Bakalářská práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Reinforcement learning pro řízení dynamických systémů

Loi Do

Vedoucí: doc. Ing. Petr Hušek, Ph.D.
Obor: Systémy a řízení
Studijní program: Robotika a Kybernetika
Květen 2017

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Do Loi**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Reinforcement learning pro řízení dynamických systémů**

Pokyny pro vypracování:

Cílem práce je aplikovat metody reinforcement learning pro řízení dynamických systémů.

1. Charakterizujte obecné principy metod založených na reinforcement learning a popište možnosti, jak lze jednotlivé kroky implementovat.
2. Implementujte vybrané metody vhodné pro řízení dynamických systémů.
3. Otestujte dané metody na konkrétních příkladech řízení dynamických systémů.

Seznam odborné literatury:

[1] R. S. Sutton, A. G: Barto: Reinforcement Learning: An Introduction, MIT Press, 1998
[2] F. L. Lewis, D. Vrabie: Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control, IEEE Circuits and Systems Magazine, 3/2009, pp. 32-50
[3] L. Pack Kaelbling, M. L. Littman, A. W. Moore: Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4 (1996) 237-285

Vedoucí: doc. Ing. Petr Hušek, Ph.D.

Platnost zadání: do konce letního semestru 2017/2018

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 2. 2017

Poděkování

Chtěl bych poděkovat své rodině a blízkým za podporu během studia. Také bych chtěl poděkovat panu doc. Ing. Petru Hušk ovi za cenné připomínky při psaní této bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 22. května 2017

. . . . . .

Abstrakt

Tato bakalářská práce se zabývá použitím metod strojového učení označované jako Posilované učení (Reinforcement learning) pro řízení dynamických systémů. První část této práce je věnovaná úvodu do problematiky Posilovaného učení. Je zde popsán teoretický rámec používaný ve strojovém učení. Definovány jsou základní pojmy a hlavně je představen popis úlohy pomocí Markovského rozhodovacího procesu. Další část práce je pak věnovaná přehledu metod, které se používají pro řešení úloh popsaných jako Markovské rozhodovací procesy. Ukázány jsou základní metody založené na dynamickém programování a posléze metody založené na konceptu Temporální diference. Vybraná metoda je nakonec použita pro řízení simulačního modelu dynamického sytému.

Klíčová slova: posilované učení, markovův rozhodovací proces, řízení dynamického sytému, aproximace funkce

Vedoucí: doc. Ing. Petr Hušek, Ph.D.

Abstract

This bachelor thesis deals with use of machine learning methods known as Reinforcement learning for control of dynamic system. First part of this thesis is dedicated to introduction to problematics of Reinforcement learning. Theoretical framework used in machine learning is described here. Basic terms are defined and mainly the description of task is described using Markov decision process. Next part of the work is dedicated to survey of methods used for solution of task described as Markov decision process. Basic methods based on Dynamic programming and Temporal difference are shown. Selected method is eventually used for control simulated model of dynamic system.

Keywords: reinforcement learning, markov decision process, control of dynamic systems, function approximation

Obsah

1 Úvod **1**
1.1 Motivace 1
1.2 PU v přírodě 2
1.3 PU v teorii strojového učení 2

2 Charakterizace obecné úlohy pro PU **5**
2.1 Markovův rozhodovací proces . . . 5
2.2 Cíl v MRP 6
2.2.1 Ohodnocovací funkce 6
2.3 Struktura PU 8
2.3.1 Interakce agenta s prostředím . 8
2.3.2 Charakterizace úlohy řízení dynamického systému pomocí MRP 8

3 Přehled metod **11**
3.1 Dynamické programování 12
3.1.1 Bellmanova rovnice 12
3.1.2 Iterace strategie 13
3.1.3 Iterace ohodnocení 14
3.1.4 Příklad využití metod DP . . . 14
3.2 Metody temporální diference . . . 17
3.2.1 Princip temporální diference . 17
3.2.2 Problém explorace a exploatace 18
3.2.3 SARSA - Online Iterace strategie 19
3.2.4 Q-učení - Iterace ohodnocení 20

4 Aproximace v PU **23**
4.1 Parametrická aproximace funkcí 23
4.2 Gradientní sestup 24
4.3 Lineární metody 24
4.3.1 Agregace stavů 25
4.3.2 Radiální bázové funkce 25
4.4 SARSA a Q-učení s aproximací . 27

5 Experimentální ověření metod PU **31**
5.1 Model DC motoru 31
5.2 Implementace Q-učení s aproximací 32
5.2.1 Volba parametrů 32
5.2.2 Definice ceny 32
5.3 Výsledky simulací 33
5.3.1 Regulace rychlosti 33
5.3.2 Potlačení poruchy 35

6 Závěr **39**

A Literatura **41**

B Terminologie a zkratky **43**
B.1 Terminologie 43
B.2 Zkratky 44

Obrázky

2.1 Struktura interakce agenta s prostředím při PU pro diskrétní kroky $k = 0, 1, 2, ...$ 9

4.1 Pokrytí 1-D stavového prostoru Agregovanými stavy. 26
4.2 Pokrytí 1-D stavového prostoru Gaussovskými RBF. 27
4.3 Konvergence Q-funkce při použití funkční aproximace. 28

5.1 Regulace rychlosti ω_k pro Q-učení s agregací stavů. 34
5.2 Akční zásahy pro regulaci rychlosti při druhých bězích. 34
5.3 Q-hodnoty pro jednotlivé páry stav-akce pro Q-učení s agregací stavů. 35
5.4 Hladové akce pro jednotlivé stavy pro Q-učení s agregací stavů pro regulaci rychlosti. 35
5.5 Regulace rychlosti ω_k pro Q-učení s Gaussovskými RBF. 36
5.6 Potlačení poruchy pro Q-učení s agregací stavů. 36
5.7 Potlačení poruchy pro Q-učení s Gaussovskými RBF. 37

Tabulky

3.1 Klasifikace metod pro řešení MRP 12
3.2 Q-hodnoty pro jednotlivé páry stav-akce při využití Iterace strategie. 16
3.3 Q-hodnoty pro jednotlivé páry stav-akce při využití offline Iterace ohodnocení. 17

5.1 Hodnoty parametrů pro DC motor a jejich popis. 32

Kapitola 1

Úvod

Tato bakalářská práce se zaobírá řešením návrhu řízení dynamického systému, které je založené na metodách Posilovaného učení (PU). Ty vycházejí z teorie strojového učení inspirovaného přírodou. Jedná se o metodu učení, která je založena na modifikaci chování na základě interakce s prostředím. V dlouhém časovém horizontu tento způsob učení často vede i k optimálnímu a adaptivnímu chování [3].

Vzhledem k tomu, že použití PU pro návrh řízení dynamického systému není příliš rozšířené, uvedeme si před samotnou aplikací těchto metod v kapitole 2 potřebný teoretický rámec. Půjde zejména o popis Markovova rozhodovacího procesu (MRP) a popis úlohy řízení pomocí tohoto rámce.

V kapitole 3 si představíme metody, které lze použít k řešení MRP. Půjde o metody založené na dynamickém programování a metody založené na principu Temporální diference. Dále si ukážeme v kapitole 4 použití aproximace funkcí v PU. Ta je nutná zejména pro řízení systémů se spojitým stavovým prostorem, kde je potřeba zobecnit již získané hodnoty při učení i na další, podobné případy.

Nakonec v kapitole 5 otestujeme vybranou metodu již k samotnému zpětnovazebními řízení modelu dynamického systému. K tomu využijeme software MATLAB.

1.1 Motivace

Na rozdíl od návrhu zpětnovazebního regulátoru klasickými metodami (stavová zpětná vazba, frekvenční metody, apod.), dokáží metody PU dosáhnout optimálního i adaptivního řízení bez jakékoliv (případně pouze s částečnou) znalosti systému [3]. Klasické metody jsou vždy založené na znalosti dynamiky systému, např. ve formě diferenciálních/diferenčních rovnic, případně stavového popisu. Ty je ale mnohdy u složitějších nelineárních systémů složité nebo dokonce nemožné nalézt.

1.2 PU v přírodě

Metody PU jsou inspirovány chováním živých organismů, v jejich metodách učení a následného chování. Organismy se učí skrze interakci se svým prostředím, využívají nabyté zkušenosti k vylepšení svého chování. Jedná se o změnu chování zejména k dosáhnutí konkrétního cíle (přežití, naučení motorických pohybů, minimalizaci úsilí, apod.).

Organismy učící se pomocí PU tedy vždy znají konkrétní cíl (*goal-directed learning*), neznají však cestu (sekvenci správných akcí), jak se k němu dostat. K naučení sekvence akcí a konvergenci k alespoň suboptimálnímu řešení je poté potřeba, aby organismus měl možnost vyhodnotit zpětnou vazbu z prostředí na jím provedenou akci. Tou může být v jednodušším případě pouze pozorování vzdálenosti do cíle. U organismů s rozvinutější nervovou soustavou může být tato zpětná vazba například i ve formě hormonů, které mohou kladně či záporně stimulovat mozek. Organismus poté preferuje v dlouhodobém měřítku ty akce, které mu zajišťují kratší vzdálenosti do cíle anebo kladné stimulace mozku.

Příklad 1.1. Jedním z příkladů PU může být způsob, kterým se mláďata některých zvířat, např. koní, učí chodit. Ta, pozorováním prostředí a zkoušením různých pohybů (akcí), jsou schopna se v krátkém čase od narození naučit pohybovat téměř bez problémů. Konkrétním cílem pro mládě může být dojít si k matce pro potravu. Zkoušením různých pohybů (akcí) dostává zpětnou vazbu od prostředí, v tomto případě např. vzdálenost od matky nebo bolest způsobená pádem při nesprávném pohybu.

1.3 PU v teorii strojového učení

V teorii strojového učení často rozlišujeme metody učení na dvě hlavní skupiny: Učení s učitelem (*supervised learning*) a Učení bez učitele (*unsupervised learning*). Metody PU tvoří samostatnou skupinu metod, není možné je zařadit ani mezi jednu z nich.

Rozdíl mezi PU a Učením s učitelem je hlavně v absenci trénovací množiny. Každý prvek trénovací množiny je tvořen párem vstupního objektu a požadovaného výstupu. Rozhodování v případě Učení s učitelem je tedy založené na znalosti této trénovací množiny a hledání podobností s touto množinou. Rozhodování v případě PU je založené pouze na vlastních zkušenostech získaných při interakci s prostředím, k dispozici nejsou příklady správného chování.

Mohlo by se zdát, že PU patří tedy mezi metody Učení bez učitele, ale opět zde můžeme nalézt rozdíly. Učení bez učitele se snaží nalézt skryté struktury (podobné vlastnosti) u neoznačených dat, zatímco PU se snaží o minimalizaci celkové ceny.

Poznámka 1.2. Je vhodné si uvědomit, že zatímco v teorii strojového učení se používá v rámci optimalizace maximalizace konkrétních hodnot, v teorii řízení se používá

konvence jejich minimalizace. Od toho se také odráží označení daných hodnot. V teorii strojového učení se můžeme setkat s pojmem užitku (*utility/reward*) a naopak v teorii řízení s pojmem ceny (*cost*). Pro konzistenci s teorií řízení budeme používat konvenci minimalizace ceny.

Kapitola 2

Charakterizace obecné úlohy pro PU

V této sekci se zaměříme na obecný popis úloh, které lze řešit pomocí PU. Zároveň ale již zavedeme notaci odpovídající té, která je používaná v teorii řízení, aby později nedocházelo k nejasnostem. Ukážeme si základní strukturu úloh a schéma interakce mezi jednotlivými aktéry. Pro exaktní popis úloh využijeme rámec Markovova rozhodovacího procesu. To nám poté v sekci 3 poskytne teoretický základ k popisu již konkrétních metod PU.

2.1 Markovův rozhodovací proces

MRP poskytují matematický rámec pro popis rozhodovacích procesů (deterministických i stochastických), které pracují v diskrétních krocích $k = 0, 1, 2, \ldots$. V každém kroku k je proces v konkrétním stavu x_k. Tento stav je možné změnit pomocí akce u_k, přičemž na základě přechodových pravděpodobností se proces v kroku $k + 1$ dostane do stavu x_{k+1} a je generována cena r_{k+1}. Přesněji lze MRP definovat [2] jako uspořádanou čtveřici $(X, U, p(\cdot), r(\cdot))$, kde:

- X je množina všech stavů,
- U je množina všech akcí,
- $p(x'|x, u) \in \langle 0; 1\rangle$ jsou pravděpodobnosti přechodu pro všechny dvojice stavů (x, x'), $x, x' \in X$ při provedení akce $u \in U$,
- $r(x, u, x') \in \mathbb{R}$ je cena získaná při přechodu mezi stavy x a x' provedením akce $u \in U$.

