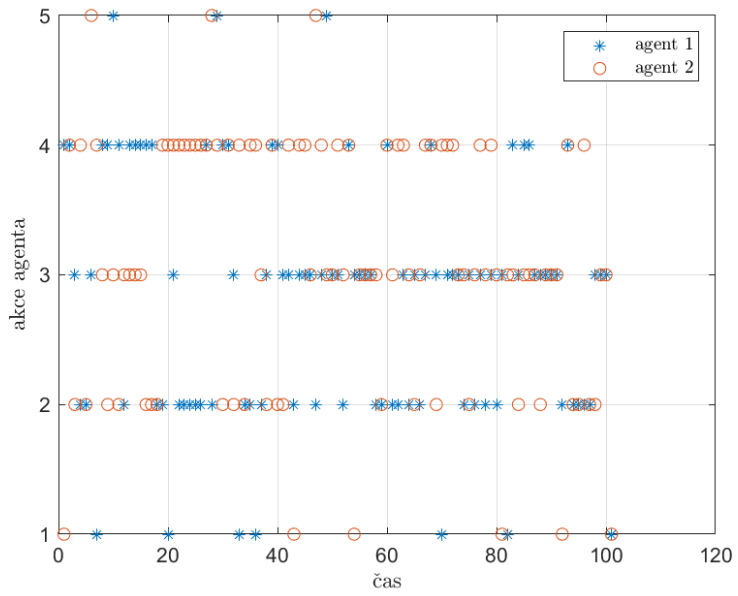
[4]Jinými slovy obchod se uskutečnil, i když nemusel být efektivní, tj. mohla zbýt nenulová částka



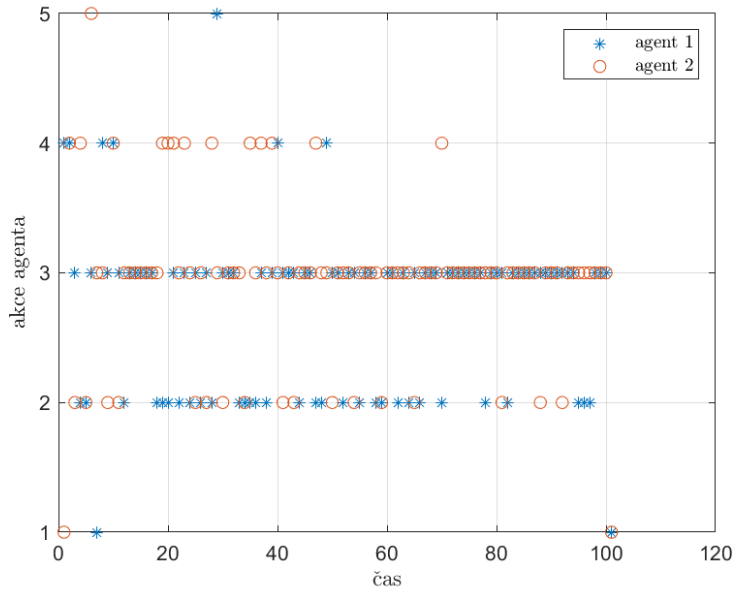
Obrázek 4.2: Experiment 2: Hodnoty nevyužité částky v průběhu hry, kdy oba hráči kooperují.



Obrázek 4.3: Experiment 2: Hodnoty nevyužité částky v průběhu hry bez kooperace.



Obrázek 4.4: Experiment 2: Časový průběh akcí hráčů bez kooperace.



Obrázek 4.5: Experiment 2: Časový průběh akcí hráčů s kooperací.

Individuální zisky ve hře, kdy kooperuje pouze jeden z hráčů, jsou asymetrické, tj. scénáře kooperace $(1, 0)$ a $(0, 1)$ nedávají stejné výsledky: vždy druhý hráč ("agent 2" v tabulkách) má o něco lepší výsledky.

Výsledky všech Experimentů 1–3 mají stejné tendence, viz Tabulka 4.1, liší se jen v počtu kol, který ovlivňuje maximální částku určenou k rozdělení a tím i hodnoty jednotlivých zisků.

Parametry experimentu	**Kooperace** (1 - ano, 0 - ne)		**Zisk**		**Nevyužitá částka**	**Úspěšnost** %
	agent 1	agent 2	agent 1	agent 2		
$T = 50$	1	1	78	80	142	52.7
$\lambda_1 = \lambda_2 = 0.7$	1	0	68	72	160	46.7
$h = 10$	0	1	68	72	160	46.7
$Q = 300$	0	0	68	72	160	46.7
$T = 100$	1	1	165	171	264	56.0
$\lambda_1 = \lambda_2 = 0.7$	1	0	149	157	294	51.0
$h = 10$	0	1	155	162	283	52.8
$Q = 600$	0	0	151	158	291	51.5
$T = 200$	1	1	332	337	531	55.7
$\lambda_1 = \lambda_2 = 0.7$	1	0	307	308	585	51.2
$h = 10$	0	1	316	316	568	52.7
$Q = 1200$	0	0	304	307	589	50.9

