UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA

DIPLOMOVÁ PRÁCE

Markovské rozhodovací procesy a jejich aplikace v energetice

Katedra matematické analýzy a aplikací matematiky
Vedoucí práce: **RNDr. Pavel Ludvík, Ph.D.**
Vypracoval(a): **bc. Marek Šuranský**
Studijní program: Aplikovaná matematika
Forma studia: prezenční
Rok odevzdání: 2024

BIBLIOGRAFICKÁ IDENTIFIKACE

Autor: bc. Marek Šuranský

Název práce: Markovské rozhodovací procesy a jejich aplikace v energetice

Typ práce: Diplomová práce

Pracoviště: Katedra matematické analýzy a aplikací matematiky

Vedoucí práce: RNDr. Pavel Ludvík, Ph.D.

Rok obhajoby práce: 2024

Abstrakt: Cílem práce je zavést pojem markovského rozhodovacího procesu a aplikovat tento koncept v praxi. V první kapitole začneme definicí těchto procesů a jejich vlastnostmi. Vysvětlíme pojem strategie, zaměříme se především na optimální strategie. Nakonec si v této kapitole připravíme teoretické základy nutné k zavedení algoritmů používaných k hledání optimální strategie. Tyto algoritmy budou pak přiblíženy v druhé kapitole, konkrétně se podíváme na algoritmy Value iteration, Policy iteration a Modified policy iteration, které také implementujeme v jazyce Python. Ve třetí kapitole se pak podíváme na konkrétní problém z oblasti energetiky, při jehož řešení se budeme snažit o snížení paměťové náročnosti našich algoritmů. Nakonec porovnáme výkon jednotlivých algoritmů při řešení našeho problému.

Klíčová slova: markovský rozhodovací proces, value iteration, policy iteration, optimalita, strategie, energetika

Počet stran: 56

Počet příloh: 1

Jazyk: český

BIBLIOGRAPHICAL IDENTIFICATION

Author: bc. Marek Šuranský

Title: Markov decision processes and their application in energy industry

Type of thesis: Master's

Department: Department of Mathematical Analysis and Application of Mathematics

Supervisor: RNDr. Pavel Ludvík, Ph.D.

The year of presentation: 2024

Abstract: The aim of this thesis is to define Markov decission processes and apply them in the field of energy industry. In the first chapter we will introduce this construkt and its propheties. Then we will focus on the concept of strategy with special focus on the optimal strategy. Last part of the chapter will be focused on theory needed for understanding of algoritms used to find the optimal strategy. We will intorduce those algoritms in the second chapter, we will focus on Value iteration, Policy iteration a Modified policy iteration, and we will implement them in Python. In the third chapter we will solve a real-life problem from the field of energy industry. We will pay close attention to space complexity of the used algoritms. Lastly we will compare how those algoritms performed while solving this problem.

Key words: Markov decision proces, Value iteration, Policy iteration, optimality, strategy, energy industry

Number of pages: 56

Number of appendices: 1

Language: Czech

Prohlášení

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně pod vedením pana RNDr. Pavel Ludvík, Ph.D. a všechny použité zdroje jsem uvedl v seznamu literatury.

V Olomouci dne ....

podpis

Obsah

Poděkování

Rád bych poděkoval mým rodičům jak za jejich podporu po celou moji cestu po akademickém prostředí, tak za to že mě ještě nevyhodili z domu. Dále bych chtěl poděkovat doktoru Pavlu Ludvíkovi za jeho pomoc, ochotu a vstřícnost. Abych předešel fyzickému násilí děkuji i svému bratrovi, i když nevím za co.

Úvod

V této diplomové práci se zaměříme na markovské rozhodovací procesy, jejichž koncept se datuje do padesátých let dvacátého století a jedná se o poměrně dobře probádanou oblast matematiky, přesto se zde najdou dosud nevyřešené problémy. Tyto procesy mají uplatnění v mnoha oblastech od ekonomie až po automatizaci nebo robotice. Jmenují se, stejně jako kráter na měsíci, po ruském matematikovi Andreji Markovovi (1856 - 1922), který se zabýval takzvanými Markovovými řetězci z nichž se tyto rozhodovací procesy postupně vyvinuly.

V této práci je nejdříve zavedeme, dokážeme si některé jejich vlastnosti. Poté si ukážeme algoritmy, které mají za cíl takovéto problémy řešit a implementujeme je v softwaru. Nakonec se pokusíme aplikovat tyto algoritmy na problém v oblasti energetiky, konkrétně se zaměříme na problém související s napětím v síti. Ještě o něco více než samotné řešení našeho problému nás budou zajímat algoritmy, které přitom použijeme.

Problémem, kvůli kterému se na tyto algoritmy chceme podívat je, že v blízké budoucnosti dojde k úpravám zákona. Tyto úpravy způsobí, že matice se kterými nyní pracujeme, se stanou jednoduše řečeno obrovskými a narazíme z tohoto důvodu problémi s hardwarem, především s velikostí pracovní paměti. Bude nás tedy kromě rychlosti konvergence algoritmu zajímat i jeho paměťová náročnost.

Věřím, že tyto procesy jsou i v dnešní době, téměř sto let od objevení, zajímavé, především díky jejich použití v robotice a oblasti umělé inteligence. Navíc se při jejich studii může člověk podívat do různých koutů matematiky, což je pro mě také velice zajímavé.

1. Úvod do MDP

Mějme proces, který dozoruje nějaký kontrolor. Kontrolor může do procesu zasáhnout v daných časech, ty nazveme *rozhodovací epochy*. Označme množinu rozhodovacích epoch jako T. Obecně platí $T \subset \mathbb{R}_0^+$. Pro potřeby této práce budeme předpokládat, že T je spočetná množina (ideálně konečná). Předpokládá se, že v každém $t \in T$ dojde k rozhodnutí, přičemž nedělat nic je také rozhodnutí.

Označme množinu všech *stavů*, v nichž se může proces nacházet jako S, množinu všech *akcí*, které může kontrolor zvolit ve stavu S jako A_S, a nakonec definujme množinu všech akcí $A = \bigcup_{s \in S} A_S$. I o množinách A a S budeme předpokládat, že jsou konečné, nebo alespoň spočetné.

Bude nás zajímat, jak se bude proces vyvíjí, tedy jak se mění stavy. Přechody mezi stavy jsou zatíženy náhodou, *pravděpodobnosti přechodu* z jednoho stavu do jiného jsou ovšem známy pro každou akci, dvojici stavů a rozhodovací epochu. Pravděpodobnost přechodu ze stavu $s \in S$ do stavu $j \in S$ za volby akce $a \in A$ v epoše $t \in T$ označme jako podmíněnou pravděpodobnost $p_t(j|s,a)$.

A nakonec se dostáváme k odměnám. Pro každý stav $s \in S$, každou akci $a \in A$ a rozhodovací epochu $t \in T$, definujeme $r_t(s,a)$ jako *očekávanou odměnu*, kterou rozhodovatel získá pro tuto kombinaci. Očekávanou odměnu $r_t(s,a)$ pak ještě můžeme rozepsat. Pro dané $j \in S$ označme jako $r_t(s,a,j)$ odměnu za to, že jsme ve stavu s zvolili akci a a skončili ve stavu j. Pak můžeme psát $r_t(s,a) = \sum_{j \in S} r_t(s,a,j) p_t(j|s,a)$, což odpovídá střední hodnotě diskrétní náhodné veličiny, která nabývá hodnoty $r_t(s,a,j)$ s pravděpodobností $p_t(j|s,a)$.

Definice 1. Uspořádanou pětici $(T, S, A_S, p_t(\cdot|s,a), r_t(s,a))$ prvků definova-

ných výše nazveme *markovský rozhodovací proces*, dále MDP (Markov Decision Process).