Dále musí platit, že přechod mezi stavy je tvz. Markovský, definovaný jako:

$$p(x', r|x, u) = Pr\{x_{k+1} = x', r_{k+1} = r|x_k = x, u_k = u\}\,, \tag{2.1}$$

tedy, že pravděpodobnosti přechodů ze stavu x do stavu x' (a zároveň také získaná cena r) závisí pouze na současném stavu x a vybrané akci u. Veškeré informace

potřebné k přechodu do dalšího stavu musí být obsaženy ve stavu x. Úlohy, které budeme chtít nadále řešit musí splňovat vztah (2.1). Říkáme pak, že jsou Markovské.

Pro téměř deterministické procesy lze cenu $r(x, u, x')$ psát i bez explicitní závislosti na stavu x', jelikož pravděpodobnost přechodu $p(x'|x, u) \approx 1$. Cena r je pak určena jednoznačně pouze dvojicí (x, u), jelikož (x, u) vede téměř vždy na stejný stav x'. Zápisem[1] $r(x_k, u_k)$ je pak možné označit získanou cenu, jestliže ve stavu x_k zvolíme akci u_k. Je nutné zdůraznit, že cenu $r(x_k, u_k)$ lze vypočítat (předpovídat) již v kroku k pouze při známých přechodových pravděpodobnostech (modelu systému). Jestliže tyto hodnoty nejsou známé, cenu lze získat až v kroku $k + 1$, tedy po použití akce u_k a pozorováním stavu x_{k+1}. V tomto případě budeme preferovat zápis r_{k+1}.

2.2 Cíl v MRP

V úlohách MRP, je hlavním úkolem v kroku k nalezení optimální strategie (*policy*) volby akce $h^*(x_k)$ tak, aby byla minimalizována dlouhodobá cena při nekonečném průchodu MRP nebo při dosažení požadovaného stavu x_c (cíl). Obecně lze strategii volby akcí $h(x_k)$ přesně definovat jako:

$$u_k = h(x_k) \ . \tag{2.2}$$

Jedná se o namapování z pozorovaného stavu $x_k \in X$ prostředí na konkrétní akci $u_k \in U$. Dlouhodobou cenu by se nabízelo definovat jako sumu všech cen získaných během průchodu MRP, nicméně pro nekonečný proces by tato suma konvergovala k nekonečnu a její minimalizace by tak nebyla možná. Je nutné tedy zavést dlouhodobou očekávanou cenu R_k [3] při počátku v časovém kroku k jako:

$$R_k = \sum_{i=k}^{T} \gamma^{i-k} r_{i+1} \ , \tag{2.3}$$

kde $\gamma \in (0; 1\rangle$ se nazývá diskontní faktor. Pro nekonečné procesy $T = \infty$ je pak zvolena hodnota parametru $\gamma < 1$, čímž je zajištěna konvergence nekonečné sumy.

2.2.1 Ohodnocovací funkce

Pomocí dlouhodobé ceny (2.3) lze nyní kvantifikovat vzdálenost použité strategie $h(x_k)$ od optimální strategie $h^*(x_k)$. Definujeme k tomuto účelu ohodnocovací funkce $V^h(x_k)$ (*Value function*) a $Q^h(x_k, u_k)$ (Q-*function*) [3]. Zkráceně budeme využívat označení V-funkce, V-hodnota a obdobně Q-funkce, Q-hodnota.

[1] V literatuře je možné se setkat i se zápisem $\mathbb{E}(r(x, u))$ (expected) zdůrazňující nejistotu v získanou cenu, jestliže proces není deterministický.

V-funkce

Funkce $V^h(x_k)$ přiřadí stavu $x_k \in X$ očekávanou dlouhodobou cenu při následování strategie $h(x_k)$ ze stavu x_k:

$$V^h(x_k) = \sum_{i=k}^{T} \gamma^{i-k} r_{k+i+1} = \sum_{i=k}^{T} \gamma^{i-k} r(x_i, h(x_i)) \ , \tag{2.4}$$

kde na faktor γ lze pohlížet v tomto případě také jako na nejistotu získané ceny v budoucích krocích. Jelikož je našim úkolem minimalizace dlouhodobé ceny, platí, že pro neoptimální strategii $h(x_k)$ je:

$$V^h(x_k) > V^{h^*}(x_k) \ , \tag{2.5}$$

tedy, že pro optimální strategii $h^*(x_k)$ bude hodnota $V^{h^*}(x_k)$ stavu x_k vždy nejmenší. Dále platí, že čím je hodnota $V^h(x_k)$ stavu menší, tím je pro danou strategii lepší se v daném stavu nacházet (očekávaná získaná cena v tomto stavu je nízká).

Pomocí funkce $V^h(x_k)$ lze pak definovat řešení MRP jako nalezení optimální strategie $h^*(x_k)$ pro kterou bude ohodnocení $V^*(x_k)$ všech stavů minimální:

$$V^*(x_k) = \min_{h(.)} V^h(x_k) \ . \tag{2.6}$$

Optimální strategie je pak dána jako:

$$h^*(x_k) = \arg\min_{h(.)} V^h(x_k) \ . \tag{2.7}$$

Nalezení řešení této úlohy pro složitější procesy je obecně velice obtížné. Je dobré si uvědomit, že je nutné minimalizovat celkovou dlouhodobou cenu, ne pouze samotnou cenu $r(x_k, h(x_k))$ v jednom kroku.

Q-funkce

Q-funkce neohodnocuje pouze stavy x_k, ale ohodnocuje použití akce u_k ve stavu x_k při následování strategie $h(x_k)$, tedy ohodnocuje páry stavu a příslušné akce (zkráceně budeme používat označení: pár stav-akce). Formálně lze zapsat tuto funkci jako:

$$Q^h(x_k, u_k) = \sum_{i=k}^{T} \gamma^{i-k} r(x_i, u_i) \ . \tag{2.8}$$

Pro tuto funkci dostáváme obdobné vztahy, jako pro V-funkci.

$$Q^*(x_k, u_k) = \min_{h(.)} Q^h(x_k, h(x_k)) \ . \tag{2.9}$$

$$h^*(x_k) = \arg\min_{h(.)} Q^h(x_k, h(x_k)) \ . \tag{2.10}$$

Jestliže následujeme strategii $h(x_k)$, lze napsat vztah mezi funkcemi $V^h(.)$ a $Q^h(.)$ jako

$$Q^h(x, h(x_k)) = V^h(x_k) \ . \tag{2.11}$$

Na Q-funkci lze tedy nahlížet jako na rozšíření V-funkce. Použitím V-funkce pro každý stav $x \in X$ uchováváme pouze jednu hodnotu, zatímco při použití Q-funkce pro daný stav x_k navíc uchováváme hodnoty pro všechny možné akce. To se bude hodit zejména v metodách, které nejsou založené na přímém následování konkrétní strategie, ale na dynamické změně strategie volby akcí.

2.3 Struktura PU

Struktura PU vždy obsahuje dvě jednotky, které spolu neustále interagují. Jedná se o agenta (*agent*) a prostředí (*environment*). Agentem je označovaná jednotka, která se učí a rozhoduje o volbě akcí. Jeho úkolem je často dosáhnutí konkrétního cíle (*goal*), případně pouze minimalizace dlouhodobé ceny. Prostředím je poté označené vše ostatní, tedy soubor všech fyzikálních veličin a parametrů okolí, které mají vliv na agenta (jeho rozhodování, dosažení cíle, apod.).

2.3.1 Interakce agenta s prostředím

Aby bylo možné použít pro popis interakce mezi agentem a prostředím rámec MRP, je nutné uvažovat tuto interakci pouze v diskrétních časových krocích $k = 0, 1, 2, \ldots$ [3]. Pro úlohy PU jsou vyvinuty i metody založené na interakci ve spojitém čase [2], nicméně stěžejní výzkum PU je zaměřen na oblast metod diskrétního času, tedy ve využití MRP.

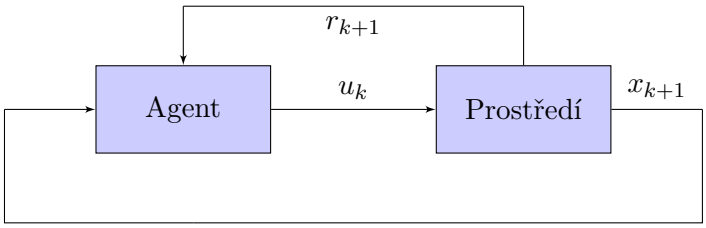
Prostředí lze uvažovat jako MRP. Jeho aktuální konfiguraci v čase k lze reprezentovat jako stav $x_k \in X$. Množina X odpovídá všem možným konfiguracím prostředí. Na agenta lze pohlížet jako na implementaci strategie $h(x)$. V každém kroku k má agent k dispozici stav x_k, na základě čehož volí podle strategie $h(x)$ akci $u_k \in U(x_k)$, kde $U(x_k) \subseteq U$ je množina všech akcí dostupných ve stavu x_k.

V čase $k+1$ se poté prostředí nachází ve stavu x_{k+1} a generuje cenu r_{k+1}, skalární hodnotu, která je vztažena k přechodu mezi stavy x_k a x_{k+1} využitím akce u_k. Struktura celé interakce je vyobrazena na obr. 2.1.

2.3.2 Charakterizace úlohy řízení dynamického systému pomocí MRP

V teorii řízení lze pohlížet na prostředí jako na řízený systém a na agenta jako na regulátor. Většinu dynamických systémů lze popsat ve formě diskrétního stavového popisu:

$$x_{k+1} = F(x_k, u_k) \ , \tag{2.12}$$



Obrázek 2.1: Struktura interakce agenta s prostředím při PU pro diskrétní kroky $k = 0, 1, 2, ...$

kde $x_k \in R^n$, $u_k \in R^m$ a $F(x, u)$ je obecně nelineární funkce. Zároveň rovnice (2.12) splňuje podmínku (2.1), je tedy Markovský. Pro lineární systémy lze vztah (2.12) napsat za pomocí matic A, B následovně:

$$x_{k+1} = Ax_k + Bu_k \ . \tag{2.13}$$

Cenu $r(x_k, u_k)$ lze v úloze řízení definovat různými způsoby. Je vhodné využít hodnoty, které chceme v čase minimalizovat. Jednou z možností definice ceny $r(x_k, u_k)$ může být např kvadratická energetická funkce [2]:

$$r(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k \ , \tag{2.14}$$

kde Q a R jsou pozitivně definitní matice. Dále by bylo možné uvažovat pouze např. vzdálenost od reference, spotřebu energie, apod..

Regulátor, který implementuje řízení podle (2.2) se nazývá zpětnovazební. Jedním z příkladů implementace strategie může být stavová zpětná vazba:

$$u_k = -Kx_k \ . \tag{2.15}$$

Cílem regulátoru je pak nejčastěji stabilizace systému na referenční hodnotě x_{ref}. Je nutné si uvědomit, že při uvedeném popisu se referenční hodnota teoreticky nesmí měnit. Znamenalo by to pak změnu získaných cen r_{k+1} pro daný pár stav-akce, což by porušovalo vztah (2.1). Proces by tak nebyl Markovský. V reálném případě se agent během učení dokáže adaptovat na mírné změny reference, které lze modelovat jako dodatečný šum. Pro úlohy, které by vyžadovaly častou změnu reference, by se dal problém vyřešit vytvořením MRP pro každou hodnotu reference zvlášť [5].

Kapitola 3

Přehled metod

V této sekci si uvedeme přehled a klasifikaci metod, které byly vypracovány pro řešení problémů popsané jako MRP. Ukážeme si také, jak by bylo možné jednotlivé metody implementovat již k samotnému návrhu řízení dynamických systémů. Řešení MRP lze nalézt dvěma možnými přístupy:[1]

- Přístup založený na Iteraci strategie
- Přístup založený na Iteraci ohodnocení

Podle způsobu jejich implementace pak dostáváme dvě hlavní skupiny metod:

- Metody Dynamického programování (DP)
- PU metody založené na Temporální diferenci (TD)

Nejprve si popíšeme implementaci daných přístupů použitím DP. Tyto metody byly historicky vyvinuty pro řešení MRP před metodami PU. I přesto, že je tato práce zaměřená zejména na metody PU, je dobré si tyto metody popsat pro snazší pochopení samotných metod PU, jelikož jsou z nich odvozené. Důvodem, proč se nejedná o metody PU je ten, že k nalezení řešení potřebují přesný popis rozhodovacího procesu (přechodových pravděpodobností), agent se zde neučí pouze na základě pozorovaných dat při interakci.

Implementace na základě PU je zajištěna za pomoci konceptu Temporální diference, díky kterému nebude nutné pro nalezení řešení znát model systému. Zaměříme se na dvě nejznámější implementace Iterace strategie a ohodnocení, po řadě známé jako algoritmus SARSA a Q-učení (*Q-learning*). V tabulce 3.1 je shrnut přehled metod a jejich zařazení do skupin.