Tabulka 4.2: Výsledky Experimentů 4–6, částka k rozdělení v každém kole je $q = 6$

Parametry Experimentů 4–6, viz Tabulka 4.2, se liší od Experimentů 1–3 pouze parametrem λ, který ale významně ovlivňuje celý průběh hry, jelikož umožňuje hráčům dělat více neurčité akce. Je to vidět z hodnot zisků jednotlivých hráčů. Navíc při malém počtu kol $T = 50$ (Experiment 4), pokud jeden z hráčů nekooperuje, výsledky jsou stejné[5]. Je to dáno tím, že 50 kol nepostačí k tomu, aby se hráči naučili modely svých protihráčů, kteří měli dost volná rozhodovací pravidla, dle kterých generovali své akce a které poskytli jako externí znalost v rámci kooperace.

Zvětšování délky hry mění tuto situaci k lepšímu. V Experimentech 5 a 6 už je vidět rozdíl a opět se projevuje tendence, kterou jsme pozorovali na Experimentech 1–3: kooperace pomáhá - individuální zisk hráčů a úspěšnost jsou maximální v kooperačním scénáři. Minima nevyužité částky je také dosaženo v případě kooperace obou hráčů.

Na rozdílnosti výsledků Experimentů 1–3 a Experimentů 4–6 jsme zpozorovali zásadní vliv vhodné konstrukce ideálu, odrážejícího preference hráče (v daném příkladě je to volba λ). V obou sadách experimentů (sada 1–3 a sada 4–6) hráči měli stejné hodnoty λ, což znamenalo, že trvali na svých cílech se stejným důrazem. Pro srovnání byl proveden Experiment 7, ve kterém hráči měli různé hodnoty λ. Výsledky (Tabulka 4.3) ukazují, že v tomto případě kooperace má podstatně menší vliv. Scénář bez kooperace má nejhorší výsledky, ovšem výsledky scénářů, kdy oba agenti kooperují nebo kooperuje jenom jeden, jsou srovnatelné.

[5] ve všech experimentech byla záměrně použita stejná hodnota *seed*, aby vliv náhody byl vyloučen.

Parametry experimentu	**Kooperace** (1 - ano, 0 - ne)		**Zisk**		**Nevyužitá částka**	**Úspěšnost** %
	agent 1	agent 2	agent 1	agent 2		
$T = 100$	1	1	177	190	233	61.2
$\lambda_1 = 0.4$	1	0	169	179	252	58.0
$\lambda_2 = 0.7$	0	1	178	190	232	61.3
$Q = 300$	0	0	167	172	261	56.5

Tabulka 4.3: Výsledky experimentu 7: částka k rozdělení v každém kole je $q = 6$

Jak bylo vidět v Experimentech 4–6, vliv nejistého popisu preferencí (vysoké hodnoty λ) lze částečně vykompenzovat délkou hraní - pak hráči mají dost času, aby se naučili modely svých protihráčů. Příkladem toho je Experiment 8, ve kterém oba hráči mají vysokou nejistotu v ideálech, přitom ale počet kol hry je stanoven na vysokou hodnotu $T = 10000$. Částka k rozdělení v každém kole je zvolena $q = 10$. Výsledky (Tabulka 4.4) ukazují, že i v tomto případě kooperace má sice malý, ale pozitivní vliv. Celková úspěšnost je ale výrazně vyšší než u Experimentu 6, který má podobné parametry (vyjma q a T).

Parametry experimentu	**Kooperace** (1 - ano, 0 - ne)		**Zisk**		**Nevyužitá částka**	**Úspěšnost** %
	agent 1	agent 2	agent 1	agent 2		
$T = 10000$	1	1	41620	41695	16685	83.3
$\lambda_1 = \lambda_2 = 0.7$	1	0	41240	40729	18031	82.0
$h = 10$	0	1	40976	40558	18466	81.5
$Q = 100000$	0	0	40843	39926	19231	80.8

Tabulka 4.4: Výsledky experimentu 8: částka k rozdělení v každém kole je $q = 10$

Závěr

Tato práce se zabývá problémem implicitní spolupráce agentů ve více–účastnickém prostředí. Pro řešení této úlohy jsem nastudoval i) pravděpodobnostní popis uzavřené smyčky, ii) nezbytnou teorii bayesovského odhadování parametrů Markovského modelu včetně zabudování externího prediktoru, iii) princip plně pravděpodobnostního návrhu optimální rozhodovací strategie. Navržená implicitní kooperace spočívá v tom, že agenti si vzájemně sdílí svá rozhodovací pravidla z minulého kola hry. Poskytnutá informace je pak agentem vnímána jako externí prediktor a pomocí již existující metody použita pro vylepšení odhadu modelu systému. Tento model je následně použit v plně pravděpodobnostním návrhu optimální strategie v dalším kole hry.

Práce používá různá, již existující teoretická, algoritmická a částečně softwarová řešení[6] s cílem vytvořit funkčního kooperujícího agenta. Součástí práce byl návrh i implementace adaptivního, plně pravděpodobnostního agenta a tvorba programu, zajišťující výše zmíněnou implicitní kooperaci.