Poznámka 1.

- Pokud platí: $\forall t, t' \in T : p_t(j|s,a) = p_{t'}(j|s,a)$ a $\forall t, t' \in T : r_t(s,a) = r_{t'}(s,a)$, budeme vynechávat index t. Takovéto procesy se nazývají homogenní.

- Toto zavedení je postačující k praktickému použití, (podrobnější definici naleznete v apendixu 4). Ovšem není zřejmé, proč by měl být proces markovský, nebo přesněji, co to vůbec Markovova vlastnost je. Formální definici můžete najít zde 2. Neformálně lze říci, že v takovémto procesu nezáleží na celé historii, ale pouze na posledním stavu, ve kterém se proces nacházel. V naší definici jsme toho dosáhli tak, že pravděpodobnosti přechodu $p_t(\cdot|s,a)$ přijímá jako argument pouze poslední stav (a akci).

Definice 2. Mějme posloupnost náhodných veličin $\{X_n\}_{n\in\mathbb{N}_0}$ nabývající hodnoty z množiny stavů S. *Markovovou vlastností* této posloupnosti nazveme rovnost

$$P(X_{n+1} = s_j | X_n = s_i, X_n = s_{i-1}, \ldots, X_0 = s_0) = P(X_{n+1} = s_j | X_n = s_i) \tag{1}$$

pro každé $n = 0, 1, \ldots$ a pro všechny stavy $s_j, s_i, s_{i-1}, \ldots, s_0$ splňující $P(X_n = s_i, X_n = s_{i-1}, \ldots, X_0 = s_0) > 0$.

Příklad 1 (Štamgastova cesta domů)**.** Štamgast stojí po zavíračce před hospodou, může jít doprava, nebo doleva po chodníku, ten má 5 metrů. Pokud dojde na pravý okraj, spadne do příkopu, což ohodnotíme jako -100. Naopak na levém okraji se nachází jeho dům, dojde-li tam dostane odměnu $+1$.

Pokud dorazí do příkopu, nebo domů, tak tam usne. Na každém metru se může rozhodnout, jestli půjde doprava, nebo doleva, ovšem díky intoxikaci půjde vytouženým směrem jen s pravděpodobností $p = 0.7$.

Zapišme tuto úlohu v jazyku MDP. Nejprve $S = \{0, 1, 2, 3, 4, 5, 6\}$, kde názvy jednotlivých stavů říkají, jak daleko jsme od domova, stav 0 je domov a stav 6 je příkop. Máme 2 akce v každém stavu $\forall s \in S : A_s = \{doprava, doleva\}$. Pravděpodobnosti přechodu pak zavedeme nejprve pro libovolnou akci $a \in A$ a stavy $j \in \{0, 6\}$ jako $p(i|j, a) = \delta_{ij}$, pro zbylé stavy pak zadefinujeme $p(j-1|j, doprava) = 0.3, p(j+1|j, doprava) = 0.7, p(j-1|j, doleva) = 0.7, p(j+1|j, doleva) = 0.3$, všechny ostatní pravděpodobnosti položíme rovny nule. Jediné nenulové odměny pak budou $r(1, a, 0) = 1$ a $r(5, a, 6) = -100$ a tedy $r(1, doleva) = 0.7$, $r(1, doprava) = 0.3$, $r(5, doprava) = -70$, $r(5, doprava) = -30$ ostatní jsou nulové. ★

Díky tomu, že předpokládáme spočetnost množiny S, můžeme pro homogenní proces definovat matici pravděpodobností přechodu pro každou akci $a \in A$ jako $\mathbb{P}_a = (p(i|j, a))_{i,j \in S}$. Jedná se stochastickou matici, tedy všechny prvky jsou nezáporné a řádky se sčítají na jedničku. Podobně do matice zapíšeme i odměny pro každé $a \in A : \mathbb{R}_a = (r(s, a, j))_{i,j \in S}$.

Poznámka 2.

- Pro nehomogenní proces by přibyl ještě index t vyjadřující epochu. Ovšem většinou se zabýváme právě homogenními procesy, proto si dovolím tento index vynechat.
- Velkou výhodou tohoto přepisu do matic je zefektivnění výpočtů a jednodušší softwarová implementace.
- Tyto matice bývají pro větší problémy řídké, je vhodné s nimi tedy takto pracovat.

Příklad 2. Jak by tyto matice vypadaly pro problém v příkladu 1?
Začneme maticí odměn, protože v tomto případě bude stejná pro obě akce. Stejně jako ostatní 2 matice bude čtvercová a 7×7. Matice bude plná nul až na dvě výjimky, podmínka $r(1, a, 0) = 1$ říká, že zde má být jednička v 2. řádku v 1. sloupci a za $r(5, a, 6) = -100$ pak umístím -100 na 6. řádek do 7 sloupce.

$$R_a = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Co se týče matice pravděpodobností přechodu pro akci doprava, první a poslední řádek jsou jednoduché, v prvním řádku je jen jednička v 1. sloupci, v 7. řádku pak v 7. sloupci. Mezi nimi bude 0 na diagonále, tato pravděpodobnost by odpovídala tomu, že štamgast zůstane na stejné pozici, pak 0.7 na horní sekundární diagonále, protože tyto pozice odpovídají pohybu doprava, a 0.3 dolní sekundární diagonále.

$$P_{doprava} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Nakonec pro akci doleva, bude matice podobná, jen se prohodí pozice 0.3 a 0.7

$$R_{doleva} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.7 & 0 & 0.3 & 0 & 0 & 0 & 0 \\ 0 & 0.7 & 0 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0.7 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0.7 & 0 & 0.3 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

★

Otázka zní, jaké akce volitv daných stavech? K tomu slouží takzvaná *rozhodovací pravidla*, $d_t : S \to A$, neboli zobrazení, které přiřadí stavu s v čase t akci a. Tato pravidla uvažujeme v této práci deterministická, pokud nebude řečeno jinak, navíc musí pracovat pouze s posledním stavem, aby se zachovala markvskost. Pokud se jich budeme držet, můžeme napsat odměny a pravděpodobnosti přechodu jako $r_t(s, d_t(s))$, respektive $p_t(\cdot|s, d_t(s))$, a hlavně:

$$r_t(s, d_t(s)) = \sum_{j \in S} r_t(j, d_t(s)) p_t(j|s, d(s)) \tag{2}$$

Tato volba požadavků na rozhodovací pravidla je poměrně přísná. Označme markovská deterministická pravidla jako MD. Další skupinu takzvaných markovských nedeterministických pravidel MR. Tato pravidla pak udávají pravděpodobnostní rozdělení na množině akcí. Pokud bychom dovolili rozhodnutí libovolnou závislost na historii, tedy nebýt obecně markovské, získáme pro deterministické volby HD, a nedeterministické HR. Označme D_t^K, $K =$

MD, MR, HR, HD, množiny všech rozhodovacích pravidel daného typu v čase t .

Nyní, když již víme, jak budeme volit akce v jednotlivých časových okamžicích, se můžeme zamyslet, jak je budeme volit dlouhodobě. *Strategií* rozumíme posloupnost rozhodovacích pravidel, $\pi = (d_1, , d_2, \ldots, d_{n-1}), n \in \mathbb{N}, d_t \in D_t^K$, $K = \mathrm{MD}, \mathrm{MR}, \mathrm{HR}, \mathrm{HD}$. A množiny všech takovýchto strategií pak značíme jako Π^K, $K = \mathrm{MD}, \mathrm{MR}, \mathrm{HR}, \mathrm{HD}$.