Poznámka 3.1. V literatuře [3] je možné se namísto výše uvedeného rozdělení přístupu setkat také s rozdělením na metody následující strategii (*On-policy*) a metody, které nenásledují strategii (*Off-policy*).

[1] Je možné se setkat ještě s třetím přístupem založeným na Přímém vyhledávání strategie (Policy Search) [1]. V této práci se mu nebudeme věnovat.

Dynamické programování	Iterace strategie
	Iterace ohodnocení
PU: Temporální Diference	Iterace strategie - SARSA
	Iterace ohodnocení - Q-učení

Tabulka 3.1: Klasifikace metod pro řešení MRP

3.1 Dynamické programování

Metody dynamického programování byly vyvinuty pro řešení MRP. Potřebují ke svému řešení přesný model systému (*model-based learning*). Označují se jako offline metody, což znamená, že optimální strategie je vypočítaná pouze za využití modelu systému a ne např. z naměřených dat při přímé interakci s prostředím. Mezi metody dynamického programování patří Iterace strategie (*Policy iteration*) a Iterace ohodnocovací funkce (*Value iteration*).

3.1.1 Bellmanova rovnice

Jedním z nejdůležitějších konceptů dynamického programování je využití Bellmanova principu optimality [2]. Napíšeme-li rovnici pro ohodnocovací funkci (2.4) následovně:

$$V^h(x_k) = r(x_k, u_k) + \gamma \sum_{k=1}^{\infty} \gamma^{k-1} r(x_k, h(x_k)) \; . \tag{3.1}$$

můžeme ji poté přepsat do tvaru:

$$V^h(x_k) = r(x_k, u_k) + \gamma V^h(x_{k+1}) \; . \tag{3.2}$$

Tuto rovnici lze identifikovat jako Bellmanovu rovnici. Místo vyčíslování nekonečné sumy je možné tedy ohodnocení stavů nalézt rekurzivně. K nalezení optimálního řešení této rovnice lze použít Bellmanův princip optimality, podle kterého nezáleží na předchozích akcích, ale pouze na těch následujících, v kterých je nutné následovat optimální strategii. Optimální ohodnocovací funkce $V^*(x_k)$ je tedy dána jako:

$$V^*(x_k) = \min_{h(.)} \left(r(x_k, h(x_k)) + \gamma V^*(x_{k+1}) \right) \; . \tag{3.3}$$

Využijeme-li místo ohodnocovací funkce $V_h(.)$ funkci $Q_h(.)$, dostaneme obdobné vztahy:

$$Q^h(x_k, u_k) = r(x_k, u_k) + \gamma Q^h(x_{k+1}, u_{k+1}) \; , \tag{3.4}$$

$$Q^*(x_k, h(x_k)) = \min_{h(.)} \left[r(x_k, h(x_k)) + \gamma Q^*(x_{k+1}, h(x_{k+1})) \right] \; , \tag{3.5}$$

Pro metody dynamického programování je znalost dynamiky systému nutná právě kvůli použití výše uvedených vztahů. Z nich je možné pozorovat, že výběr optimální

akce ve stavu x_k je závislý na výběru optimální akce ve stavu x_{k+1}. Bez znalosti modelu by ale nebylo možné zjistit, jaký bude konkrétně stav x_{k+1} a ani cena $r(x_k, u_k)$ při použití akce u_k ve stavu x_k. Nebylo by možné ani vyzkoušet všechny akce ve stavu x_k, protože bez znalosti modelu není možné vrátit konfiguraci systému ze stavu x_{k+1} zpět do stavu x_k.

3.1.2 Iterace strategie

Metoda iterace strategie obsahuje dva procesy, které se nazývají Vyhodnocení strategie (*Policy evaluation*) a Vylepšení strategie (*Policy improvement*).

Nejprve je zvolena počáteční strategie $h_0(x)$. Ta musí mít vlastnost, že je asymptoticky stabilní [2]. Pomocí Bellmanovy rovnice 3.4 jsou pak pro všechny stavy $x \in X$ nalezeny příslušné hodnoty $Q^{h_i}(x, h_i(x))$, čímž je hotový proces vyhodnocení dané strategie. Díky stabilitě strategie jsou tyto Q-hodnoty konečné.

Proces vylepšení strategie spočívá v tom, že je následně pro každý stav $x \in X$ nalezena taková akce u', pro kterou platí:

$$u' = \arg\min_u Q^{h_i}(x, u),\ u \in U(x)\ . \tag{3.6}$$

Po skončení tohoto kroku je tímto nalezena nová strategie $h_{i+1}(x_k)$. Je dokázáno (za určitých podmínek), že $h_{i+1}(x_k)$, které jsou získané iterací těchto procesů, konvergují k optimální strategii $h^*(x_k)$ [3]. Algoritmus je shrnutý níže:

Algoritmus 1: Offline Iterace strategie

Vstupní data: *Počáteční strategie* $h_0(x_k)$, *Model systému* (2.12), *parametr* γ.

1. **Cyklus:** $i = 0, 1, 2, ...$

   - **Vyhodnocení strategie:** Aplikací strategie $h_i(x_k)$ nalezneme ohodnocení $Q^{h_i}(x, u)$ pro všechny $x \in X$ podle (3.4):

     $$Q^{h_i}(x_k, u_k) = r(x_k, u_k) + \gamma Q^{h_i}(x_{k+1}, u_{k+1})\ .$$

   - **Vylepšení strategie:** Nalezneme novou strategii h_{i+1}:

     $$h_{i+1}(x_k) = \arg\min_{h(.)} Q^{h_i}(x_k, h(x_k))\ .$$

   - Opakuj dokud $h_{i+1} \approx h_i$ (konvergence).

Výsledek: *Optimální strategie* $h^*(x_k)$.

Poznámka 3.2. Samotné Vyhodnocení strategie použitím Bellmanovy rovnice (3.4) je často výpočetně náročné (jedná se o řešení soustavy rovnic). Bylo ukázáno, že Bellmanova rovnice je rovnice s pevným bodem [3]. Samotné vyhodnocení strategie lze pak provést i iteračně (mluvíme pak o Iteračním vyhodnocení strategie). Na začátku

je zvoleno počáteční ohodnocení $Q^0(x_k, h_i(x_k))$. Následováním zvolené strategie h_i pak v krocích $j = 0, 1, 2, ...$ ohodnocení $Q^j(x_k, h_i(x_k))$ konverguje k $Q^{h_i}(x_k, h_i(x_k))$. Krok Vyhodnocení strategie může být pak nahrazen následujícím vztahem:

$$Q^{j+1}(x_k, u_k) = r(x_k, u_k) + \gamma Q^j(x_{k+1}, u_{k+1}) \text{ pro } j = 1, 2, \dots . \tag{3.7}$$

3.1.3 Iterace ohodnocení

Metoda Iterace ohodnocení spočívá v přímém nalezení optimální ohodnocovací funkce, která pak udává i optimální strategii $h^*(x_k)$. Na počátku jsou pro všechny páry stavů $x \in X$ a akcí $u \in U(x)$ zvoleny libovolné Q-hodnoty $Q_i(x_k, u_k)$. Následně jsou s využitím Bellmanovy rovnice (3.4) iteračně získávány nové Q-hodnoty jako:

$$Q_{i+1}(x_k, u_k) = r(x_k, u_k) + \gamma \min_{u_{k+1}} Q_i(x_{k+1}, u_{k+1}) . \tag{3.8}$$

Tato rovnice by se dala zapsat i využitím V-funkce jako:

$$Q_{i+1}(x_k, u_k) = r(x_k, u_k) + \gamma V_i(x_{k+1}) . \tag{3.9}$$

Lze ukázat [3], že hodnoty $Q_{i+1}(x_k, u_k)$ konvergují k optimální hodnotě $Q^*(x_k, u_k)$. Nakonec nalezneme optimální strategii ze vztahu (2.10). Algoritmus je shrnut níže:

Algoritmus 2: Offline Iterace ohodnocení

Vstupní data: *Počáteční ohodnocení* $Q_i(x_k, u_k)$, *Model systému* (2.12).

1. **Cyklus:** $i = 0, 1, 2, ...$
   - **Vylepšení ohodnocení:** Nalezení hodnot $Q_{i+1}(x_k, u_k)$ s využitím vztahu
     $$Q_{i+1}(x_k, u_k) = r(x_k, u_k) + \gamma \min_{u_{k+1}} Q_i(x_{k+1}, u_{k+1}) .$$
   - Opakuj, dokud $Q_{i+1} \approx Q_i$.
2. **Nalezení optimální strategie:** Platí $Q_{i+1}(.) = Q^*(.)$, využijeme vztah:
   $$h^*(x_k) = \arg\min_{h(.)} Q^*(x_k, u_k) .$$

Výsledek: *Optimální strategie* $h^*(x_k)$.

3.1.4 Příklad využití metod DP

Uvažujme model napouštění vodní nádrže popsaný rovnicí:

$$Q_{in}(t) - Q_{out}(t) = S\dot{x}(t) , \tag{3.10}$$

kde $Q_{in}(t)$, resp. $Q_{out}(t)$ je vstupní, resp. výstupní objemový průtok $[\mathrm{m}^3\,\mathrm{s}^{-1}]$, $S[m^2]$ je obsah podstavy nádrže a $x(t)[m]$ je výška hladiny vody v nádrži. Úkolem je volbou hodnot $Q_{in}(t)$ a $Q_{out}(t)$ nastavit $x(t)$ na požadovanou referenční hodnotu x_{ref}.

Aby bylo možné jednoduše znázornit princip použité metody, zvolíme parametry úlohy tak, abychom pracovali pouze s omezeným počtem hodnot $Q(x_k, u_k)$:

- Výšku hladiny v nádrži omezíme na hodnoty: $x \in \langle 0; 6\rangle$ a zároveň ji budeme pozorovat pouze v diskrétních časových okamžicích $k = 0, 1, 2, ...[s]$.
- Označíme akci $u_k := Q_{in}(t) - Q_{out}(t)$, která může nabývat pouze diskrétních hodnot $\{-1; 0; 1\}$. Změna hodnoty akce u_k bude možná také pouze pro $k = 0, 1, 2, ...[s]$.
- Zvolíme obsah nádrže $S = 1\,\mathrm{m}^2$ a referenční hodnotu $x_{ref} = 3\,\mathrm{m}$.
- Cenu definujeme jako vzdálenost od reference: $r(x_k, u_k) = |x_{ref} - x_{k+1}|$.

Poznámka 3.3. Pokud bychom neuvažovali následující omezení, tak by princip řešení úlohy byl stejný, pracovali bychom avšak se spojitým stavovým prostorem, což by mohlo přinášet řadu problémů. Řešením úloh PU ve spojitém stavovém prostoru se věnuje kapitola 4.

Použití offline Iterace strategie

Budeme postupovat podle algoritmu 1. Zvolíme[2] $\gamma = 0,5$ a počáteční (stabilní) strategii $h_0(x_k) = 0$, tedy v každém stavu bude zvolena akce $u_k = 0$.

Nyní provedeme vyhodnocení strategie $h_0(x)$ na základě rovnice 3.4 (bylo by možné postupovat iteračně, nicméně v tomto případě, vzhledem k nízkému počtu stavů, to není nutné). Díky známému modelu systému(3.10) lze pro pár stav-akce v kroku k vždy jednoznačně určit následující stav v kroku $k + 1$ (pro počáteční strategii bude následující stav roven tomu aktuálnímu). Napíšeme-li Bellmanovu rovnici pro každý stav se zvolenou strategií, dostáváme soustavu rovnic:

$$\begin{aligned} Q_0(0,0) =& |3-0| + 0,5Q_0(0,0)\ , \\ Q_0(1,0) =& |3-1| + 0,5Q_0(1,0)\ , \\ &\vdots \\ Q_0(5,0) =& |3-5| + 0,5Q_0(5,0)\ , \\ Q_0(6,0) =& |3-6| + 0,5Q_0(6,0)\ . \end{aligned} \tag{3.11}$$

V tomto případě Q-hodnoty pro jednotlivé stavy nezávisí na jiných stavech a rovnají se pouze součtu nekonečné geometrické řady s koeficientem γ a prvním členem řady

[2] V reálných případech je vhodné tento parametr volit blízko 1, zde je tato hodnota zvolena pro jednodušší výpočty.

rovnající se $|x_{ref} - x_{k+1}|$. Výsledné hodnoty jsou zobrazené v tabulce 3.2 (hodnoty v zeleně zbarvených buňkách).