Výsledné řešení bylo testováno na Nashově obchodní hře. Bylo provedeno několik sérií experimentů, část z nich je prezentována v práci. Při vytváření plně pravděpodobnostního agenta byl navržen ideální model[7], který odpovídá maximalizaci individuálního zisku hráče. Získané výsledky ukazují, že implicitní kooperace pomáhá zvýšit nejen individuální zisky, ale i více využít potenciál hry, tj. zvýšit úspěšnost kol a snížit nevyužitou částku. Zásadní vliv na výsledky experimentů má parametr λ, který ve volbě ideálu odráží míru důležitosti cílů pro rozhodujícího agenta. Provedené experimenty ukázali, že čím přesnost naplnění cíle je méně důležitá pro agenta, tím horší výsledky agent získává. Tento výsledek je odůvodněn tím, že větší λ znamená vyšší rozptyl pravděpodobnostní funkce popisující ideál, což přirozeně způsobuje vyšší rozprostřenost výsledných akcí a horší přesnost z hlediska sdíleného rozhodovacího pravidla. Na druhou stranu právě tento jev může být extrémně důležitý pro systémy s velkou neurčitostí (neznalostí), jelikož zajišťuje vysokou míru průzkumu (*exploration*). Průzkum je nezbytně nutný v případě větších prostorů stavů a malém množství dat. V případě NOH tato neurčitost je poměrně nízká, proto použití vysokých hodnot λ je spíš na škodu.

Do budoucna by bylo třeba:

- vylepšit algoritmus hráče a to tak, že by si nejdřív vytvořil představu o rozhodovacím pravidle, a vzápětí by tuto představu sdílel. Následně by proběhl stávající algoritmus, popsaný v práci. Toto vylepšení by umožnilo zbavit se kroku zpoždění, spočívající v tom, že agent ve stávajícím kole hry zpracovává rozhodovací pravidlo protihráče z minulého kola (ne stávající), což může být poněkud pozdě.

- vylepšit volbu ideálu zabudováním informace o částce, která zbyla z minulého kola s cílem její minimalizace. Tím by došlo k dodatečné zpětné vazbě.

[6] původně vytvořená v oddělení Adaptivních systémů ÚTIA

[7] který vyjadřuje cíle agenta

- provést detailní analýzu podmínek, při kterých nastane rovnovážný stav (hodnoty stavů a akcí se ustálí).
- navrhnout implicitní kooperaci pro obecnější příklad než Nashova obchodní hra, tj. kdy stav agenta není dán pouze akcí protihráče.
- navrhnout rozšířené experimenty a případně otestovat navrženého agenta v reálném obchodování nebo hře s lidskými hráči.
- rozšířit teorii pro případ spojitých množin stavů a akcí.

Příloha A

Doplněk ke Kapitole 1

A.1 Alfa-funkce aneb o názvosloví Beta a Gamma funkcí

Alfa funkce v tomto kontextu neexistuje, jedná se o výhradu názvosloví. Ač beta funkci a gama funkci poprvé definoval Leonhard Euler (1707–1783), nazýval je "Eulerovský integrál prvního a druhého druhu[1]".

Symbol "Γ" pro gamma funkci zavedl (bez bližšího vysvětlování) Adrien Marie Legendre (1752–1833). Existuje domněnka, že Γ zvolil kvůli vizuální podobnosti s prvním písmenem svého příjmení (osově souměrné) a fonetické podobnosti ("Le Gendre" a "gamma").

Jacques P. M. Binet (1786–1856) používal pro dnešní beta funkci označení "β", avšak zanedlouho byl inspirován Legendreovým označením gama funkce. Zdůvodňuje obvyklostí pro Evropana písmenka B, propůjčil označení této funkce první písmeno svého příjmení – "B" [21].

A.2 Kullback–Leiblerova divergence

Kullback–Leiblerova divergence je míra, která udává, jak se liší dvě distribuční funkce (například $F(x)$ a $G(x)$). Její definice pro spojitý i diskrétní případ viz Definice 4.

Definice 14 (Gaussova křivka)**.** *Gaussova křivka* [2] je funkce $f : \mathbb{R} \to \mathbb{R}$, definovaná takto:

$$f(x) = a \cdot e^{-\frac{(x-b)^2}{2c^2}} \tag{A.1}$$

kde $a, b, c \in \mathbb{R}$. Její graf má klasický "zvonový" tvar. Parametr a představuje výšku peaku, b je poloha osy symetrie, a parametr c má vliv na šířku "zvonu".