Významnou skupinu pak tvoří *stacionární strategie*, tyto strategie nejsou závislé na čase, platí totiž pro každé $t \in T : d_t = d$, tedy $\pi = (d, d, \ldots)$. Tyto strategie budeme značit $d^{+\infty}$. Množinu všech stacionárních deterministických, respektive nedeterministických, strategií budeme značit Π^{SD}, respektive Π^{SR}, Stacionární deterministické strategie se někdy označují jako ryzí. Právě takovéto strategie budou hrát zásadní roly později, až se dostaneme k MDP s nekonečným počtem kroků.

Poznámka 3. Povšimněme si, že při stacionární strategii nejsou rozhodovací pravidla funkcí času a nejsou tedy závislá na historii, každá stacionární strategie je tedy markovská.

Poznámka 4. Platí inkluze $\Pi^{\mathrm{SD}} \subset \Pi^{\mathrm{SR}} \subset \Pi^{\mathrm{MR}} \subset \Pi^{\mathrm{HR}}$, $\Pi^{\mathrm{SD}} \subset \Pi^{\mathrm{MD}} \subset \Pi^{\mathrm{MR}} \subset \Pi^{\mathrm{HR}}$ a $\Pi^{\mathrm{SD}} \subset \Pi^{\mathrm{MD}} \subset \Pi^{\mathrm{MD}} \subset \Pi^{\mathrm{HR}}$. Z poznámky 3 víme, že každá stacionární strategie je markovská, $\Pi^{\mathrm{SR}} \subset \Pi^{\mathrm{MR}}$ a $\Pi^{\mathrm{SD}} \subset \Pi^{\mathrm{MD}}$. Pokud povolíme závislost na historii v libovolném tvaru ,HD, HR, a za libovolný tvar zvolím závislost pouze na posledním stavu, získám markovskost, tedy $\Pi^{\mathrm{MR}} \subset \Pi^{\mathrm{HR}}$ a $\Pi^{MD} \subset \Pi^{\mathrm{HD}}$. Pokud definujeme rozdělení pravděpodobnosti jako $P(x) = 1$ pro dané x a nula jinak, získáme deterministickou volbu a tedy $\Pi^{\mathrm{HD}} \subset \Pi^{\mathrm{HR}}$, $\Pi^{MD} \subset \Pi^{\mathrm{MR}}$, a $\Pi^{\mathrm{SD}} \subset \Pi^{\mathrm{SR}}$.

1.1. Jednokrokový proces

Nyní již menší ochutnávka toho, co bude naším cílem v této práci a to je volba vhodné strategie. Zatím se podíváme pouze na proces s $T = 1, 2$, tedy proces, u kterého proběhne jen jeden krok. Navíc budeme předpokládat konečnost množin A a S.

Celková odměna se bude skládat z okamžité odměny $r_1(s, a)$ a koncové odměny $v : S \to \mathbb{R}$. Ta v podstatě přiřadí každému stavu, ve kterém skutečně skončíme, nějakou odměnu, například později to bude očekávaná odměna procesu, který započne v tomto stavu. Protože neznáme výsledný stav s jistotou, budeme i tady uvažovat střední hodnotu všech možných výsledků. Označíme-li náhodnou proměnnou, která odpovídá koncovému stavu jako X_2, dostaneme:

$$r_1(s, a) + E_s^a(v(X_2))) = r_1(s, a) + \sum_{j \in S} p_1(j|s, a)v(j). \tag{3}$$

My bychom chtěli zvolit takovou strategii $\pi = (d_1)$, která zvolí akci $a^* \in A_s$, pro kterou bude tato hodnota maximální:

$$r_1(s, a^*) + \sum_{j \in S} p_1(j|s, a^*)v(j) = \max_{a \in A_s}\{r_1(s, a) + \sum_{j \in S} p_1(j|s, a)v(j)\}. \tag{4}$$

Pro zjednodušení zavedeme notaci:

$$a^* \in \arg\max_{a \in A_s}\{r_1(s, a) + \sum_{j \in S} p_1(j|s, a)v(j)\}. \tag{5}$$

Poznámka 5. Zatímco $\max\{\cdot\}$ vrací číslo konkrétně hodnotu maxima, $\arg\max\{\cdot\}$ vrací množinu všech čísel maximalizujících výraz v závorce.

Protože předpokládáme konečnost množiny A, musí být množina A_s také

konečná. Maximum tedy existuje, protože hledáme maximum přes prvky konečné množiny. Nemusí být ovšem dáno jednoznačně.

Řešení takovýchto problémů, s danou funkcí $v(\cdot)$, bude naprosto stěžejní pro používání algoritmů, které budou uvedeny později v této práci.

1.2. Nekonečné procesy

V této kapitole se začneme zabývat případem kdy $T = \mathbb{N}_0$. Konkrétně si připravíme pozici pro samotné algoritmy. Projdeme optimalizační kritéria a zavedeme pojmy, potřebné pro práci s těmito procesy.

Nekonečnost našeho procesu vede, mimi jiné, k problémům s definicí odměn. Dá se předpokládat, že se neobejdeme bez pojmu limita, a také to tak bude. Zatím jsme zavedli odměny, které získáme za jeden krok, ovšem víc by nás zajímala celková odměna za celý proces, tedy jejich součet přes všechny epochy. To je ovšem v tomto případě nekonečná suma. Zpočátku budeme chtít její bodovou konvergenci pro každý stav $s \in S$. Později zavedeme i konvergenci ve vhodné normě.

První otázkou bude, jakou odměnu můžeme očekávat od dané strategie. Máme-li strategii $\pi = (d_1, d_2, \dots)$, můžeme náš proces přepsat jako $\{(X_t, r_t(X_t, Y_t))\}_{t\in\mathbb{N}}$, kde X_t je stav procesu v čase t, $Y_t = d_t(X_t)$ je zvolená akce a $r_t(\cdot,\cdot)$ je očekávaná odměna. Pro markovskou strategii se jedná o markovský řetězec s odměnami, který je také vysvětlen v apendixu 4).

Nyní již zavedeme několik základních pojmů. Tyto pojmy zavedeme pro obecnější skupinu strategií $\pi \in \Pi^{\mathrm{HR}}$. Ve všech těchto definicích, tedy definicích 3, 4 a 5, bude vystupovat stav $s \in S$, který bude značit stav, ve kterém proces započal:

Definice 3. *Očekávanou celkovou odměnou* strategie $\pi \in \Pi^{\mathrm{HR}}$ definujeme

jako:

$$v^{\pi}(s) = \lim_{n \to +\infty} E_s^{\pi}\{\sum_{t=1}^{n} r(X_t, Y_t)\} = \lim_{n \to +\infty} v_{n+1}^{\pi}(s) \tag{6}$$

kde $v_{n+1}^{\pi}(s)$ je očekávaná celková odměna téže strategie za $n+1$ kroků s nulovou koncovou hodnotou:

$$v_{n+1}^{\pi}(s) = E_s^{\pi}\{\sum_{t=1}^{n} r(X_t, Y_t)\} \tag{7}$$

Může se samozřejmě stát, že tato posloupnost diverguje k $+\infty$, $-\infty$, nebo limita dokonce ani nemusí existovat. Pokud je to možné, například pomocí Lebesgueovy věty, prohazuje se limita a střední hodnota:

$$v^{\pi}(s) = E_s^{\pi}\{\sum_{t=1}^{+\infty} r(X_t, Y_t)\} \tag{8}$$

Odměny v budoucnosti pro nás často nemají tak velkou cenu jako ty současné, toto chování nám pomůže zachytit tzv. *diskontovaná odměna*, kdy ponecháme hodnotu odměny v prvním kroku, ale ostatní pak zmenšujume pomocí mocnění faktoru $\lambda \in [0, 1)$

Definice 4. *Diskontovanou odměnu* strategie $\pi \in \Pi^{\mathrm{HR}}$ s *diskontním faktorem* $\lambda \in [0, 1)$ definujeme jako:

$$v_{\lambda}^{\pi}(s) = \lim_{n \to +\infty} E_s^{\pi}\{\sum_{t=1}^{n} \lambda^{t-1} r(X_t, Y_t)\} \tag{9}$$

I tato suma nám může divergovat, podívejme se na postačující podmínku její konvergence:

Lemma 1. *Nechť existuje $M \in \mathbb{R}$ splňující:*

$$\sup_{s\in S}\sup_{a\in A}\{|r(s,a)|\} \leqq M, \tag{10}$$

potom limita $\lim_{n\to+\infty} E_s^\pi\{\sum_{t=1}^n \lambda^{t-1} r(X_t, Y_t)\}$ *je vlastní pro* $\lambda \in [0,1)$.