Následně provedeme proces Vylepšení strategie. K tomu je potřeba vypočítat zbývající Q-hodnoty pro páry stav-akce. Např. pro stav $x = 1$ dostáváme dvě rovnice:

$$\begin{aligned} Q_0(1,-1) =& |3-0| + 0,5Q_0(0,0) \ , \\ Q_0(1,1) =& |3-2| + 0,5Q_0(2,0) \ . \end{aligned} \tag{3.12}$$

Po vypočítání všech Q-hodnot vybereme pro každý stav akci s nejnižší Q-hodnotou, tedy např. pro stav $x = 1$ vybereme akci $u = 1$. Minimální Q-hodnoty pro všechny stavy v iteraci $i = 0$ jsou napsány v tabulce 3.2 červeným písmem. Tyto akce pak udávají strategii v následující iteraci, v které postupujeme stejným způsobem. Pro konvergenci strategie k optimální bylo zapotřebí pouze dvou iterací.

	0 m	1 m	2 m	3 m	4 m	5 m	6 m
$Q_0(x,-1)$	6	6	4	2	0	2	4
$Q_0(x,0)$	6	4	2	0	2	4	6
$Q_0(x,1)$	4	2	0	2	4	6	6
$Q_1(x,-1)$	4,25	4,25	2,5	1	0	1	2,5
$Q_1(x,0)$	4,25	2,5	1	0	1	2,5	4,25
$Q_1(x,1)$	2,5	1	0	1	2,5	4,25	4,25

Tabulka 3.2: Q-hodnoty pro jednotlivé páry stav-akce při využití Iterace strategie.

Použití offline Iterace ohodnocení

Postupujeme podle algoritmu 2. Zvolíme opět $\gamma = 0,5$ a libovolné počáteční ohodnocení, např. $Q(x,u) = 4$ pro všechny páry stav-akce. Nyní provedeme vylepšení stávajícího ohodnocení podle rovnice (3.8). Například pro stav $x = 2$ dostáváme následující rovnice:

$$\begin{aligned} Q_1(2,-1) =& |3-1| + 0,5 \min_{u'} Q_0(1,u') \ , \\ Q_1(2,0) =& |3-2| + 0,5 \min_{u'} Q_0(2,u') \ , \\ Q_1(2,1) =& |3-3| + 0,5 \min_{u'} Q_0(3,u') \ . \end{aligned} \tag{3.13}$$

Výsledky pro všechny stavy jsou opět shrnuté v tabulce 3.3. Červeným písmem je vyznačena minimální Q-hodnota pro daný stav. Pro námi zvolené počáteční Q-hodnoty dostáváme optimální Q-hodnoty až v nekonečnu. Např. pro počáteční ohodnocení $Q(x,u) = 0$ by konvergence nastala již po konečném počtu kroků. Výsledné hodnoty všech Q-hodnot vyjdou stejně jako pro použití Iterace strategie. Díky zjištěným optimálním Q-hodnotám nalézáme optimální strategii $h^*(x_k)$ (v tabulce vyznačené zeleně zbarvenými buňkami).

x	0 m	1 m	2 m	3 m	4 m	5 m	6 m
$Q_0(x,-1)$	4	4	4	4	4	4	4
$Q_0(x,0)$	4	4	4	4	4	4	4
$Q_0(x,1)$	4	4	4	4	4	4	4
$Q_1(x,-1)$	5	5	4	3	2	3	4
$Q_1(x,0)$	5	5	3	2	3	5	5
$Q_1(x,1)$	4	3	2	3	4	5	5
$Q_2(x,-1)$	5	5	3,5	2	1	2	3,5
$Q_2(x,0)$	5	3,5	2	1	2	3,5	5
$Q_2(x,1)$	3,5	2	1	2	3,5	5	5
⋮							
$Q_i(x,-1)$	4,25	4,25	2,5	1	0	1	2,5
$Q_i(x,0)$	4,25	2,5	1	0	1	2,5	4,25
$Q_i(x,1)$	2,5	1	0	1	2,5	4,25	4,25

Tabulka 3.3: Q-hodnoty pro jednotlivé páry stav-akce při využití offline Iterace ohodnocení.

3.2 Metody temporální diference

Největší nevýhodou metod dynamického programování je nutná znalost přesného modelu systému, např. ve formě stavového popisu (2.12). Zároveň se také jednalo o offline metody, tedy metody, které vypočítaly optimální řešení pouze na základě daného modelu. Využití temporální diference (TD) je jednou ze základních myšlenek PU. U metod TD se již agent učí, nalézá optimální strategii pouze na základě pozorovaných dat z prostředí. Zároveň pracují online, učí se v každém časovém kroku. Díky tomu lze nalézt optimální, ale zároveň také adaptivní řešení.

S využitím TD je možné implementovat Iteraci strategie a Iteraci ohodnocení jako online metody. Uvedeme si zde dva nejdůležitější algoritmy, které se v literatuře nazývají SARSA a Q-učení. K jejich popisu využijeme opět ohodnocení pomocí Q-funkce. V tomto případě by použití V-funkce mělo svá omezení.

3.2.1 Princip temporální diference

Znalost modelu systému v metodách dynamického programování byla nutná zejména k nalezení ohodnocení $Q^h(x_k, u_k)$ pro všechny stavy x_k, neboli vyhodnocení strategie $h(x_k)$. Jak již bylo řečeno, metody TD pracují s naměřenými daty, nepoužívají model systému. K naučení těchto hodnot je tedy třeba ve stavu x_k provést konkrétní akci u_k.

Myšlenka TD je následující: namísto znalosti přesných hodnot $Q^h(x_k, u_k)$ pro

každý pár stav-akce, můžeme uchovávat pouze odhady těchto hodnot, značeno $Q^{\backsim h}(x_k, u_k)$. Po provedení konkrétní akce u_k ve stavu x_k je pokaždé pozorována cena r_{k+1} a hodnota $Q^{\backsim h}(x_{k+1}, h(x_{k+1}))$. Původní odhad $Q^{\backsim h}(x_k, u_k)$ je poté aktualizován na základě těchto nově získaných hodnot. K tomuto účelu si definujeme TD-chybu (temporal difference error):

$$\delta_k = [r_{k+1} + \gamma Q^{\backsim h}(x_{k+1}, u_{k+1})] - Q^{\backsim h}(x_k, u_k) \ . \tag{3.14}$$

Jedná se o rozdíl mezi předpokládaným ohodnocením $Q^{\backsim h}(x_k, u_k)$ a nově pozorovaným ohodnocením $[r_{k+1} + \gamma Q^{\backsim h}(x_{k+1}, u_{k+1})]$. Je možné si všimnout, že tento vztah je odvozen ze vztahu 3.4 pro $\delta_k = 0$. Bylo ukázáno [3], že aktualizace ohodnocení na základě TD konverguje k $Q^h(x_k, u_k)$. V tom případě poté bude ohodnocení $Q^{\backsim h}(x_k, u_k) = Q^h(x_k, u_k)$ a TD-chyba bude nulová. Nadále již nebudeme kvůli přehlednosti využívat značení $Q^{\backsim h}(.)$.

3.2.2 Problém explorace a exploatace

Při využití metod nezávislých na modelu systému vzniká problém, který je v literatuře [3] nazýván jako rovnováha mezi explorací a exploatací (*exploration vs. exploitation*). Explorace (objevování) je proces, při kterém agent zkouší různé akce k nalezení neznámých, případně vylepšení odhadů hodnot $Q^h(x_k, u_k)$ pro všechny páry stav-akce. Získává tím nové zkušenosti, které mohou vést k nalezení dalších řešení. Exploatace (využití) je proces, při kterém je využívána znalost hodnot $Q^h(x_k, u_k)$ pro minimalizaci dlouhodobé ceny. Jedná se o výběr hladové (*greedy*) akce vzhledem k ohodnocení. V metodách DP jsou všechny hodnoty $Q^h(x_k, u_k)$ vypočítány, používá se proto pouze proces exploatace. Pro metody TD je nutné tyto hodnoty nalézt přímým experimentem.

Najít rovnováhu mezi těmito procesy je velice důležité. Jestliže by míra explorace byla příliš vysoká, bylo by možné, že by agent dostatečně nevyužil získané hodnoty. Konvergence k optimálnímu řešení by pak mohla být příliš pomalá, případně by učení nemuselo vůbec konvergovat. Naopak jestliže by agent výhradně preferoval exploataci, bylo by možné, že by řešení konvergovalo pouze k suboptimálnímu řešení. To by bylo způsobené právě tím, že by agent pracoval pouze s omezenými znalostmi prostředí.

Nejčastěji se tento problém řeší pomocí mechanismu náhodné selekce. Ten funguje tak, že namísto exploatační akce je s jistou pravděpodobností vybrána jiná, explorační akce. Jedná se o rozšíření strategie, kde namísto přesného mapování ze stavu na akci (výběr optimální akce vzhledem k Q-hodnotě), je přidána možnost náhodného výběru. Běžně se používá ε-hladová selekce (ε-*greedy*) nebo softmax selekce [3].

- ε-hladová selekce je nejčastější řešení pro výběr náhodné akce. Tento mechanismus funguje na principu toho, že s danou pravděpodobností (1-ε) je vybrána exploatující akce (hladová vzhledem k ohodnocení) a s pravděpodobností ε je

vybrána náhodná (explorační) akce $u \in U(x_k)$. Tedy:

$$u_k = \begin{cases} \arg\min_{u_k} Q(x_k, u_k) & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_k) & \text{s pravděpodobností } \varepsilon_k. \end{cases} \tag{3.15}$$

Parametr ε_k nemusí být během celého učení konstantní, ale může se vyvíjet v závislosti na kroku k. Často se volí na počátku tento parametr vysoký a postupně se snižuje až k nule.

- Softmax selekce pracuje na stejném principu jako ε-hladová selekce s tím rozdílem, že výběr explorační akce nemá rovnoměrné rozdělení. Pravděpodobnost použití akce u_k je vztažená k její Q-hodnotě. Jsou tak upřednostňovány akce, které mají nižší Q-hodnotu. K tomu lze použít např. Boltzmannovo rozdělení:

$$u_k = u \in U(x_k) \quad \text{s pravděpodobností} \quad \frac{\exp(-Q(x_k, u_k)/\tau)}{\sum_u \exp(-Q(x_k, u)/\tau)}. \tag{3.16}$$

Role jmenovatele v uvedeném vztahu je ta, aby součet pravděpodobností všech akcí v daném stavu byl roven jedné. Parametr $\tau \in (0; \infty)$ je označován jako *teplota*. Jestliže je tento parametr blízko nuly, je tato selekce ekvivalentní hladové selekci. Naopak pro vysoké hodnoty je pravděpodobnost selekce pro všechny akce u stejná.

Použití konkrétní selekce závisí zejména na typu úlohy. Výhodou ε-hladové selekce je jednoduchá implementace a jednoduchá volba parametru. Stejná pravděpodobnost pro všechny možné akce avšak může být i nevýhodou. Např. v případech, kdy by opakovaná volba špatné akce mohla mít destruktivní vliv na agenta či systém, je lepší použít Softmax selekci.

3.2.3 SARSA - Online Iterace strategie

Algoritmus SARSA[3] je online implementací Iterace strategie. Při popisu budeme vycházet z již popsaného algoritmu offline Iterace strategie. Algoritmus SARSA opět obsahuje procesy vyhodnocení a vylepšení strategie, které zde avšak nejsou až tak patrné.

Namísto explicitní volby počáteční strategie $h_0(x_k)$, jsou zvoleny již samotné hodnoty $Q_0(x_k, u_k)$ (libovolně). Ty poté zpětně určují strategii volby akcí (vybírány jsou hladové akce). Iteračně jsou pak Q-hodnoty (a tedy i strategie) vyhodnocovány na základě vztahu:

$$Q(x_k, u_k) = Q(x_k, u_k) + \alpha[r_{k+1} + \gamma Q(x_{k+1}, u_{k+1}) - Q(x_k, u_k)]\,, \tag{3.17}$$

[3]Název algoritmu je odvozen od hodnot, které se využívají k aktualizace ohodnocení páru stav-akce. V anglické terminologii strojového učení se jedná po řadě o hodnoty (**S**tate(k), **A**ction(k), **R**eward(k), **S**tate(k+1), **A**ction(k+1).

kde akce u_k, u_{k+1} jsou v každém kroku vybírány hladově vůči aktuálním ohodnocení (s použitím mechanismu náhodné selekce, např. (3.15)). Parametr $\alpha \in (0; 1\rangle$ se nazývá míra učení. Udává nám, jak moc nově získaná informace (TD-chyba) změní starou informaci (aktuální odhad Q-hodnoty). Je vhodné si všimnout, že vztah (3.17), aktualizuje hodnoty $Q(x_k, u_k)$ na základě $Q(x_{k+1}, u_{k+1})$, tedy stejně jako offline Iterace strategie, viz (3.4).