Tuto funkci používáme nejčastěji k popisu normálního rozdělení pravděpodobnosti se střední hodnotou $\mu = b$ a rozptylem (variancí) $\sigma^2 = c^2$. S použitím takové volby konstant máme:

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Výpočet KLD mezi dvěma Gaussovými křivkami Mějme dvě Gaussovy křivky [3].

$$F(x) = a_1 \cdot e^{-\frac{(x-b_1)^2}{2c_1^2}} \quad \text{a} \quad G(x) = a_2 \cdot e^{-\frac{(x-b_2)^2}{2c_2^2}}$$

[1] angl. Eulerian integral of the first kind and second kind

[2] také Gaussova funkce

[3] Odteď k jednoznačnému určení takové funkce budeme používat pouze tři parametry a, b, c. Jejich význam viz Definice 14

Poměr

$$\frac{F(x)}{G(x)} = \frac{a_1 \cdot e^{-\frac{(x-b_1)^2}{2c_1^2}}}{a_2 \cdot e^{-\frac{(x-b_2)^2}{2c_2^2}}} = \frac{a_1}{a_2} \cdot \exp\left(-\frac{(x-b_1)^2\, c_2^2}{2c_1^2 c_2^2} + \frac{(x-b_2)^2\, c_1^2}{2c_1^2 c_2^2}\right)$$

Dosazení do vzorce (1.4)

$$\mathrm{D}(F \parallel G) = \int_{x\in\mathbf{X}} F(x) \cdot \ln\left(\frac{a_1}{a_2} \cdot \exp\left(-\frac{(x-b_1)^2\, c_2^2}{2c_1^2 c_2^2} + \frac{(x-b_2)^2\, c_1^2}{2c_1^2 c_2^2}\right)\right) \mathrm{d}x$$

Nyní ke zjednodušení použijeme dvě základní vlastnosti logaritmu $\forall a, b: \ \ln(a \cdot b) = \ln a \cdot \ln b$ a $\forall w : \ln e^w = w$:

$$\mathrm{D}(F \parallel G) = \int_{x\in\mathbf{X}} F(x) \cdot \ln\left(\frac{a_1}{a_2}\right) \mathrm{d}x + \int_{x\in\mathbf{X}} F(x) w \mathrm{d}x$$

kde $w = -\frac{(x-b_1)^2\, c_2^2}{2c_1^2 c_2^2} + \frac{(x-b_2)^2\, c_1^2}{2c_1^2 c_2^2}$
S využitím linearity integrálu:

$$\mathrm{D}(F \parallel G) = \ln\left(\frac{a_1}{a_2}\right) \cdot \int_{x\in\mathbf{X}} F(x)\, \mathrm{d}x \ + \ \int_{x\in\mathbf{X}} F(x) \cdot w\, \mathrm{d}x$$

Z vlastností Gaussovy funkce platí, $\int_{x\in\mathbf{X}} F(x)\, \mathrm{d}x = 1$.

$$\mathrm{D}(F \parallel G) = \ln\left(\frac{a_1}{a_2}\right) + a_1 \cdot \int_{x\in\mathbf{X}} \exp\left(-\frac{(x-b_1)^2}{2c_1^2}\right) \cdot \left(K_1 x^2 + K_2 x + K_3\right)\, \mathrm{d}x,$$

kde K_1, K_2 a K_3 jsou konstanty závislé pouze na $\{b_1, b_2, c_1, c_2\}$ a vznikly v důsledku úprav zlomků. Očividně levá strana je představena součtem tří Gaussových integrálů, které díky linearitě integrálu lze počítat zvlášť.

Příloha B

Doplněk ke Kapitole 4

B.1 Spouštěcí skript

```
%% MAIN_test is a main programme that runs experiments

% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20220105
%
%% maintenance
%clc
%close all
clear variables

%% pre-defining constants
num_of_s = 6;           % number of states {6,8,10}
num_of_a = num_of_s;    % number of actions is equal to number of states (see ...
    the NDG rules)
ndat = 100;             % the number of simulation steps
rnd_seed = 10;          % seed for the random generator
rng(rnd_seed);
horizon_1 = 10;         % horison for optimisation for player 1
horizon_2 = horizon_1;  % horison for optimisation for player 2
w_1 = 1;                % indicates whether player 1 cooperates (=1) or not (=0).
%Genereally this weight reflects player's trust in
%the information provided by the co-player.
w_2 = 0;

%% constructors
agent_1 = agent_con (1, num_of_s, num_of_a, w_1, horizon_1, 0.4);       % ...
    constructor of the 1st player
agent_2 = agent_con (2, num_of_s, num_of_a, w_2, horizon_2, 0.4);       % ...
    constructor of the 2nd player

data_1 = data_con (1, rnd_seed, ndat);                                % data of ...
    the 1st player
data_2 = data_con (2, rnd_seed, ndat);                                % data of ...
    the 2nd player

%% time loop over simulation steps
%figure(1);
```

```
33 t_vect=[];                                                          ...
       %initialising time vector
34 for t = 1: ndat                                                     % loop ...
       over time
35 data_1.t = t;                                                    % storing time ...
       in both data
36 data_2.t = t;
37 %% creating DM rule (FPD)
38 agent_1 = FPD(agent_1);
39 agent_2 = FPD(agent_2);
40 %% sampling optimal actions and sharing future states
41 [data_1,data_2] = generate_data (agent_1,agent_2,data_1,data_2);
42 %% cooperation - step when players share their previous DM rules (2x ...
       Janek-Karny's formula is used)
43 s_t_1 = data_1.s_to_t(t);
44 agent_1 = cooperate(agent_1, data_1, agent_2.r(:,s_t_1));
45 s_t_2 = data_2.s_to_t(t);
46 agent_2 = cooperate(agent_2, data_2, agent_1.r(:,s_t_2));
47
48 %% learning step of both players, when both players update their models
49 %
50 agent_1 = learn(agent_1, data_1);
51 agent_2 = learn(agent_2, data_2);
52 t_vect = [t_vect, t];
53 end
54 %% Game evaluation, where 'unused cake' means unclaimed part of the entire ...
       amount; 'succes rate' is a percent of games with non-zero profit
55 [pr1, pr2, unused_cakes, success_rate] = profit (data_1, data_2, num_of_a, ...
       num_of_s, ndat)
```