Důkaz. Platí:

$$\lim_{n\to+\infty} |E_s^\pi\{\sum_{t=1}^{n} \lambda^{t-1} r(X_t,Y_t)\}| \leqq \lim_{n\to+\infty} E_s^\pi\{\sum_{t=1}^{n} \lambda^{t-1}|r(X_t,Y_t)|\} \leqq \tag{11}$$

$$\leqq \lim_{n\to+\infty} E_s^\pi\{\sum_{t=1}^{n} \lambda^{t-1} M\} = \sum_{t=1}^{+\infty} \lambda^{t-1} M = M\sum_{t=1}^{+\infty}\lambda^{t-1} < +\infty \tag{12}$$

První nerovnost získáme z trojúhelníkové nerovnosti, druhou z předpokladu, pak využijeme toho, že střední hodnota reálného čísla je právě toto číslo a odstraníme ji. Z výpočtu plyne také existence limity. ■

Nakonec by nás zajímalo, jak rychle se naše odměna mění:

Definice 5. *Průměrným ziskem strategie* $\pi \in \Pi^{\mathrm{HR}}$ je

$$g^\pi(s) = \lim_{n\to+\infty} \frac{1}{n} E_s^\pi\{\sum_{t=1}^{n} r(X_t,Y_t)\} = \lim_{n\to+\infty} \frac{1}{n} v_{n+1}^\pi(s). \tag{13}$$

pokud tato limita neexistuje, zavádí se dvojice pojmů limes inferior průměrného zisku a limes superior průměrného zisku:

$$g_-^\pi(s) = \liminf_{n\to+\infty} \frac{1}{n} v_{n+1}^\pi(s), \text{ respektive } g_+^\pi(s) = \limsup_{n\to+\infty} \frac{1}{n} v_{n+1}^\pi(s). \tag{14}$$

1.3. Kritéria optimality

Nyní již k možným kritériím, které budeme sledovat při volbě ideální strategie.

Maximalizace očekávané celkové odměny

Toto kritérium nejspíše nikoho nepřekvapí. Jak napovídá nadpis, budeme hledat takovou strategii π maximalizující

$$v^{\pi}(s) = \lim_{n \to +\infty} v^{\pi}_{n+1}(s). \tag{15}$$

Problém tohoto kritéria je právě existence této limity, případně její konečnost. Pokud k takovémuto problému dojde pro jedno nebo více kritérií, můžeme provést nějakou restrikci, nebo zvolit jiné kritérium.

Podívejme se, jaké modely ovšem pomocí tohoto kritéria zkoumat můžeme. Definujme:

$$v^{\pi}(s)_{+} = E^{\pi}_{s}\{\sum_{t=1}^{+\infty} r^{+}(X_t, Y_t)\} \tag{16}$$

a

$$v^{\pi}(s)_{-} = E^{\pi}_{s}\{\sum_{t=1}^{+\infty} r^{-}(X_t, Y_t)\}, \tag{17}$$

kde $r^{-}(X_t, Y_t) = \max\{-r(s,a), 0\}$ a $r^{+}(X_t, Y_t) = \max\{r(s,a), 0\}$. Jsou to tedy sumy kladných sčítanců, budou tedy existovat. Přidáme-li k tomuto předpoklad, že pro každou strategii π je buď $v^{\pi}(s)_{+}$, nebo $v^{\pi}(s)_{-}$ konečná, bude limita $v^{\pi}(s) = \lim_{n \to +\infty} v^{\pi}_{n+1}(s)$ existovat, navíc platí:

$$v^{\pi}(s) = \lim_{n \to +\infty} v^{\pi}_{n+1}(s) = v^{\pi}(s)_{+} - v^{\pi}(s)_{-}. \tag{18}$$

Uveďme zajímavé skupiny takovýchto modelů:

- *Nezáporné omezené modely*: $\forall s \in S \exists a \in A_s : r(s,a) \geqq 0$ a $v^{\pi}(s)_{+}$ je konečná pro všechny přípustné strategie.

- *Nekladné modely*: Existuje strategie π taková, že $\forall s \in S, a \in A$: $r(s,a) \leqq 0$ a pro všechny strategie $v^\pi(s) > -\infty$.

- *Konvergentní modely*: Pro každý stav a každou strategii $v^\pi(s)_+$ a $v^\pi(s)_-$ jsou konečné.

Problémy modelované nezápornými omezenými modely jsou typicky takové, ve kterých se snažíme vyhnout nějakému konečnému stavu. Například hrajeme na výherních automatech, můžeme hrát, dokud nám nedojdou peníze, snažíme se tedy hrát tak, abychom se nedostali do stavu „Už mi nic nezůstalo“. Nekladným modelům připadá opačná situace, chci se co nejrychleji dostat do nějaké množiny a tam zůstat.

Pro tyto modely nazveme strategii maximalizující celkovou očekávanou odměnu *optimální pro celkovou odměnu*. Takováto strategie $\pi^* \in \Pi^{\mathrm{MD}}$ splňuje:

$$\forall \pi \in \Pi^{\mathrm{MD}}, \forall s \in S : v^{\pi^*}(s) \geqq v^\pi(s). \tag{19}$$

Hodnotou daného MDP rozumíme

$$v^*(s) = \sup_{\pi \in \Pi^{\mathrm{MD}}} v^\pi(s). \tag{20}$$

Strategie optimální pro celkovou odměnu tedy splňuje:

$$\forall s \in S : v^*(s) = v^{\pi^*}(s). \tag{21}$$

Existence strategie splňující tyto rovnost je pak postačující podmínkou existence strategie optimální pro celkovou odměnu.

Maximalizace diskontované očekávané celkové odměny

I význam tohoto kritéria je zřejmý. Cílem této kapitoli ovšem bude ukázat jiný pohled na tuto charakteristiku, pro připomenutí:

$$v_\lambda^\pi(s) = \lim_{n \to +\infty} E_s^\pi \{ \sum_{t=1}^{n} \lambda^{t-1} r(X_t, Y_t) \}. \tag{22}$$

Mějme náhodný markovský proces s náhodnou délkou ν nezávislou na volbě akcí. Definujeme očekávanou celkovou odměnu pro strategii π pro takovýto proces:

$$v_\nu^\pi(s) = E_s^\pi [E_\nu \{ \sum_{t=1}^{\nu} r(X_t, Y_t) \}]. \tag{23}$$

Zvolme nyní vhodně pravděpodobnostní distribuci ν, konkrétně zvolíme geometrické rozdělení s parametrem λ, $Ge(\lambda)$:

$$P(v = n) = \lambda^{n-1}(1 - \lambda). \tag{24}$$

Lemma 2. *Nechť platí předpoklady lemmatu 1 a $\nu \sim Ge(\lambda)$, pak*

$$v_\nu^\pi(s) = v_\lambda^\pi(s) \tag{25}$$

Důkaz. Nejdříve si rozepíšeme střední hodnotu E_ν jako $E_s^\pi [E_\nu \{ \sum_{t=1}^{\nu} r(X_t, Y_t) \}] = E_s^\pi \sum_{n=1}^{+\infty} [\{ \sum_{t=1}^{n} r(X_t, Y_t) \lambda^{n-1}(1 - \lambda) \}]$.