Jinými slovy, proces vyhodnocení strategie je přítomný v každém kroku k, kdy jsou postupně získávány lepší odhady hodnot $Q(x_k, u_k)$ (iterativní vyhodnocení strategie). Zároveň díky tomuto zpřesňování odhadů ohodnocení (i za pomocí náhodných akcí) jsou nalézány akce s nižšími Q-hodnotami. Vylepšení strategie tedy nastává tehdy, když je v daném stavu nalezena akce u'_k s nižší hodnotou ohodnocení $Q(x_k, u'_k)$ než původní akce u_k, jelikož jsou vybírány hladové akce. Algoritmus je shrnut v 3.

Algoritmus 3: SARSA: online iterace strategie s ε-hladovou selekcí

Vstupní data: *Počáteční ohodnocení* $Q_0(x_k, u_k)$ *počáteční stav* x_0*, parametry:* ε_k, α_k, γ.

1. **Inicializace:**
   Ve stavu x_0 vybereme akci:

$$u_0 = \begin{cases} \arg\min_{u'} Q(x_0, u') & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_0) & \text{s pravděpodobností } \varepsilon_k. \end{cases}$$

2. **Cyklus:** $k = 0, 1, 2, ...$
   Pozorujeme x_{k+1}, r_{k+1} a vybereme akci:

$$u_{k+1} = \begin{cases} \arg\min_{u'} Q(x_{k+1}, u') & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_{k+1}) & \text{s pravděpodobností } \varepsilon_k. \end{cases}$$

   Aktualizujeme ohodnocení $Q(x_k, u_k)$:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha[r_{k+1} + \gamma Q_k(x_{k+1}, u_{k+1}) - Q_k(x_k, u_k)] \,.$$

3.2.4 Q-učení - Iterace ohodnocení

Algoritmus Q-učení je opět pozměněný algoritmus DP Iterace ohodnocení použitím TD. Na začátku jsou, stejně jako v algoritmu SARSA, vybrány libovolně Q-hodnoty. Vylepšení tohoto ohodnocení je ale pak provedeno podle vztahu:

$$Q(x_k, u_k) = Q(x_k, u_k) + \alpha[r_{k+1} + \gamma \min_{u'} Q(x_{k+1}, u') - Q(x_k, u_k)] \,, \tag{3.18}$$

kde akce u_k je opět vybrána jako hladová (s náhodnou selekcí). Výrazným rozdílem je použití hodnoty $\min_{u'} Q(x_{k+1}, u')$ namísto $Q(x_{k+1}, u_{k+1})$ u SARSA. Q-učení tedy nevylepšuje ohodnocení na základě opravdu použité akce ve stavu x_{k+1}, ale odhadované akce (s minimální Q-hodnotou), která bude použita. Tento způsob vylepšování ohodnocení je opět obdobou offline Iterace ohodnocení, viz. (3.8). Algoritmus je shrnut na 4.

Algoritmus 4: Q-učení: online iterace ohodnocení s ε-hladovou selekcí

Vstupní data: *Počáteční ohodnocení* $Q_i(x_k, u_k)$*, počáteční stav* x_0*, parametry:* ε_k, α_k, γ.

1. **Cyklus:** $k = 0, 1, 2, ...$
   Ve stavu x_k vybereme akci:

$$u_k = \begin{cases} \arg\min_{u'} Q(x_k, u') & \text{s pravděpodobností (1-}\varepsilon_k\text{),} \\ u \in U(x_k) & \text{s pravděpodobností } \varepsilon_k. \end{cases}$$

   Pozorujeme x_{k+1}, r_{k+1}. Aktualizujeme ohodnocení $Q(x_k, u_k)$:

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha[r_{k+1} + \gamma \min_{u'} Q_k(x_{k+1}, u') - Q_k(x_k, u_k)] \,.$$

Kapitola 4

Aproximace v PU

Pro metody, které jsme si popsali v kapitole 3, bylo zapotřebí mít uloženou příslušnou Q-hodnotu pro všechny páry stav-akce. Ohodnocovací funkce byla tak realizovaná ve formě tabulky, jak bylo např. ukázáno na příkladě 3.1.4. Tato realizace je však vhodná pouze pro diskrétní systémy s konečným počtem stavů a akcí.

Pro systémy se spojitým stavovým prostorem a tedy s nekonečně mnoha stavy, by realizace ohodnocovací funkce ve formě tabulky nebyla možná. Případná velice jemná diskretizace stavů by také nebyla vhodnou volbou, jelikož by s sebou přinášela řadu dalších problémů, souhrnně označovaných jako Prokletí rozměrnosti (*Curse of dimensionality*).

Abychom mohli využít efektivně metody PU i pro spojité, či vícerozměrné systémy, je nutné zvolit takovou reprezentaci ohodnocovací funkce (Q-funkce nebo V-funkce), která nezpůsobí dané problémy. Jednou z možností je použití její parametrické aproximace. Díky této aproximaci bude možné pracovat pouze s podmnožinou celého stavového prostoru a zároveň nám dá možnost zobecnění již získaných hodnot i na celý stavový prostor. Nadále budeme již uvažovat pouze aproximaci Q-funkce.

4.1 Parametrická aproximace funkcí

Parametrická aproximace funkcí patří mezi metody Učení s učitelem, při kterém je požadovaná funkce, v našem případě $Q(x, u)$, aproximována za pomocí n-dimenzionálního vektoru. Přesněji, je definován vektor reálných parametrů

$$\boldsymbol{\theta} = [\theta_1, \theta_2, ..., \theta_n]^T, \tag{4.1}$$

a zobrazení

$$F : \mathbb{R}^n \to \mathbb{D}\ , \tag{4.2}$$

kde $\mathbb{R}^n$ je prostor parametrů a $\mathbb{D}$ je prostor Q-funkcí. Aproximaci funkce $Q(x, u)$, kterou označíme jako $\widetilde{Q}(x, u)$, pak definujeme jako (pro zjednodušení zápisu nebudeme uvádět explicitní závislost funkce $\widetilde{Q}(x, u)$ na vektoru parametrů $\boldsymbol{\theta}$):

$$\widetilde{Q}(x, u) = [F(\theta)](x, u)\ . \tag{4.3}$$

Úlohou tohoto učení je nalézt takový vektor parametrů $\boldsymbol{\theta}$ tak, aby rozdíl mezi funkcemi $Q(x,u)$ a $\widetilde{Q}(x,u)$ byl co nejmenší. Tento rozdíl lze kvantifikovat např. pomocí kvadratického průměru aproximační chyby (RMSE) [3]. Díky aproximaci pak není třeba uchovávat Q-hodnoty pro všechny navštívené stavy, ale pouze konstantní počet parametrů n. Počet parametrů je volen tak, aby byl mnohem menší, než případný počet všech Q-hodnot bez použití této aproximace.

4.2 Gradientní sestup

K nalezení vektoru parametrů (4.1) si ukážeme metodu založenou na Gradientním sestupu (*Gradient descent*). Pro tyto účely je potřeba předpokládat, že funkce $\widetilde{Q}(x,u)$ je diferencovatelná vzhledem k $\boldsymbol{\theta}$ v každém bodě jejího definičního oboru.

Tato metoda vychází z toho, že za pomocí vzorků z funkce, kterou chceme aproximovat, lze iteračně vylepšovat parametry $\boldsymbol{\theta}$ tak, aby se každým krokem snižovala aproximační chyba. Při aplikaci v online metodách PU lze tato vylepšení realizovat v každém časovém kroku k, kdy jsou interakcí se systémem získávány hodnoty $Q_k(x_k,u_k)$. Vylepšený vektor parametrů je pak dán vztahem:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta_k} + \alpha \left[Q_k(x_k,u_k) - \widetilde{Q}_k(x_k,u_k) \right] \nabla_\theta \widetilde{Q}(x,u) \ , \tag{4.4}$$

kde parametr $\alpha \in \langle 0;1 \rangle$ je označován jako délka kroku a operátor ∇_θ značí gradient funkce vzhledem k parametrům $\boldsymbol{\theta}$.

4.3 Lineární metody

Obecně může být zobrazení 4.2 nelineární vzhledem k parametrům. Jedná se např. o použití neuronových sítí, fuzzy aproximace, apod.. V PU je nicméně nejčastěji využíváno takové zobrazení, které je lineární vzhledem k parametrům (zkráceně lineární).

Pro účely lineární parametrické aproximace Q-funkce definujeme vektor bázových funkcí:

$$\boldsymbol{\phi}(x,u) = [\phi_1(x,u), \phi_2(x,u), ..., \phi_n(x,u)]^T \ , \tag{4.5}$$

kde každá bázová funkce $\phi_i(x,u)$ přiřadí každému páru stav-akce reálné číslo, obvykle z intervalu $\langle 0;1 \rangle$. Počet prvků vektoru $\boldsymbol{\phi}(x,u)$ je vždy roven počtu prvků vektoru $\boldsymbol{\theta}$ z toho důvodu, aby ke každému parametru θ_i existovala příslušná bázová funkce $\phi_i(x,u)$. S pomocí vektoru bázových funkcí pak definujeme $\widetilde{Q}(x,u)$ jako:

$$\widetilde{Q}(x,u) = \sum_{i=1}^{n} \theta_i \phi_i(x,u) = \boldsymbol{\theta}^T \boldsymbol{\phi}(x,u) \ . \tag{4.6}$$

Lze pozorovat, že $\widetilde{Q}(x,u)$ je dána jako lineární kombinace prvků vektoru $\boldsymbol{\phi}(x,u)$, kde jednotlivé koeficienty jsou dány vektorem $\boldsymbol{\theta}$. Každá bázová funkce $\phi_i(x,u)$ pak

vlastně určuje vliv daného parametru θ_i na aproximovanou funkci. Použití lineárních metod přináší zejména výhodu ve výpočtu $\nabla_\theta \widetilde{Q}(x,u)$, kdy s využitím linearity (vztahu (4.6)) dostáváme, že:

$$\nabla_\theta \widetilde{Q}(x,u) = \boldsymbol{\phi}(x,u)\ . \tag{4.7}$$

Představíme si dvě metody, kterými lze bázové funkce realizovat:

- Agregace stavů
- Radiální bázové funkce

Pro obě metody platí, že využívají pouze konečnou množinu akcí $U_d = \{u_1, u_2, \ldots, u_m\}$, kterou získáme výběrem m akcí z celé množiny U. Tím se nám zjednoduší výběr minimální akce v daném stavu.

4.3.1 Agregace stavů

Uvažujme množinu stavů X rozdělenou na l disjunktních množin: $X = X_1 \cup X_2 \cup \cdots \cup X_l$. Tento proces nazveme Agregací stavů [1], X_i pak Agregovaný stav. Často jsou sjednoceny ty stavy, jejichž vzdálenost ve stavovém prostoru je nízká. Všechny stavy náležící do agregovaného stavu X_i pak při volbě akce $u' \in U$ sdílí hodnotu $\widetilde{Q}(X_i, u')$, čímž se výrazně sníží počet ukládaných Q-hodnot. Jedná se o jednu z nejjednodušších metod, kterými lze aproximovat funkci $Q(x,u)$.

K tomuto účelu je definován vektor bázových funkcí s $n = l \times m$ bázovými vektory:

$$\boldsymbol{\phi} = [\phi_1(x_1,u_1), ..., \phi_{m(i-1)+1}(x_i,u_1), \phi_{m(i-1)+2}(x_i,u_2), \ldots, \phi_n(x_l,u_m)]^T\ , \tag{4.8}$$

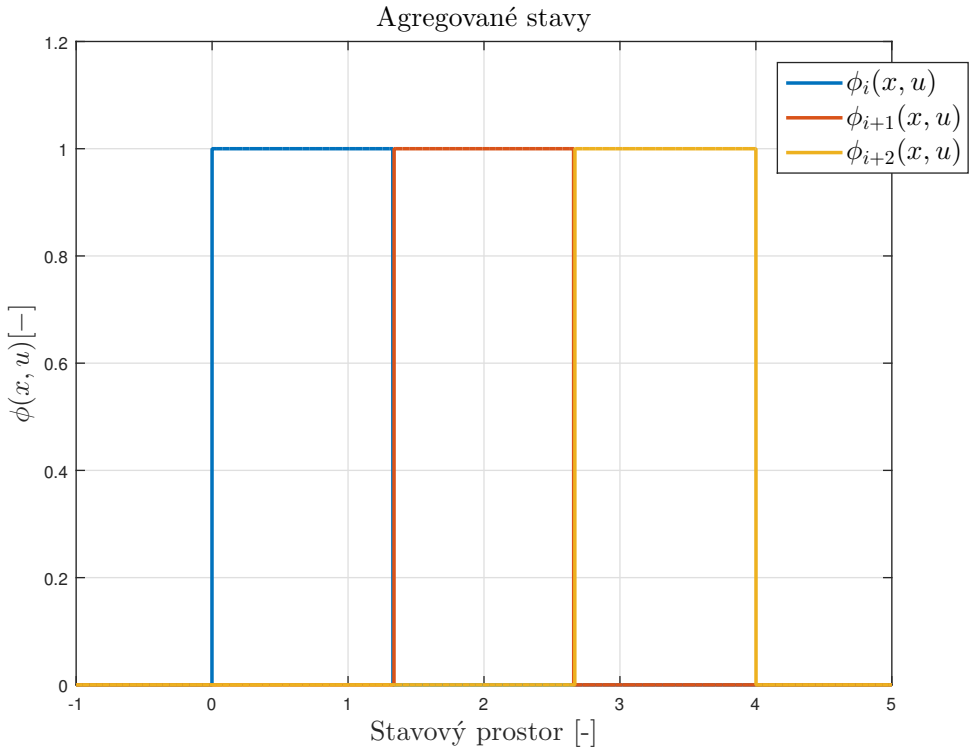
kde

$$\phi_{m(i-1)+j}(x,u) = \begin{cases} 1 & \text{pokud } x \in X_i \text{ a } u = u_j \\ 0 & \text{jinak} \end{cases} \tag{4.9}$$

Jednotlivé bázové funkce (4.9) jsou definované tak, že při dosazení konkrétního páru (x_i, u_j) do všech bázových funkcí je nenulová právě jedna z nich, s indexem $m(i-1)+j$, kde m je počet možných akcí. Jelikož je příslušná bázová funkce $\phi_{m(i-1)+j} = 1$ a hodnota $\widetilde{Q}(X_i, u_j)$ je dána vztahem (4.6), pak hodnota $\widetilde{Q}(X_i, u_j)$ se rovná hledanému parametru $\theta_{m(i-1)+j}$. Příklad pokrytí jednorozměrného stavového prostoru je ukázán na obr. 4.1.