B.2 Konstruktor agenta

```
function agent = agent_con (id, num_of_s, num_of_a, w, horizon, lambda)
%% Constructor of the agent
%
% OUTPUT:
% agents structure contains:
%  - id ... agent identifier
%  - model ... model of the environment, p(s_{t+1} | a_t, s_t)
%  - V_t ... non normalised model
%  - r ... decision-making rule, p(a_t | s_t)
%  - m_i ... ideal model
%  - r_i ... ideal decision-making rule
%  - w ...  the agent's trust [0,1] in external predictor p_ext provided by ...
    the co-player
%  - horizont ... time horizont
%  - lambda ... weight used for constructing ideals from the loss (low values ...
    of lambda correpsonds to more strict preferencing of the particular values)
%  - p_ext  ... state predictor offered by an external source
%
% INPUT:
%  - id ... agent identifier
%  - ns ... number of state  values
%  - na ... number of action values
%  - w  ... weight (of trust in p_ext)
%  - horizont ... time horizont
%  - lambda ... weight used for constructing ideals from
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20220102
%

%% Default values
V_t    = ones(num_of_s,num_of_a,num_of_s);        % non-normalised prior model ...
    (ones correspond uniform distribution).
m      = V_t/num_of_s;                            % model of the environment, ...
    p(s_{t+1} | a_t, s_t) (normalized though states)
r      = ones(num_of_a,num_of_s)/num_of_a;        % decision-making rule, ...
    p(a_t | s_t)
p_ext  = ones(num_of_s,1)/num_of_s;               % external state predictor, ...
    i.e. information provided by the cooperating co-player (column vector)
[m_i,r_i]= ideal_con(num_of_s,num_of_a,lambda);    % function for ...
    constructing ideal from the known loss

%% struct
agent    = struct('id', id, 'm', m, 'V_t', V_t, 'r', r,'r_i', r_i,...
'm_i', m_i, 'w', w, 'horizon', horizon, 'lambda', lambda, 'p_ext', p_ext);
end
```

B.3 Konstruktor ideálu

```
function [m_i, r_i] = ideal_con (num_of_s, num_of_a, lambda)
%% Constructor of the agent ideal DM rule and ideal model
%
% OUTPUT:
%  - m_i ... ideal model
%  - r_i ... ideal DM rule
%
% INPUT:
%  - num_of_s ... number of state  values
%  - num_of_a ... number of action values
%  - lambda ... weight used for constructing ideal from the known loss
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20211230
%

%% Default values
m_i   = ones(num_of_s, num_of_a, num_of_s) / num_of_s ; % ideal model of the ...
    environment, p^i(s_{t+1} | a_t, s_t, \theta)
r_i   = ones(num_of_a, num_of_s) / num_of_a;            % ideal DM rule, ...
    p^i(a_t | s_t)

%% non-normalised joint ideal model
c_i   = ones(num_of_s, num_of_a, num_of_s);             % p^i(s_{t+1}, a_t | s_t)
for s_tp1 = 1:num_of_s
for a_t = 1:num_of_a
if (a_t + s_tp1 ≤ num_of_s)
c_i(s_tp1, a_t, :) = exp(a_t / lambda); % if demands of the players are ...
    compartible.
% lambda influence exploration: high lambda - high exploration
% and vice versa
else
c_i(s_tp1, a_t, :) = 1; % if players' demands are not comparable and the ...
    entire amount is lost and both players get zero profit
end
end
end


%% decomposition : c_i <> p^i(s_{t+1}, a_t | s_t) =|chain rule|= p^i(s_{t+1}| ...
    a_t, s_t)*p^i(a_t | s_t) % decompostion the ideal onto the ideal system ...
    model and ideal DM rule
for a_t = 1:num_of_a
for s_t = 1:num_of_s
r_i(a_t, s_t) = sum(c_i(:, a_t, s_t)); % step 1 : marginal of c_i over all ...
    future states s_tp1
for s_tp1 = 1:num_of_s
m_i(s_tp1, a_t, s_t) = c_i(s_tp1, a_t, s_t)/ r_i(a_t, s_t); %
end
end
end

%% normalisation of r_i and m_i
for s_t = 1:num_of_s
```

```
48  r_i(:, s_t) = r_i(:, s_t) / sum(r_i(:, s_t)); % normalisation via a_t
49  for a_t = 1: num_of_a
50  m_i(:, a_t, s_t) = m_i(:, a_t, s_t) / sum(m_i(:, a_t, s_t)); % normalisation ...
        via state s_tp1
51  end
52  end
53  end
```