Z předpokladů, těch z lemmatu 1, je tato suma absolutně konvergentní. Můžeme použít Fubiniho větu a prohodit sumy $E_s^\pi \sum_{t=1}^{+\infty} [\sum_{n=t}^{+\infty} r(X_t, Y_t) \lambda^{n-1}(1 - \lambda)\}]$.

Provedeme několik úprav

$$(1-\lambda)\sum_{n=t}^{+\infty}\lambda^{n-1} = (1-\lambda)(\sum_{n=1}^{+\infty}\lambda^{n-1} - \sum_{n=1}^{t-1}\lambda^{n-1}) \tag{26}$$

$$= 1 - \sum_{n=1}^{t-1}(\lambda^{n-1} - \lambda^n) = 1 - 1 + \lambda^{t-1} = \lambda^{t-1}, \tag{27}$$

kde jsme použili sčítací vzorec $\sum_{n=1}^{+\infty}\lambda^{n-1} = \frac{1}{(1-\lambda)}$ a fakt, že všechny ostatní členy se odečetly mezi sebou.

Dosadíme a získáme

$$v_\nu^\pi(s) = E_s^\pi\{\sum_{t=1}^{n}\lambda^{t-1}r(X_t, Y_t)\} = v_\lambda^\pi(s).$$

■

To v podstatě odpovídá procesu s absorbčními stavy. Proces se nějak chová, než se v čase t dostane do některého z těchto stavů, a tam už zůstane. Takovými stavy byly třeba domov nebo příkop v příkladu se štamgastem.

Strategii nazveme *diskont-optimální* pro dané $\lambda \in [0, 1)$, pokud

$$\forall\pi \in \Pi^{\mathrm{MD}}, s \in S : v_\lambda^{\pi^*}(s) \geqq v_\lambda^\pi(s). \tag{28}$$

Pro modely s diskontním faktorem hodnotou daného MDP rozumíme

$$v_\lambda^*(s) = \sup_{\pi\in\Pi^{\mathrm{MD}}} v_\lambda^\pi(s). \tag{29}$$

Diskont-optimální strategie tedy splňuje:

$$\forall s \in S : v_\lambda^*(s) = v_\lambda^{\pi^*}(s), \tag{30}$$

Existence strategie splňující tyto rovnosti je opět postačující podmínkou existence diskont-optimální strategie.

Pokud bude zjevné, že se jedná o diskontovaný MDP, budeme pro jednoduchost nazývat tyto strategie jednoduše *optimální*.

Ne všechny algoritmy nám ale dovolí této strategie s jistotou dosáhnout, zavádí se tedy slabší pojem ϵ-*optimální strategie*. To je taková strategie π^*_ϵ splňující pro každý stav $s \in S$:

$$v_\lambda^{\pi^*_\epsilon}(s) \geqq v_\lambda^*(s) - \epsilon. \tag{31}$$

Další kritéria optimality

Nejdříve stručně zformulujeme strategie optimalizující zisk. Pokud existuje $g(s)$ pro všechny stavy, pak strategií s optimálním ziskem rozumíme strategii π^* takovou, že

$$\forall \pi \in \Pi^{\mathrm{MD}}, s \in S : g^{\pi^*}(s) \geqq g^\pi(s). \tag{32}$$

Optimálním ziskem g^* pak rozumíme

$$g^*(s) = \sup_{\pi \in \Pi^{\mathrm{MD}}} v^\pi(s). \tag{33}$$

Jako v minulých případech, i tady existují rovnosti, které slouží i jako postačující podmínka existence:

$$\forall s \in S : g^*(s) = g^{\pi^*}(s). \tag{34}$$

Pokud by $g^\pi(s)$ pro některé strategie neexistovalo, můžeme využít slabší podmínky (14) v definici 5. Strategii π^* nazveme *průměrně optimální* pokud

pro každý stav $s \in S$ a každou strategii $\pi \in \Pi^{\mathrm{MD}}$ platí

$$g_{-}^{\pi^*}(s) = \liminf_{n\to+\infty} \frac{1}{n} v_{n+1}^{\pi}(s) \geqq \limsup_{n\to+\infty} \frac{1}{n} v_{n+1}^{\pi}(s) = g_{+}^{\pi}(s) \tag{35}$$

Případně ještě slabší kritéria $g_{-}^{\pi^*}(s) \geqq g_{-}^{\pi}(s)$ a $g_{+}^{\pi^*}(s) \geqq g_{+}^{\pi}(s)$. Takovéto π^* pak nazveme *liminf průměrně optimální*, respektive *limsup průměrně optimální*.

Asymptotická optimalita

Co když nevíme, kolik v_n^{π} je, nebo jestli existuje? Přesto bychom chtěli srovnávat strategie mezi sebou? Řekneme, že strategie π^* *asymptoticky dominuje* strategii π, pokud pro každý stav $s \in S$ a každou akci $a \in A$ platí

$$\liminf_{n\to+\infty} [v_n^{\pi^*}(s) - v_n^{\pi}(s)] \geqq 0 \tag{36}$$

Ne všechny dvojice strategií je ovšem možno touto statistikou rozlišit. Definujeme tedy ještě jedno kritérium, které je o něco jemnější. V tomto případě řekneme, že π^* *v průměru asymptoticky dominuje strategii* π, pokud platí pro každý stav a každou strategii

$$\liminf_{n\to+\infty} \frac{1}{n} \sum_{t=1}^{n} [v_t^{\pi^*}(s) - v_t^{\pi}(s)] \geqq 0. \tag{37}$$

Pokud i toto selže, můžeme se pokusit použít ještě:

$$\liminf_{n\to+\infty} \sum_{t=1}^{n} [v_t^{\pi^*}(s) - v_t^{\pi}(s)] \geqq 0. \tag{38}$$

Citlivá kritéria pro diskontovaný MDP

Pro diskontovaný MDP se používá skupina kritérií založená na limitním chování daných strategií pro λ jdoucí k 1 zleva. Konkrétně pro $n = -1, 0, 1, 2, \ldots$ řekneme, že π^* je *n-diskont-optimální*, pokud pro každý stav $s \in S$ a strategii $\pi \in \Pi^{\mathrm{MD}}$:

$$\liminf_{\lambda \to 1^-} (1-\lambda)^{-n} [v_\lambda^{\pi^*}(s) - v_\lambda^{\pi}(s)] \geqq 0. \tag{39}$$

Pokud je π^* n-diskont-optimální pro každé $n = -1, 0, 1, 2, \ldots$ značíme ji jako *∞-diskont-optimální*. A *1-optimální*, nebo také *blackwell optimální*, je strategie π^*, pokud existuje $\lambda^*(s)$ taková, že

$$[v_\lambda^{\pi^*}(s) - v_\lambda^{\pi}(s)] \geqq 0 \tag{40}$$

pro všechny přípustné strategie π a pro všechna λ taková, že $\lambda^*(s) \leqq \lambda < 1$.

Označme $\lambda^* = \sup_{s \in S}\{\lambda^*(s)\}$. Pro nekonečnou spočetnou množinu S může dojít k tomu, že $\lambda^* = 1$. Pokud $\lambda^* < 1$, nazveme strategii *silně blackwellovsky optimální*.