4.3.2 Radiální bázové funkce

Na radiální bázové funkce (RBF) lze pohlížet jako na zobecnění agregace stavů. Použití RBF vede k pokrytí stavového prostoru množinami s definovaným středovým stavem μ_i. Tyto množiny, oproti agregaci stavů, mohou mít neprázdné průniky, stav $x \in X$ tak může náležet do více těchto množin najednou. Zároveň RBF nenabývají



Obrázek 4.1: Pokrytí 1-D stavového prostoru Agregovanými stavy.

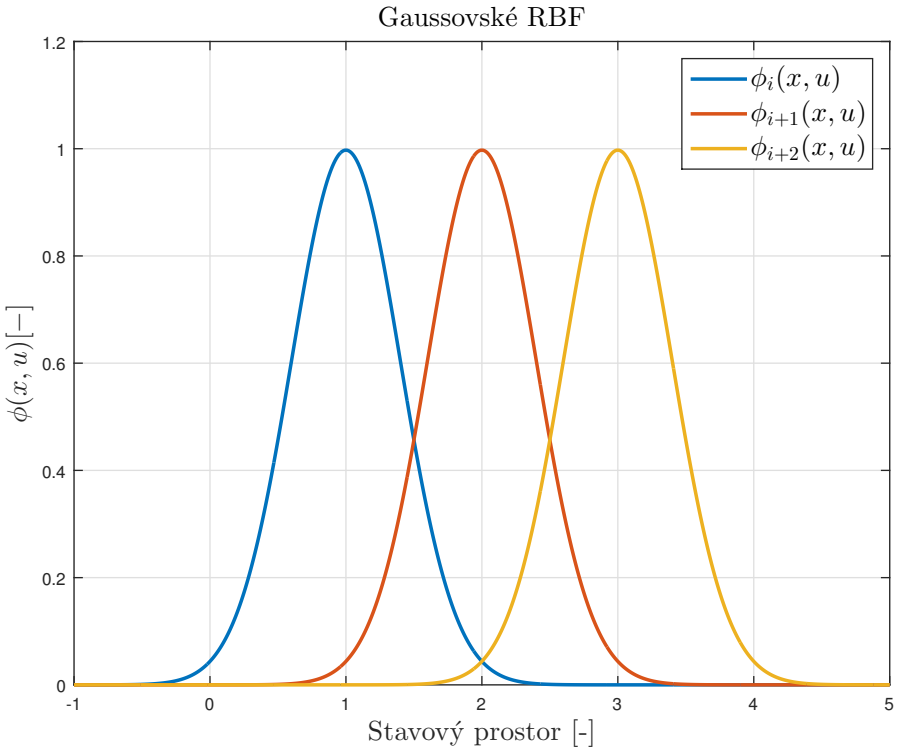
pouze hodnot $\{0;1\}$, ale mohou nabývat všech hodnot v intervalu $\langle 0;1\rangle$. Pro konkrétní stav $x \in X$ pak tato hodnota kvantifikuje vzdálenost stavu x od středu μ_i.

Typicky se využívají Gaussovské RBF, definované následovně:

$$\phi_{k(i-1)+j}(x,u) = \begin{cases} \exp\left(-\frac{||x-\mu_i||}{2\sigma_i^2}\right) & \text{pokud } u = u_j \\ 0 & \text{pokud } u \neq u_j \end{cases} \tag{4.10}$$

Jako norma $||.||$ je nejčastěji volena euklidovská metrika a σ_i je parametr, označován jako rozptyl. Na obr. 4.2 je vyobrazen příklad pokrytí jednodimenzionálního stavového prostoru pomocí těchto bázových funkcí. Intuitivně, hodnota $\widetilde{Q}(x,u)$ je dána jako lineární kombinace Q-hodnot blízkých centrálních stavů, která je vážená příslušnou vzdáleností od nich.

Poznámka 4.1. Bázové funkce jsou konstruovány tak, aby se lišily v závislosti na umístění ve stavovém prostoru. Z toho vyplývá, že všechny funkce $\phi(x,u)$ s indexem $m(i-1)+j$, $j = 1,2,\ldots,m$ jsou totožné. Vektor je nutné takto zkonstruovat, aby existovaly bázové funkce pro všechny páry stav-akce a tedy vektorové násobení (4.6) bylo definované.



Obrázek 4.2: Pokrytí 1-D stavového prostoru Gaussovskými RBF.

4.4 SARSA a Q-učení s aproximací

V této sekci si ukážeme již konkrétní využití aproximace v algoritmech SARSA a Q-učení. K aproximaci bude tedy využita lineární parametrická aproximace, kde aktualizace vektoru parametrů bude prováděna gradientním sestupem. Postupy jsou shodné pro oba typy bázových funkcí.

Při popisu metod s aproximací vyjdeme z algoritmů (3) a (4). Tyto algoritmy je nutné upravit tak, aby namísto Q-hodnot byly využívány jejich aproximace $\widetilde{Q}(x_k, u_k)$, které jsou upravovány za pomocí parametrů $\boldsymbol{\theta}_k$.

Na počátku je zvolen libovolný vektor $\boldsymbol{\theta}_k$, všechny parametry lze např. položit nule. Následně je tento parametr v každém časovém kroku k vylepšován gradientním sestupem podle vztahu (4.4).

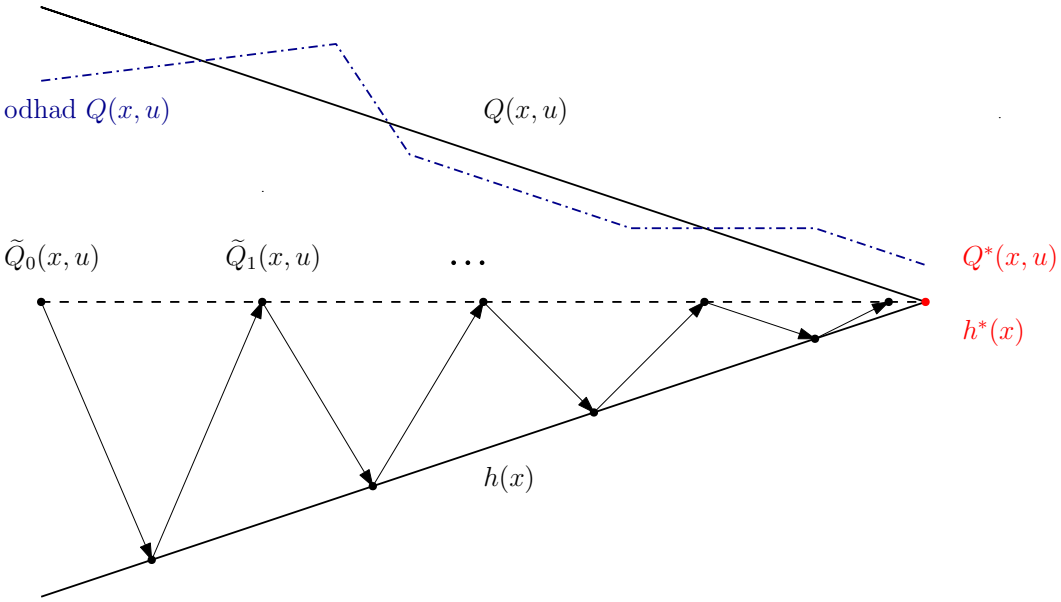
Je nutné si uvědomit, že vztah (4.4) uvažuje pro aktualizaci parametrů $\boldsymbol{\theta}_k$ již přesné hodnoty $Q_k(x_k, u_k)$. V metodách založených na TD jsou nicméně tyto hodnoty známy až při konvergenci učení. Namísto přesných hodnot jsou proto použity jejich odhady. Pro metody SARSA a Q-učení jsou odhady následující:

- SARSA: $Q_k(x_k, u_k) = r_{k+1} + \gamma \widetilde{Q}_k(x_{k+1}, u_{k+1})$

- Q-učení: $Q_k(x_k, u_k) = r_{k+1} + \gamma \min_{u'} \widetilde{Q}_k(x_{k+1}, u')$

Kvůli použití těchto odhadů není žádoucí, aby získané hodnoty $Q_k(x_k, u_k)$ měly v každém kroku příliš velký vliv na aktualizaci vektoru $\boldsymbol{\theta}$. Zejména se jedná o případy, kdy je získaný odhad $Q_k(x_k, u_k)$ nepřesný či zašuměný a mohl by tak vést ke špatné změně vektoru $\boldsymbol{\theta}$. K omezení vlivu slouží právě parametr α. Ten je vhodné volit tak, aby se v průběhu učení postupně snižoval až k nule.

I přesto, že TD metody pracují s $\widetilde{Q}(x, u)$, je i pro tento proces, za jistých podmínek, dokázána konvergence k alespoň lokálnímu optimu $Q^*(x, u)$ [3]. Jednou z podmínek je právě snižování parametru α. Situace je zobrazená na obr. 4.3.



Obrázek 4.3: Konvergence Q-funkce při použití funkční aproximace.

V každém kroku je za pomocí hodnot $\widetilde{Q}_k(x_k, u')$ vybrána akce, v případě TD metod opět s ϵ-hladovým výběrem:

$$h(x_k) = u_k = \begin{cases} \arg\min_{u'}(\widetilde{Q}_k(x_k, u_k)) & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_k) & \text{s pravděpodobností } \varepsilon_k \end{cases}$$

Následně jsou aktualizované hodnoty $\boldsymbol{\theta}$, tudíž i funkce $\widetilde{Q}(x, u)$. Algoritmy jsou pro SARSA i Q-učení shrnuty v (5) a (6).

Algoritmus 5: SARSA: s ε-hladovou selekcí a lineární aproximací funkce

Vstupní data: *Počáteční vektor parametrů* $\boldsymbol{\theta}_0$ *počáteční stav* x_0*, parametry:* ε_k, α_k, γ.

1. **Inicializace:**
   Ve stavu x_0 vybereme akci:

$$u_0 = \begin{cases} \arg\min_{u'}(\boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_k, u')) & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_0) & \text{s pravděpodobností } \varepsilon_k. \end{cases}$$

2. **Cyklus:** $k = 0, 1, 2, ...$
   Pozorujeme x_{k+1}, r_{k+1} a vybereme akci:

$$u_{k+1} = \begin{cases} \arg\min_{u'}(\boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_k, u')) & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_{k+1}) & \text{s pravděpodobností } \varepsilon_k. \end{cases}$$

   Aktualizujeme vektor parametrů $\boldsymbol{\theta}_k$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha[r_{k+1} + \gamma \boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_{k+1}, u_{k+1}) - \boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_k, u_k)]\boldsymbol{\phi}(x_k, u_k) \, .$$

Algoritmus 6: Q-učení: online iterace ohodnocení s ε-hladovou selekcí

Vstupní data: *Počáteční vektor parametrů* $\boldsymbol{\theta}_0$*, počáteční stav* x_0*, parametry:* ε_k, α_k, γ.