B.4 Konstruktor dat

```
1   function data = data_con (id, rnd_seed, ndat)
2   %% Data constructor
3   %
4   % data = data_con (id, rnd_seed, ndat)
5   %
6   % INPUT:
7   %  - id       ... agent identifier
8   %  - rnd_seed ... seed of the random generator
9   %  - ndat     ... simulation length
10  %
11  % OUTPUT:
12  % data structure contains:
13  %  - id ... agent identifier
14  %  - t  ... current time
15  %  - rnd_seed ... seed of the random generator
16  %  - s_to_t   ... array for storing states
17  %  - a_to_t   ... array for storing actions
18  %  - ndat     ... simulation length
19  %
20  % Part of BSc project of AG (FJFI, CVUT)
21  % Last updated by AG, 20220102
22  %
23
24  %% defaults
25  t = 1;                          % current time
26  s_to_t = ones(1,ndat+1);        % array for storing states sequence
27  a_to_t = ones(1,ndat+1);        % array for storing action sequence
28  %% struct
29  data = struct('id', id, 't', t, 'rnd_seed', rnd_seed, 's_to_t', s_to_t, ...
        'a_to_t', a_to_t, 'ndat', ndat);
30  end
```

B.5 PPN (FPD)

```
function agent = FPD (agent)
%% This function sovles FPD and provides an optimal DM rule
%
% INPUT and OUTPUT
% agent  = structure describing the agent
%
%  - id ... agent identifier
%  - model ... parameterised model of the environment, p(s_{t+1} | a_t, s_t, ...
    \theta)
%  - V_t ... non-normalised model
%  - r ... DM rule, p(a_t | s_t)
%  - m_i ... ideal model describing the agent preferences
%  - r_i ... ideal DM rule describing the agent preferences
%  - w ... trust [0,1] in external predictor p_ext provided by the co-player
%  - horizont ... time horizont
%  - lambda ... weight used for constructing ideals from the loss (low values ...
    of lambda correpsonds to more strict preferencing of the particular values)
%  - p_ext  ... state predictor offered by the co-player
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20211230
%
%% Importing matrices from agent struct for shorter code
m   = agent.m;                              % - the agent model of the co-player
m_i = agent.m_i;                            % - the agent's ideal model
r_i = agent.r_i;                            % - the agent's ideal DM rule
[num_of_a, num_of_s] = size(agent.r);       % - getting number of states and ...
    number of actions
h = ones(1, num_of_s);                      % - auxiliary function h; preallocation

%% dyn.prog. (backward recursion)
for t = agent.horizon:-1:1                  % backward induction over time
d = zeros(num_of_a,num_of_s);             % - function d: preallocation
h_given = h;                             % - store h from the previous iteration
h       = 0*h;                           % - array for new h
for s_t = 1:num_of_s                     % - possible states  at time t
for a_t = 1:num_of_a                  % - possible actions at time t
for s_tp1 = 1:num_of_s           % - possible states  at time t+1
% Sum d(a_t, s_t) over all s_{t+1} \in S
d(a_t, s_t) = d(a_t, s_t) + m(s_tp1, a_t, s_t) * log ( (m(s_tp1, a_t, s_t)) / ...
    (m_i(s_tp1, a_t, s_t) * h_given(s_tp1)) );
end
% sum over all a_t \in A
h(s_t) = h(s_t) + r_i(a_t, s_t) * exp( - d(a_t, s_t)); %
end
end
end

%%  evaluate only the last optimal DM rule and store it into agent structure
for s_t = 1:num_of_s
agent.r(:,s_t) = r_i(:,s_t) .* exp(-d(:, s_t))/h(s_t);  %
end

end
```

B.6 Generování akcí

```
function agent = FPD (agent)
%% This function sovles FPD and provides an optimal DM rule
%
% INPUT and OUTPUT
% agent  = structure describing the agent
%
%  - id ... agent identifier
%  - model ... parameterised model of the environment, p(s_{t+1} | a_t, s_t, ...
    \theta)
%  - V_t ... non-normalised model
%  - r ... DM rule, p(a_t | s_t)
%  - m_i ... ideal model describing the agent preferences
%  - r_i ... ideal DM rule describing the agent preferences
%  - w ... trust [0,1] in external predictor p_ext provided by the co-player
%  - horizont ... time horizont
%  - lambda ... weight used for constructing ideals from the loss (low values ...
    of lambda correpsonds to more strict preferencing of the particular values)
%  - p_ext  ... state predictor offered by the co-player
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20211230
%
%% Importing matrices from agent struct for shorter code
m   = agent.m;                              % - the agent model of the co-player
m_i = agent.m_i;                            % - the agent's ideal model
r_i = agent.r_i;                            % - the agent's ideal DM rule
[num_of_a, num_of_s] = size(agent.r);       % - getting number of states and ...
    number of actions
h = ones(1, num_of_s);                      % - auxiliary function h; preallocation

%% dyn.prog. (backward recursion)
for t = agent.horizon:-1:1                  % backward induction over time
d = zeros(num_of_a,num_of_s);          % - function d: preallocation
h_given = h;                           % - store h from the previous iteration
h       = 0*h;                         % - array for new h
for s_t = 1:num_of_s                   % - possible states  at time t
for a_t = 1:num_of_a              % - possible actions at time t
for s_tp1 = 1:num_of_s       % - possible states  at time t+1
% Sum d(a_t, s_t) over all s_{t+1} \in S
d(a_t, s_t) = d(a_t, s_t) + m(s_tp1, a_t, s_t) * log ( (m(s_tp1, a_t, s_t)) / ...
    (m_i(s_tp1, a_t, s_t) * h_given(s_tp1)) );
end
% sum over all a_t \in A
h(s_t) = h(s_t) + r_i(a_t, s_t) * exp( - d(a_t, s_t)); %
end
end
end

%%  evaluate only the last optimal DM rule and store it into agent structure
for s_t = 1:num_of_s
agent.r(:,s_t) = r_i(:,s_t) .* exp(-d(:, s_t))/h(s_t);  %
end

end
```