Platí, že pro $n = -1$ dostaneme obdobu kritéria optimálního zisku a že blackwellovsky optimální strategije jsou právě ∞-optimální. Navíc 0-optimalitu budeme nazývat *biased optimalitou*. Je zjevné, že n-optimalita implikuje m-optimalitu pro $n > m$.

1.4. Vektorová notace

V této kapitole si nejdříve zavedeme prostor, ve kterém se budeme pohybovat a později si ještě zopakujeme vektorovou notaci našeho problému, která bude dále často využívána.

Mějme vektorový prostor reálných omezených funkcí na S, který označíme jako V. Na této množině zavedeme částečné upořádání po komponentech, značíme $\leqq$. Pro nespočetnou S bychom museli ještě zavést V_M jako prostor borelovsky měřitelných funkcí z V. Pro nejvýše spočetné platí $V = V_M$. Na této množině pak zavedeme supremovou normu $\|v\| = \sup_{s\in S} |v(s)|$. Pro matici $M \in \mathbb{R}^n$, $M = (m_{ij})_{i,j=1}^n$, definujeme její normu jako: $\|M\| = \sup_{i\in\{1,2,\dots n\}} \sum_{j=1}^n m_{ij}$. Dále ještě označme $e \in V$ jako funkci nabývající hodnoty jedna v každém stavu z S, tedy $\forall s \in S : e(s) = 1$.

Mějme libovolné rozhodovací pravidlo $d \in d^{\mathrm{MD}}$ a libovolné stavy j a s. Pak definujeme $r_d(s) = r(s, d(s))$ a $p_d(j|s) = p(j|s, d(s))$. Tyto hodnoty poskládáme k sobě jako vektor $r_d = (r_d(s)_{s\in S})$, a matici $P_d = (p_d(j|s))_{j,s\in S}$. Očekávanou odměnu za jeden krok pak můžeme získat jako $r_d + \lambda P_d v$ pro $0 \leqq \lambda \leqq 1$. Bylo by dobré, kdybychom takto nemohli opustit prostor V, což si nyní dokážeme:

Lemma 3. *Nechť S je diskrétní množina, pak pro omezené odměny, tedy takové splňující pro libovolný stav $s \in S$ a pro každou přípustnou akci $a \in A_s$ podmínku $|r(s,a)| < M < +\infty$, $0 \leqq \lambda \leqq 1$ a rozhodovací pravidlo $d \in D^{\mathrm{MD}}$ platí, že pro každé $v \in V$ je $r_d + \lambda P_d v$ je prvkem V.*

Důkaz. Pro důkaz tohoto tvrzení nám v podstatě stačí ukázat omezenost funkce $r_d + \lambda P_d v$, první člen můžeme jednoduše omezit z předpokladů $r_d(s) < M$, tedy $r_d \in V$. Druhý člen pak z $\|P\| = 1$ můžeme omezit normou v, která je konečná protože $v \in V$, tedy $\lambda P_d v \in V$. ■

Matice pravděpodobností přechodu pro více než jeden krok, pokud bychom se drželi markovské deterministické strategie $\pi = (d_1, \dots)$, by pak byla rovna součinu příslušných matic, tedy pro n kroků: $P_{d_1} P_{d_2} \dots P_{d_n}$ tento součin označíme jako P_π^n.

Očekávanou diskontovanou odměnu strategie π tedy můžeme zapsat pomocí vzorce:

$$v_\lambda^\pi = \sum_{t=1}^{+\infty} \lambda^{t-1} P_\pi^{t-1} r_d \tag{41}$$

1.5. Rekurze a Bellmanovy rovnice

V této kapitole se již seznámíme s principy, na základě kterých budou fungovat samotné algoritmy. V celé kapitole budeme předpokládat následujíci čtyři podmínky:

- Budeme předpokládat stacionaritu odměn a pravděpodobností přechodu,
- omezenost odměn, tedy existuje $M > 0$ takové, že $|r(s,a)| \leqq M < +\infty$ pro všechna $s \in S, a \in A_S$,
- vždy diskontujeme odměny faktorem $0 \leqq \lambda < 1$,
- S je nejvýše spočetná množina.

Začneme tím, že si ukážeme, jak přepsat očekávanou discountovanou odměnu jako rekurzi. Právě tento tvar se nám bude později hodit při hledání optimálních strategií.

Pro danou strategii $\pi \in \Pi^{\mathrm{MR}}$ a přípustné λ jsme definovali očekávanou odměnu jako

$$v_\lambda^\pi(s) = E_s^\pi \{ \sum_{t=1}^{+\infty} \lambda^{t-1} r(X_t, Y_t) \}. \tag{42}$$

To nejdříve přepíšeme do vektorové notace, ve které položíme P_0 rovnu

jednotkové matici, a poté upravíme pomocí (41):

$$\begin{aligned} v_\lambda^\pi &= \sum_{t=1}^{+\infty} \lambda^{t-1} P_\pi^{t-1} r_d \\ &= r_{d_1} + \lambda P_{d_1} r_{d_2} + \lambda^2 P_{d_1} P_{d_2} r_{d_3} + \dots \\ &= r_{d_1} + \lambda P_{d_1} (r_{d_2} + \lambda^2 P_{d_2} r_{d_3} + \dots) \\ &= r_{d_1} + \lambda P_{d_1} v_\lambda^{\pi'} \end{aligned}$$

Pro stacionární strategii $d^\infty = (d, d, \dots)$ se problém zjednoduší na

$$v_\lambda^{d^\infty} = r_d + \lambda P_d v_\lambda^{d^\infty}. \tag{43}$$

Tyto rovnosti říkají, že očekávanou diskontovanou odměnu jedné strategie získáme jako okamžitou odměnu plus lambda násobek váženého součtu očekávaných diskauntovaných odměn, kde váhy jsou pravděpodobnosti přechodu. Pro ještě lepší náhled se podívejme na jeden řádek této soustavy:

$$v_\lambda^\pi(s) = r_{d_1}(s) + \sum_{j \in S} \lambda p_{d_1}(j|s) v_\lambda^{\pi'}(j). \tag{44}$$

Pro samotné zkoumání MDP zavedeme lineární operátor $L_d : V \to V$ jako

$$L_d v = r_d + \lambda P_d v \tag{45}$$

Rovnici (43) pak můžeme napsat ve tvaru:

$$v_\lambda^{d^\infty} = L_d v_\lambda^{d^\infty}. \tag{46}$$

Což vede na důležité pozorování, že $v_\lambda^{d^\infty}$ je *pevným bodem* operátoru L,

což později využijeme.

Podívejme se ještě na dvojici vět, které se nám později budou hodit:

Věta 4. *Nechť* $0 \leqq \lambda < 1$. *Pak pro libovolnou stacionární strategii* $d^\infty, d \in D^{\mathrm{MD}}$, *je* $v_\lambda^{d^\infty}$ *jediné řešení rovnice*

$$L_d v = r_d + \lambda P_d v = v \tag{47}$$

a navíc můžeme psát

$$v_\lambda^{d^\infty} = (I - \lambda P_d)^{-1} r_d. \tag{48}$$

Důkaz. Jednoduchými úpravami získáme z (47):

$$(I - \lambda P_d) v = r_d. \tag{49}$$

Pak použitím poznatků z lineární algebry získáme $\sigma(\lambda P_d) \leqq \|\lambda P_d\| \leqq \lambda < 1$ a tedy existuje inverze k $(I - \lambda P_d)$, kterou umíme vyjádřit jako $(I - \lambda P_d)^{-1} = \sum_{t=1}^{+\infty} \lambda^{t-1} P_d^{t-1}$. Vynásobíme rovnici (49) touto inverzí zleva. Rozepíšeme inverzi jako sumu a nakonec z (41)

$$v = (I - \lambda P_d)^{-1} r_d = \sum_{t=1}^{+\infty} \lambda^{t-1} P_d^{t-1} r_d = v_\lambda^{d^\infty}. \tag{50}$$

■

Věta 5. *Nechť* $0 \leqq \lambda < 1$ *a* $u, v \in V$. *Pak pro libovolnou stacionární strategii* d^∞ *platí:*

- *pokud platí* $u \geqq 0$, *pak* $(I - \lambda P_d)^{-1} u \geqq 0$ *a* $(I - \lambda P_d)^{-1} u \geqq u$,
- *dále pokud* $u \geqq v$, *pak* $(I - \lambda P_d)^{-1} u \geqq (I - \lambda P_d)^{-1} v$, *a nakonec*

- *je-li* $u \geqq 0$, *platí* $u^T(I - \lambda P_d)^{-1} \geqq 0$ *a* $u^T(I - \lambda P_d)^{-1} \geqq u^T$.