1. **Cyklus:** $k = 0, 1, 2, ...$
   Ve stavu x_k vybereme akci:

$$u_k = \begin{cases} \arg\min_{u'}(\boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_k, u')) & \text{s pravděpodobností (1-}\varepsilon_k\text{)}, \\ u \in U(x_k) & \text{s pravděpodobností } \varepsilon_k. \end{cases}$$

   Pozorujeme x_{k+1}, r_{k+1}. Aktualizujeme vektor parametrů $\boldsymbol{\theta}_k$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha[r_{k+1} + \gamma \min_{u'} \boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_{k+1}, u') - \boldsymbol{\theta}_k^T \boldsymbol{\phi}(x_k, u_k)]\boldsymbol{\phi}(x_k, u_k) \, .$$

Kapitola 5

Experimentální ověření metod PU

Tato kapitola se zabývá experimentálním ověření metod PU. Ukážeme si implementaci již konkrétní metody PU pro zpětnovazební řízení dynamického systému. Experimenty budeme provádět pouze na simulaci daného systému, abychom se vyhnuli případným problémům spojených s reálným systémem (šum, saturace akčních členů apod.). Pro numerické výpočty a simulace je použit program MATLAB.

5.1 Model DC motoru

Jako dynamický systém, na kterém otestujeme vybrané metody, je zvolen systém druhého řádu: lineární model DC (stejnosměrného) motoru. Rovnice popisující dynamiku systému jsou:

$$\begin{aligned} J\frac{\mathrm{d}\omega(t)}{\mathrm{d}t} + b\omega(t) &= K_t i(t) \,, \\ L\frac{\mathrm{d}i(t)}{\mathrm{d}t(t)} + Ri(t) &= u(t) - K_e\omega(t) \,, \end{aligned} \tag{5.1}$$

se stavovými veličinami:

- $\omega(t)\,[\mathrm{rad\,s^{-1}}]$: úhlová rychlost otáčení motoru,
- $i(t)\,[\mathrm{A}]$: elek. proud protékající obvodem.

Vstupem do systému je elek. napětí $u(t) = \langle -10; 10\rangle$ [V]. Hodnoty a popis parametrů jsou shrnuty v tabulce 5.1. Diskretizací systému se vzorkovací periodou $t_d = 0,01\,\mathrm{s}$ a převedením na stavový popis(2.13) dostáváme matice A, B:

$$A = \begin{bmatrix} 0,9048 & 0,0463 \\ -0,0019 & 0,9607 \end{bmatrix}, \quad B = \begin{bmatrix} 0,0005 \\ 0,0196 \end{bmatrix} \,, \tag{5.2}$$

Parametr	Hodnota	Jednotka	Popis
b	0,2	$\mathrm{N\,m\,s}$	Koeficient viskózního tření
J	0,02	$\mathrm{kg\,m^2}$	Moment setrvačnosti motoru
Ke	0,1	$\mathrm{V\,rad^{-1}}$	Konstanta zpětné elektromotorické síly
Kt	0,1	$\mathrm{N\,m\,A^{-1}}$	Momentová konstanta motoru
L	0,5	H	Indukčnost vinutí obvodu
R	2	Ω	Odpor vinutí obvodu

Tabulka 5.1: Hodnoty parametrů pro DC motor a jejich popis.

5.2 Implementace Q-učení s aproximací

K řízení systému popsaného v sekci 5.1 použijeme Q-učení s parametrickou aproximací. Otestujeme využití agregace stavů i Gaussovských RBF.

Jako hlavní úkol pro návrh řízení si stanovíme regulaci rychlosti motoru ω_k k referenční hodnotě $\omega_{ref} = 1$. Předpokládáme, že dynamika systému, rovnice (5.1), nejsou známy. Stanovíme si, že agent může měřit pouze hodnoty ω_k, hodnoty proudu i_k již ne. Pro agenta je tak stav systému x_k dán pouze rychlostí ω_k.

5.2.1 Volba parametrů

Jelikož nás zajímá regulace systému k referenci $\omega_{ref} = 1$, omezíme se při simulacích pouze na interval $x \in \langle -1,5; 1,5\rangle$. Při agregaci stavů je tento interval ekvidistantně rozdělen na 31 agregovaných stavů, pro RBF volíme 27 množin definované vztahem (4.10). Diskrétní množinu akcí volíme:

$$U_d = \{-8; -7; \ldots; 0; \ldots; 7; 8\} \ .$$

Další parametry byly zvoleny následovně:

- $\alpha = \dfrac{T-k}{T}$, kde T je celkový počet iterací, k je současná iterace;
- $\gamma = 0,95$;
- $\epsilon = 0,01$;
- $\sigma = 0,45$ (pro Gaussovské RBF).

5.2.2 Definice ceny

Díky znalosti ω_k známe v každém kroku vzdálenost stavu od reference ω_{ref}. Tuto vzdálenost lze pak využít k definici ceny r_{k+1} následovně:

$$r_{k+1} = 1 - \exp(-d(\omega_{k+1}, \omega_{ref})) \ , \tag{5.3}$$

kde jako $d(\omega_{k+1}, \omega_{ref})$ je zvolena Euklidova vzdálenost:

$$d(\omega_{k+1}, \omega_{ref}) = |\omega_{k+1} - \omega_{ref}| \,. \tag{5.4}$$

Výhodou definice ceny podle vztahu (5.3) namísto např. použití pouze Euklidovy vzdálenosti je saturace hodnoty pro velké vzdálenosti stavů od reference.

5.3 Výsledky simulací

Pro ověření implementovaných metod otestujeme regulaci rychlosti ω_k k dané referenční hodnotě ω_{ref} a poté schopnost agenta potlačit poruchu zavedenou do systému.

5.3.1 Regulace rychlosti

Regulaci systému k referenci provedeme ve dvou bězích. V obou případech bude na počátku systém v klidovém stavu $x = [0, 0]^T$. V prvním běhu se bude jednat o agenta s počáteční, libovolně zvolenou ohodnocovací funkcí a konstantami nastavenými dle sekce 5.2.1. Tento běh zastavíme při nalezení strategie h, která dokáže regulovat rychlost ke zvolené referenci.

V druhém běhu demonstrujeme strategii, kterou se agent naučil během prvního běhu. Aby se agent držel dané strategie, zvolíme následující nastavení konstant:

- $\alpha = 0$,
- $\gamma = 0$.

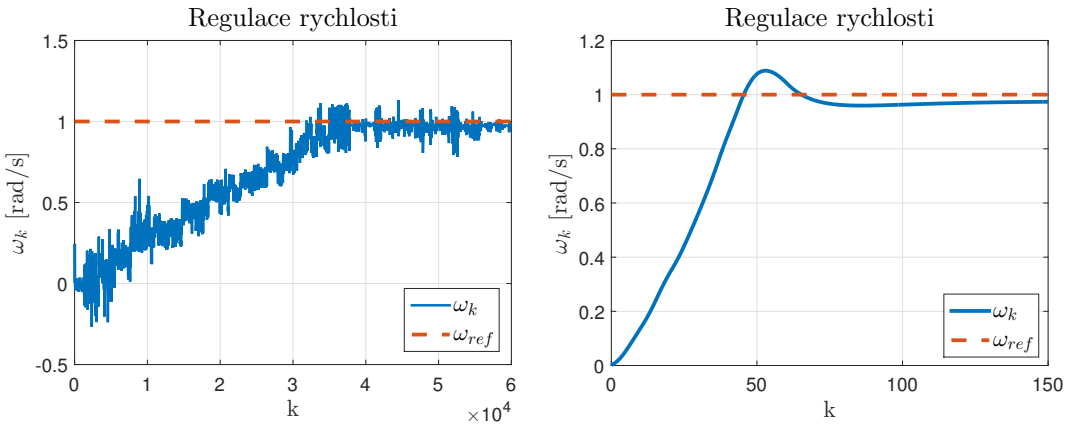
Díky tomu agent již v druhém běhu nebude strategii měnit, či volit explorační akce.

Agregace stavů

Na obr. 5.1a je zobrazen první běh simulace regulace rychlosti motoru. Ke nalezení řešení dochází po $\sim 5 \times 10^4$ krocích. Druhý běh je zobrazen na obr. 5.1b. Velikost akce při ustálení činí $u_k = 4$, viz obr. 5.2a. Kvůli diskrétní množině akcí avšak agent nedokáže regulovat rychlost přímo na referenční hodnotě. V ustálení je hodnota výstupu rovna $\omega \doteq 0,97$.

Pro znázornění vývoje hodnot $Q(x_k, u_k)$ při učení jsou na obr. 5.3 zobrazeny hodnoty před a následně po prvním běhu programu. Před prvním během jsou Q-hodnoty zvoleny náhodně, jak je ukázáno na 5.3a. Jelikož se agent během prvního běhu pohyboval zejména v intervalu $(0; 1)$, jsou získány dobré odhady Q-hodnot zejména pro tento interval, viz 5.3b.

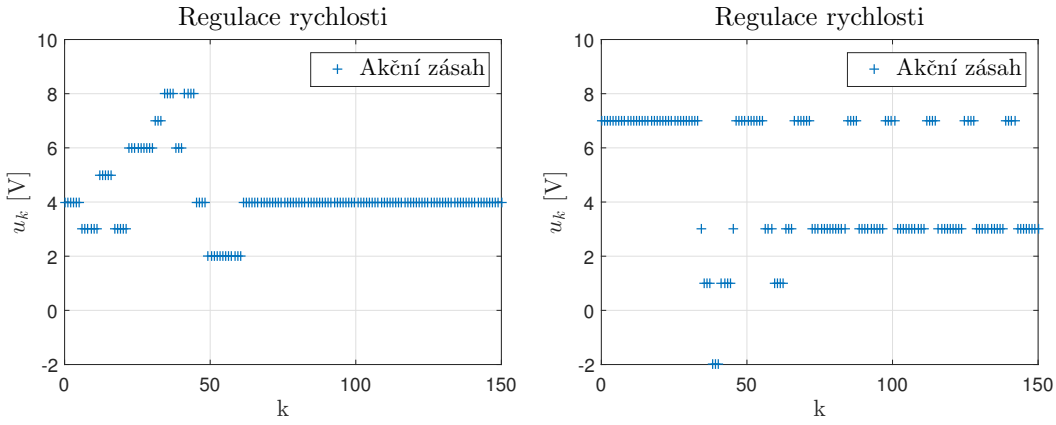
Kvůli zvolené funkci ceny (5.3) jsou rozdíly v hodnotách $Q(x_k, u),\ u \in U$ pro jednotlivé stavy x_k v řádech $\sim 10^{-1}$. Hladové akce (s nejnižší Q-hodnotou) v daném stavu x_k tak nejsou z obr. 5.3 příliš patrné. Uvádíme proto změnu volby hladových



(a) : První běh **(b) :** Druhý běh

Obrázek 5.1: Regulace rychlosti ω_k pro Q-učení s agregací stavů.

akcí pro jednotlivé stavy na obr. 5.4. Opět je volba akcí přesná pouze v intervalu $(0; 1)$.

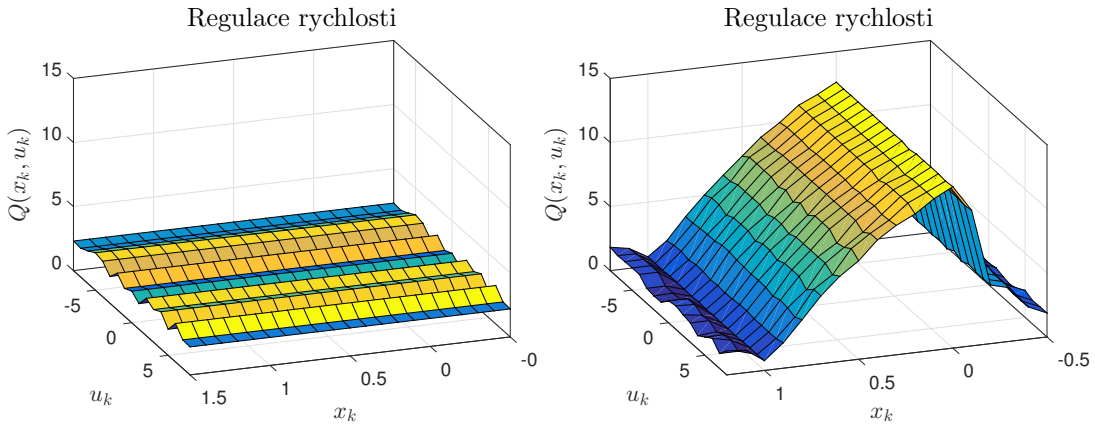


(a) : Agregace stavů **(b) :** Gaussovské RBF

Obrázek 5.2: Akční zásahy pro regulaci rychlosti při druhých bězích.