B.7 Kooperace

```
function agent = cooperate(agent, data, p_ext)
%% Cooperation of the agents using merging given by Kracik-Karny (2005) formula
%
% agent = cooperate(agent, data, p_ext)
%
% INPUT: agent ... structure describing the agent
%        data  ... structure describing the data
%        p_ext ... external state predictor, i.e. information provided by the ...
    cooperating co-player
%
% OUTPUT:
%       agent ... structure  with updated statistics agent.V_t and
%       model of the agent agent.m
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20220102
%

%% 'importing' useful constants
[num_of_a, num_of_s] = size(agent.r);

%% choosing the current triple of states and action
t = data.t;
s_t = data.s_to_t(t);
a_t = data.a_to_t(t);

%% updating non-normalised model (V_t)
agent.V_t(:, a_t, s_t) = agent.V_t(:, a_t, s_t) + agent.w * p_ext;   % update ...
    of the statistics

%% updating normalised model (including normalisation)
for a_t = 1:num_of_a
for s_t = 1:num_of_s
agent.m(:, a_t, s_t) = agent.V_t(:, a_t, s_t) / (sum(agent.V_t(:,a_t,s_t))); ...
        % update agent's model and ensure it stays normalized
end
end

end
```

B.8 Učení

```
function agent = learn(agent, data)
%% The function learns the agent's system model, which describes the ...
    co-player from the player's viewpoint.
%
% agent = learn(agent, data)
%
% INPUT:
%   agent ... structure describing the player
%   data  ... structure containng data
%
% OUTPUT:
%   agent ... struct., describing agent
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20220102
%

%% 'importing' information about number of actions and states
[num_of_a, num_of_s] = size(agent.r);

%% choosing current tripple of states and action
t = data.t;
s_t = data.s_to_t(t);
a_t = data.a_to_t(t);
s_tp1 = data.s_to_t(t+1);

%% updating non-normalised model (V_t)
agent.V_t(s_tp1, a_t, s_t) = agent.V_t(s_tp1, a_t, s_t) + 1;   % update the ...
    statistics matrix

%% updating normalised model (including normalisation)
for a_t = 1:num_of_a
for s_t = 1:num_of_s
agent.m(:, a_t, s_t) = agent.V_t(:, a_t, s_t) / (sum(agent.V_t(:,a_t,s_t)));
% update agent's model and ensure it stays normalized
end
end

end
```

B.9 Výpočet zisků a vyhodnocení

```
function [pr1, pr2, unused_cakes, success_rate] = profit (data_1, data_2, ...
    num_of_a, num_of_s, ndat)
%% PROFIT calculates profit of each player, total unclaimed amount and ...
    percentage of games that ended with non-zero profit
%
% INPUT:
% data1 ... structure containng data of player_1
% data2 ... structure containng data of player_2
%
% num_of_s ... number of states
% num_of_a ... number of actions
%
% OUTPUT:
% pr1 .... profit of player 1
% pr2 ... profit of player 2
%
% unused_cake   ... unclaimed part of the entire amount
% success_rate  ... percent of succesful games, i.e. games with non-zero profit
%
% Part of BSc project of AG (FJFI, CVUT)
% Last updated by AG, 20220103
%

%% validation error
if num_of_a ≠ num_of_s
error('func PROFIT says: Number of actions and number of states are ...
    different! Aborted.')
end

%% setting output variables
K = num_of_s;       % the entire amount to be divided
pr1 = 0;            % initialising the 1st player profit
pr2 = 0;            % initialising the 2nd player profit
unused_cakes = 0;   % initialising
success_rate = 0;   % initialising

%% calculation of profit
for t = 1:ndat
a1_t = data_1.a_to_t(t);  % action of player 1 in time t
a2_t = data_2.a_to_t(t);  % action of player 2 in time t

if a1_t + a2_t ≤ K    % if both demands (actions) are comparable, then each ...
    player gets what he demanded
pr1 = pr1 + a1_t;  % profit of player 1
pr2 = pr2 + a2_t;  % profit of player 2
unused_cakes = unused_cakes + (K - a1_t - a2_t); % unclaimed amount of the ...
    current game
else
unused_cakes = unused_cakes + K; %
end
success_rate = 1 - unused_cakes / (K * ndat); % ratio of succesfully ended ...
    games to the number of all games
end
```

Literatura

[1] L. Chen and P. Pu. Survey of preference elicitation methods. Technical Report IC/2004/67, HCI Group Ecole Politechnique Federale de Lausanne, Switzerland, 2004.