Důkaz. Stejně jako v minulém příkladě můžeme díky tomu, že $\sigma(\lambda P_d) < 1$, rozvinout $(I - \lambda P_d)^{-1}$ do řady a poté využít faktu, že všechny prvky matice P_d jsou nezáporné, tedy všechny prvky, které vynecháme, jsou také nezáporné vektory, poté dostaneme například pro první případ:

$$(I - \lambda P_d)^{-1}u = u + \lambda P_d u + \lambda^2 P_d^2 u + \cdots \geqq u \geqq 0. \tag{51}$$

Zbylá tvrzení dostaneme pouhou substitucí $u = u - v$, případně se analogický postup aplikuje pro násobení zleva u^T. ■

Nyní se podívejme na soustavu rovnic optimality:

$$v(s) = \sup_{a \in A_S} \{r(s,a) + \sum_{j \in S} \lambda p(j|s,a)v(j)\}. \tag{52}$$

Tyto rovnice budeme nazývat *Bellmanovy rovnice.*

Definujme si operátor $\mathcal{L} : V \to V$ jako:

$$\mathcal{L}v = \sup_{d \in D^{\mathrm{MD}}} \{r_d + \lambda P_d v\}. \tag{53}$$

Nejdříve se podívejme na vlastnosti tohoto operátoru v závislosti na vztahu v a optimální strategie v_λ^*:

Věta 6. *Nechť* $v \in V$, *potom platí:*

- *pokud* $v \geqq \mathcal{L}v$, *pak* $v \geqq v_\lambda^*$,
- *pokud* $v \leqq \mathcal{L}v$, *pak* $v \leqq v_\lambda^*$,
- *pokud* $v = \mathcal{L}v$, *pak* $v = v_\lambda^*$.

Důkaz. Začneme první vlastností, zvolme libovolnou strategii $\pi = (d_1, \dots)$ pak $v_\lambda^\pi = \sum_{t=1}^{\infty} \lambda^{t-1} P_\pi^{t-1} p_{d_t}$. Z $v \geqq \mathcal{L}v$ získáme $v \geqq r_{d_i} + \lambda P_{d_i} v$ pro libovolné $d_i \in \pi$, rozepíšeme tedy:

$$v \geqq r_{d_1} + \lambda P_{d_1} v \geqq r_{d_1} + \lambda P_{d_1} r_{d_2} + \lambda^2 P_{d_1} P_{d_2} v \geqq \dots \tag{54}$$

Toto platí pro libovolný počet kroku n, odečteme nyní v_λ^π na obou stranách nerovnice:

$$v - v_\lambda^\pi \geqq \lambda^n P_\pi^n v - \sum_{t=n}^{\infty} \lambda^t P_\pi^t p_{d_{t+1}}. \tag{55}$$

Zvolme $\epsilon > 0$ libovolné. Pak z $\|\lambda^n P_\pi^n v\| \leqq \lambda^n \|v\|$ a $0 \leqq \lambda < 1$ existuje dostatečně velké $n \in \mathbb{N}$, že platí:

$$-(\epsilon/2)e \leqq \lambda^n P_\pi^n v \leqq (\epsilon/2)e. \tag{56}$$

Druhý člen pak můžeme odhadnout, stačí zespodu, pomocí předpokladu o omezenosti odměn a toho že součet prvků na libovolném řádku P je rovna 1

$$-\frac{\lambda^n M e}{1-\lambda} \geqq \sum_{t=n}^{\infty} \lambda^t P_\pi^t p_{d_{t+1}}. \tag{57}$$

Tato nerovnice taky platí až od určitého, obecně jiného, n. Vezmeme tedy takový index pro který platí oba naše odhady pak:

$$v \geqq v_\lambda^\pi - \epsilon. \tag{58}$$

Pokud aplikujeme limitu $\epsilon \to 0$ a poté suprémum přes $\pi \in \Pi^{\mathrm{MD}}$, dostaneme požadovanou vlastnost.

Pro důkaz druhé vlastnosti zvolíme opět libovolné $\epsilon > 0$, z předpokladu

plyne existence $d \in D^{\mathrm{MD}}$ splňující: $v \leqq r_d + \lambda P_d v + \epsilon e$. To upravíme na $v \leqq (I - \lambda P_d)^{-1}(r_d + \epsilon e) = v_\lambda^{d^\infty} + (I - \lambda P_d)^{-1}\epsilon e$ pomocí věty 4. Opět pomocí limity $\epsilon \to 0$ a aplikací suprema přes strategie dostaneme druhou vlastnost.

Třetí vlastnost je pak kombinace prvních dvou.

■

První dvě vlastnosti nejsou příliš zajímavé, spíš slouží k důkazu třetí. Právě z té třetí vidíme, že námi hledané řešení, pokud tedy existuje, je pevným bodem našeho operátoru.

Znovu jsme tedy narazili na pojem pevný bod, s ním souvisí následující známá věta věta:

Věta 7 (Banachova věta o pevném bodě)**.** *Nechť U je Banachův prostor a operátor $T : U \to U$ je kontrakce, tedy $\exists \lambda \in [0, 1) \forall u, v \in V : \|Tv - Tu\| \leqq \lambda \|v - u\|$. Pak:*

- *Existuje právě jedno $v^* \in U$ splňující $Tv^* = v^*$ a*
- *Pro libovolné v^0 z U, konverguje posloupnost $\{v^n\}_{n=0}^{+\infty}$ definovaná pro každé přirozené n, jako $v^n = Tv^{n-1} = T^n v^0$ k v^**

Důkaz. Začneme důkazem konvergence posloupnosti z druhého bodu věty. Nechť $m \geqq 1$:

$$\|v^{n+m} - v^n\| \leqq \sum_{k=0}^{m-1} \|v^{n+k+1} - v^{n+k}\| \leqq \sum_{k=0}^{m-1} \|T^{n+k}v^1 - T^{n+k}v^0\|$$
$$\leqq \sum_{k=0}^{m-1} \lambda^{n+k}\|v^1 - v^0\| = \frac{\lambda^n(1 - \lambda^m)}{(1 - \lambda)}\|v^1 - v^0\|$$

První nerovnost je přičtení nuly a trojúhelníkové nerovnosti, pak použijeme definiceí prvků posloupnosti a toho že T je kontrakce. $\{v^n\}_{n=0}^{+\infty}$ je tedy cauchyovská a tedy v Banachově prostoru i konvergentní, označme limitu v^* a

dokažme že se jedná o v^* ze znění věty.

$$0 \leqq \|Tv^* - v^*\| \leqq \|Tv^* - v^n\| + \|v^n - v^*\| = \|Tv^* - Tv^{n-1}\| + \|v^n - v^*\|$$
$$\leqq \lambda\|v^* - v^{n-1}\| + \|v^n - v^*\|$$