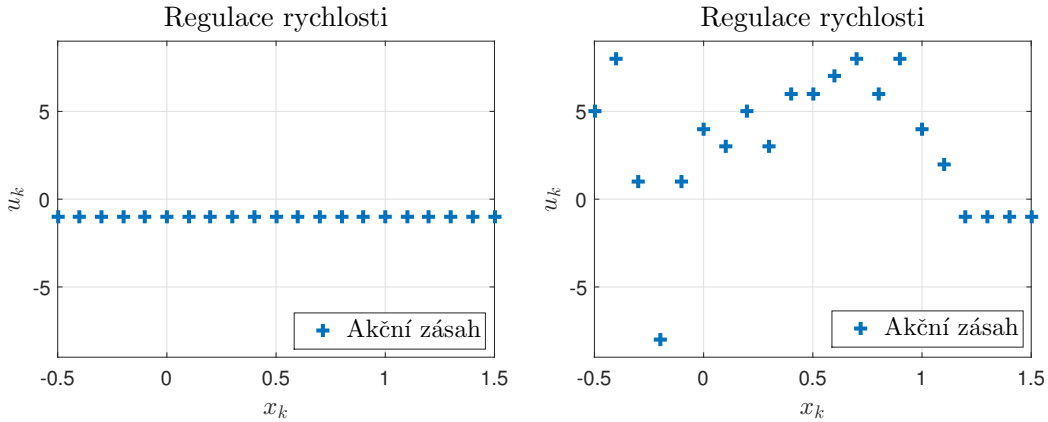
Gaussovské RBF

První běh regulace rychlosti pro RBF je ukázán na obr. 5.5a. Na rozdíl od použití agregace stavů je strategie nalezena již po $\sim 1 \times 10^4$ krocích. To je dané zejména lepší schopností RBF zobecnit již získané zkušenosti i na další stavy. Druhý běh je vyobrazen na obr. 5.5b. V tomto případě sice využití RBF vede k menší vzdálenosti od reference než u agregace stavů, nicméně je toho docíleno za pomocí střídání akcí $u = 3$ a $u = 7$, viz obr. 5.2b. Výstup ω_k proto kmitá okolo hodnoty $\omega = 1,01$.



(a) : Před prvním během (nenaučený) **(b) :** Po prvním běhu (naučený)

Obrázek 5.3: Q-hodnoty pro jednotlivé páry stav-akce pro Q-učení s agregací stavů.



(a) : Před prvním během (nenaučený) **(b) :** Po prvním běhu (naučený)

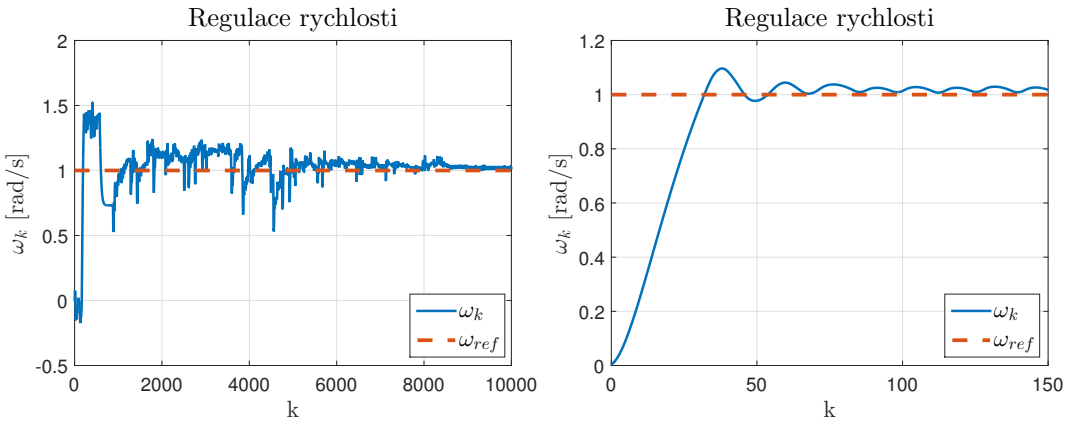
Obrázek 5.4: Hladové akce pro jednotlivé stavy pro Q-učení s agregací stavů pro regulaci rychlosti.

5.3.2 Potlačení poruchy

V experimentech k ověření schopnosti potlačit poruchy zavedenou do systému budeme již pracovat s Q-funkcí danou po prvním běhu regulace rychlosti k referenci. Jako poruchu volíme zatížení hřídele motoru, přesněji je hřídel motoru zatížena v intervalu $k \in \langle 200; 400 \rangle$ hodnotou $-0,1\,\mathrm{N\,m}$.

Agregace stavů

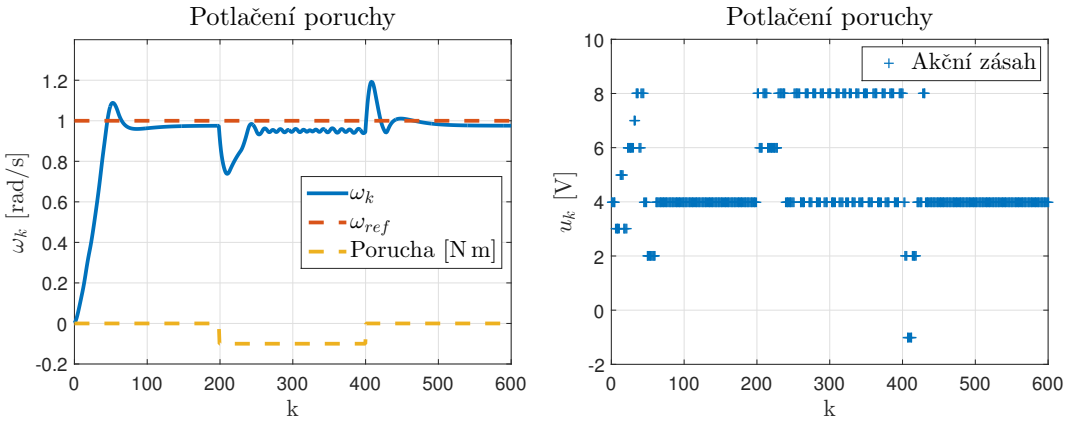
Na obr. 5.6 lze pozorovat schopnost agenta potlačit zavedenou poruchu do systému společně s akčním zásahem. Vlivem poruchy se systém dostává do stavu, v kterém



(a) : První běh **(b) :** Druhý běh

Obrázek 5.5: Regulace rychlosti ω_k pro Q-učení s Gaussovskými RBF.

je agent naučený (z předchozího běhu) volit akci $u = 8$. Díky této akci se opět dostává blíže k referenci. Při dosažení stavu blízko reference agent avšak opět volí akci $u = 4$. To opět vede ke snížení rychlosti ω_k. Z těchto důvodů pak rychlost ω_k osciluje přibližně kolem hodnoty $\sim 0,95$. Při skoku poruchy tak agent nedosahuje stejné odchylky od reference, jako v ustáleném stavu.



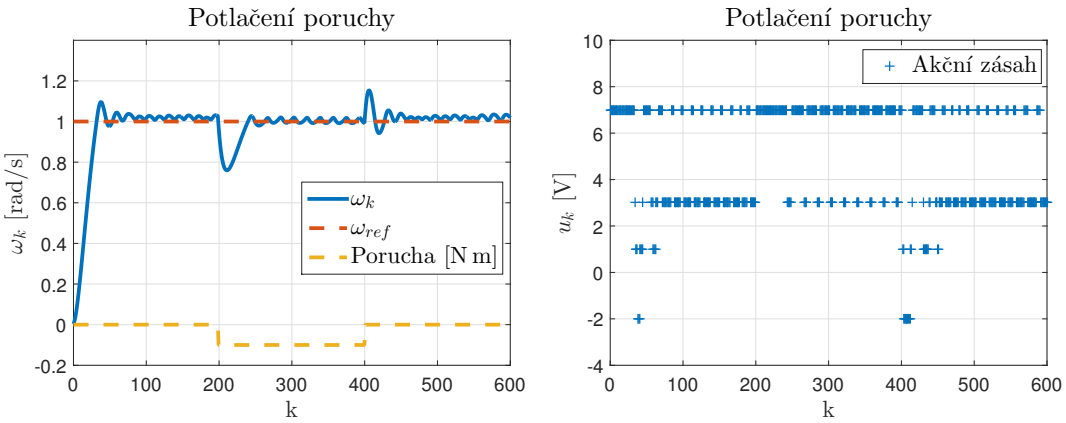
(a) : Výstup systému **(b) :** Akční zásah

Obrázek 5.6: Potlačení poruchy pro Q-učení s agregací stavů.

Gaussovské RBF

V případě použití Gaussovských RBF je situace podobná s použitím agregací stavů (výstup je při skoku poruchy opět kmitavý) s tím rozdílem, že je při poruše dosaženo menší odchylky od reference (rychlost kmitá okolo hodnoty $\omega = 1,01$). Výsledek

simulace je ukázán na obr. 5.7.



(a) : Výstup systému **(b) :** Akční zásah

Obrázek 5.7: Potlačení poruchy pro Q-učení s Gaussovskými RBF.

Kapitola 6

Závěr

Cílem této bakalářské práce bylo využít metody PU k návrhu zpětnovazebního řízení dynamického systému. K tomuto účelu jsme si nejprve uvedli základní teoretický rámec používaný pro teorii strojového učení, který jsme následně vztáhli na úlohu řízení dynamického systému. Dále jsme si představili základní metody založené na dynamickém programování a principu temporální diference.

K samotnému otestování jsme si vybrali Q-učení s parametrickou aproximací realizovanou jak za pomocí agregace stavů, tak i Gaussovských RBF. Tuto metodu jsme otestovali na řízení simulačního modelu DC motoru v programu MATLAB. Otestována byla regulace rychlosti hřídele motoru ω_k k referenční hodnotě a poté také schopnost potlačit zavedenou poruchu do systému. V obou případech se podařilo nalézt řešení, které ale nebylo optimální.

Ustálit výstup na zvolené referenci se nepodařilo zejména kvůli diskrétní množině akcí, která neobsahovala potřebnou hodnotu akce. Jako řešení by se nabízelo rozšířit množinu akcí U. Agent by avšak při učení musel volit z větší množiny akcí, což by mohlo zpomalit nalezení řešení. Další z možností by bylo využít aproximaci funkce i na strategii $h(.)$. Agent by pak mohl operovat se spojitým prostorem akcí bez ztráty rychlosti učení.

Reakce agenta na poruchu a důvod pro neoptimální potlačení poruchy byl již popsán v sekci 5.3.2. Aby se agent mohl lépe adaptovat na zavedenou poruchu, bylo by nutné měnit strategii během poruchy, tedy zvolit nenulový parametr α. Proces adaptace by mohl nicméně trvat déle než samotná porucha, což by nakonec mohlo vést k pozměnění správných Q-hodnot.

Ukazuje se tak, že elementární metody, které jsme si představili a implementovali, mají jisté problémy již s řízením jednoduchého lineárního systému. Pro složitější (nelineární systémy) by bylo vhodné využit pokročilejší metody PU pracující i se znalostí modelu systému popisované v literatuře [1], [5].

Příloha A

Literatura

[1] L. Buşoniu, D. Ernst, B. De Schutter, and R. Babuška, *Reinforcement learning and dynamic programming using function approximators*, Taylor & Francis CRC Press. ISBN 978-1-4398-2108-4.

[2] F. L. Lewis, D. Vrabie: *Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control*, IEEE Circuits and Systems Magazine, March 2009, s. 32-50.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning—An Introduction.* Cambridge, MA: MIT Press, 1998.

[4] A. Simonian, *Feedback control for planar parallel magnetic manipulator.* Master's thesis, Czech Technical University in Prague, May 2014, s. 31-50.

[5] R. Hafner, M. Riedmiller *Mach Learn* 2011 s. 84-137. doi:10.1007/s10994-011-5235-x,

[6] M. P. Deisenroth, C. E. Rasmussen, *Efficient reinforcement learning for motor control*, In 10th International PhD Workshop on Systems and Control, Hluboká nad Vltavou, Czech Republic, September 2009.

Příloha B

Terminologie a zkratky

Zde je uveden výčet důležité anglické terminologie a k nim příslušné použité české překlady a výčet zkratek:

B.1 Terminologie

- *Agent*: Agent
- *Curse of dimensionality*: Prokletí rozměrnosti
- *Dynamic programming*: Dynamické programování
- *Enviroment*: Prostředí
- *Exploration*: Explorace
- *Exploitation*: Exploatace
- *Gradient descent*: Gradientní sestup
- *Greedy action*: Hladová akce
- *Policy*: Strategie
- *Q-function*: ohodnocovací Q-funkce
- *Q-learning*: Q-učení
- *Reinforcement learning*: Posilované učení
- *Reward/Utility/Cost*: Cena
- *State agregation*: Agregace stavů
- *Temporal difference*: Temporální diference
- *Value function*: ohodnocovací V-funkce

B.2 Zkratky

- PU: Posilované učení
- MRP: Markovův rozhodovací proces
- DP: Dynamické programování
- TD: Temporální diference
- RBF: Radiální bázové funkce
- V-funkce / Q-funkce: ohodnocovací funkce $V(x)$ / $Q(x, u)$
- V-hodnota / Q-hodnota: hodnota ohodnocovací funkce $V(x)$ / $Q(x, u)$