[2] S. Cohen and N. Agmon. Convexified graph neural networks for distributed control in robotic swarms. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2307–2313. International Joint Conferences on Artificial Intelligence Organization, 2021.

[3] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, 2nd ed edition, 2006.

[4] F. Cucker and S. Smale. Emergent behavior in flocks. *IEEE Transactions on Automatic Control*, 52(5):852–862, 2007.

[5] S. Hawking and L. Mlodinow. *The Grand Design*. Dokořán, 2011.

[6] J. Homolová, T.V. Guy, and A. Gaj. Indirect dynamic negotiation in the Nash Demand Game. submitted.

[7] V. Hurtado-Marín, J. Agudelo-Giraldo, S. Robledo, and E. Restrepo-Parra. Analysis of dynamic networks based on the Ising model for the case of study of co-authorship of scientific articles. *Scientific Reports*, 11, 2021.

[8] T. Ito, H. Hattori, M. Zhang, and T. Matsuo. *Rational, Robust, and Secure Negotiations in Multi-Agent Systems*. Studies in Computational Intelligence. Springer Berlin Heidelberg, 2008.

[9] M. Kárný. Towards fully probabilistic control design. *Automatica*, 32(12):1719–1722, 1996.

[10] M. Kárný. Fully probabilistic design unifies and supports dynamic decision making under uncertainty. *Inf. Sci.*, pages 104–118, 2020.

[11] M. Kárný and T.V. Guy. Preference elicitation within framework of fully probabilistic design of decision strategies. In *IFAC Int. Workshop on Adaptive and Learning Control Systems*, volume 52, pages 239–244, 2019.

[12] M. Kárný and T. Kroupa. Axiomatisation of fully probabilistic design. *Inf. Sci.*, 186(1):105–113, 2012.

[13] M. Kárný and T. Siváková. Agent's feedback in preference elicitation. In *Prep. the 4th Int. Conf. on Data Sci & Computational Intelligence, DSCI*. IEEE CS, London, 2021.

[14] D. Knebel, C. Sha-ked, N. Agmon, G. Ariel, and A. Ayali. Collective motion as a distinct behavioral state of the individual. *iScience*, 24(4):102299, 2021.

[15] J. Kracík and M. Kárný. Merging of data knowledge in Bayesian estimation. In J. Filipe, J. Andrade-Cetto, and J.-L. Ferrier, editors, *ICINCO 2005, Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics, Barcelona, Spain, September 14-17, 2005, 4 Volumes / CD*, pages 229–232. INSTICC Press, 2005.

[16] M. Krbálek and D. Helbing. Determination of interaction potentials in freeway traffic from steady-state statistics. *Physica A-statistical Mechanics and Its Applications*, 333:370–378, 2004.

[17] V. Kůs. *Míra a pravděpodobnost*. Skripta FJFI, 2019.

[18] L. Li, K. Robert Lai, and S. Zhu. Data-driven behavior-based negotiation model for cyber-physical-social systems. *IEEE Access*, 7:83319–83331, 2019.

[19] B. López, B. Innocenti, and D. Busquets. A multiagent system for coordinating ambulances for emergency medical services. *IEEE Intelligent Systems*, 23(5):50–57, 2008.

[20] MATLAB. *version R2021a*. The MathWorks Inc., Natick, Massachusetts, 2021.

[21] J. Miller. Earliest uses of function symbols. `http://jeff560.tripod.com/functions.html`. (Accessed on 11/28/2021).

[22] J. F. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[23] V. Peterka. *Bayesian System Identification In. P. Eykhoff, Trends and Progress in System Identification*. Pergamon Press, Oxford, 1981.

[24] M. Puterman. *Markov decision processes*. John Wiley & Sons, 1994.

[25] M. Rached, Z. Bahroun, and J.-P. Campagne. Decentralised decision-making with information sharing vs. centralised decision-making in supply chains. *International Journal of Production Research*, 54(24):7274–7295, 2016.

[26] A. Rubinstein. Perfect equilibrium in a bargaining model. *Econometrica*, 50(1):97–109, 1982.

[27] J. Šindelář, I. Vajda, and M. Kárný. Stochastic control optimal in the Kullback sense. *Kybernetika*, 44(1):53–60, 2008.

[28] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[29] Y.-Y. Yang and X.-M. Xie. Research on the effectiveness of network negotiation based on evolutionary game model. *IEEE Access*, 8:194623–194630, 2020.

[30] E. Zehavi and N. Agmon. Hybrid path planning for uav traffic management. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6427–6433, 2021.

Rejstřík