Protože $\{v^n\}_{n=0}^{+\infty}$ konverguje k v^* můžeme vhodnou volbou n, zmenšit pravou stranu pod libovolně malou hodnotu a limitně získáme $0 = \|Tv^* - v^*\|$. Jednoznačnost je pak vidět téměř okamžitě z důkazu sporem, necht $u, v \in V$ jsou dva různé pevné body V:

$$\|v - u\| = \|Tv - Tu\| \leqq \lambda\|v - u\|$$

Což je spor, protože λ je ostře menší než 1 . ■

Tato věta nám dává velice silný nástroj, pokud by tedy operátor $\mathcal{L}$ byl kontrakce. Podívejme se, že tomu tak skutečně je:

Věta 8. *Nechť $\lambda \in [0, 1)$, pak $\mathcal{L}$ je kontrakce na V.*

Důkaz. Mějme $u, v \in V$, $s \in S$ libovolné pevné. Začněme například s možností $Lv \geqq Lu$ a zvolme $a_s^* \in \arg\sup_{a \in A_s}\{r(s,a) + \lambda\sum_{j \in S} p(j|s,a)v(j)\}$. Mějme na paměti, že pracujeme v supremovské normě. Pak

$$0 \leqq Lv(s) - Lu(s) \leqq r(s, a_s^*) + \lambda\sum_{j \in S} p(j|s, a_s^*)v(j) - r(s, a_s^*) - \lambda\sum_{j \in S} p(j|s, a_s^*)u(j)$$
$$= \lambda\sum_{j \in S} p(j|s, a_s^*)(v(j) - u(j)) \leqq \lambda\sum_{j \in S} p(j|s, a_s^*)\|v - u\| = \lambda\|v - u\|.$$

Pro předpoklad $Lu(s) \geqq Lv(s)$ bychom dostali po stejném postupu $Lu - Lv \leqq \lambda\|v-u\|$. Dohromady pak $|Lv(s)-Lu(s)| \leqq \lambda\|v-u\|$. Aplikací suprema přes s na rovnici dostaneme $\|Lv-Lu\| \leqq \lambda\|v-u\|$ což jsme chtěli dokázat ■

Nyní již nám nic nebrání dokázat tvrzení, které bylo naším cílem:

Věta 9. *Nechť platí předpoklady ze začátku kapitoly, konkrétně stacionarita odměn a pravděpodobností přechodů, omezenost odměn, diskontování pomocí* $\lambda \in [0, 1)$ *a* S *je nejvýše spočetná množina. Pak existuje právě jedno* $v^* \in V$ *splňující* $\mathcal{L}v^* = v^*$. *Navíc* $v^* = v^*_\lambda$.

Důkaz. Důkaz je nyní již jednoduchý, existence a jednoznačnost plyne z věty 7. A rovnost $v^* = v^*_\lambda$ z věty 6. ■

Optimální očekávaná odměna tedy existuje a je dána jednoznačně. Strategii která jí nabývá pak můžeme identifikovat poměrně jednoduše s pomocí následující věty:

Věta 10. *Strategie* $\pi^* \in \Pi^{\mathrm{MD}}$ *je optimální právě tehdy, když je* $v^{\pi^*}_\lambda$ *řešení Bellmanových rovnic.*

Důkaz. Pokud by byla π^* optimální, pak ihned dostáváme $v^{\pi^*}_\lambda = v^*_\lambda$. Opačnou implikaci dostaneme z věty 9 a toho, že $v^{\pi^*}_\lambda$ jakožto řešení Belmanových rovnic je pevným bodem $\mathcal{L}$. ■

Stojí za to poznamenat, že tato věta nám poskytuje možnost, jak hledat optimální strategii. Teoreticky můžeme generovat strategie a poté je testovat pomocí této věty. O nepraktičnosti tohoto řešení snad nemusím nikoho přesvědčovat. Navíc tato strategie již jednoznačně dána není.

Tyto výsledky již jsou poměrně silné, my bychom ovšem pořád chtěli víc. Zajímalo by nás, jestli náhodou mezi všemi optimálními strategiemi neexistuje alespoň jedna stacionární. Pro naše zkoumání se nejdříve seznámíme s následujícím pojmem:

Definice 6. Pro dané $v \in V$ nazveme rozhodovací pravidlo $d^* \in D^{\mathrm{MD}}$ *vylepšujícím* v, nebo *zachovávajícím* v, pokud:

$$d^* \in \arg\max_{d \in D^{\mathrm{MD}}} \{r_d + \lambda P_d v\},$$

nebo ekvivalentně po složkách:

$$r(s, d^*(s)) + \sum_{j \in S} \lambda p(j|s, d^*(s)) v(j) = \max_{a \in A_s} \{r(s,a) + \sum_{j \in S} \lambda p(j|s,a) v(j)\}$$

No a nyní již k dvojici vět, které nám dovolí zaměřit se právě na stacionární strategie:

Věta 11. *Nechť S je diskrétní množina a supremum $\mathcal{L}$ je nabýváno pro každé $v \in V$. Pak:*

- *Existuje zachovávající rozhodovací pravidlo $d^* \in D^{\mathrm{MD}}$,*
- *pro zachovávající rozhodovací pravidlo d^* je deterministická stacionární strategie $(d^*)^\infty$ optimální a*
- $v_\lambda^* = \sup_{d \in D^{\mathrm{MD}}} v_\lambda^{d^\infty}$.

Důkaz. První bod plyne z toho, že suprema se nabývají, můžeme místo nich psát maxima, což odpovídá definici zachovávajícího rozhodovacího pravidla.

Pro druhý bod si musíme vzpomenout na to, že z věty6 existuje právě jedno řešení $\mathcal{L}v = v$ a to v_λ^*.

$$v_\lambda^* = \mathcal{L}v_\lambda^* = r_{d^*} + \lambda P_{d^*} v_\lambda^* = \mathcal{L}v_\lambda^* = \mathcal{L}v_\lambda^*$$

tedy

$$v_\lambda^* = (I - \lambda P_{d^*}) r_{d^*}$$

ze věty 4 pak víme, že se musí jednat právě o $v_\lambda^{(d^*)^\infty}$.

No a třetí bod platí, protože se toto supremum nabývá právě v d^*. ■

Věta 12. *Pokud existuje zachovávající rozhodovací pravidlo, nebo optimální strategie, pak existuje optimální strategie která je stacionární.*

Důkaz. Pokud existuje zachovávající rozhodovací pravidlo, tak se stačí odkázat na druhý bod předchozí věty 11.

Zajímavější část tvrzení pak můžeme dokázat tak, že zvolíme libovolnou optimální strategii $\pi \in \Pi^{\mathrm{MD}}$, pak existuje $d' \in D^{\mathrm{MD}}$ takové, že platí:

$$v_\lambda^\pi = r_{d'} + \lambda P_{d'} v_\lambda^\pi \leqq \sup_{d \in D^{\mathrm{MD}}} \{r_d + \lambda P_d v_\lambda^\pi\} = \mathcal{L} v_\lambda^\pi = v_\lambda^\pi, \tag{59}$$

tedy d' je zachovávající a dostaneme se rovnou na první případ. ■

2. Algoritmy řešení MDP

Nyní se již konečně dostáváme k samotným způsobům řešení našeho problému. Konkrétně si ukážeme tři nejběžnější algoritmy. V textu si uvedeme pseudokódy těchto algoritmů, jejich plnou implementaci v jazyce Python najde čtenář v příloze.

Zajímat nás bude i rychlost konvergence, proto si o ní něco připomeneme:

Definice 7. Nechť posloupnost $\{x_n\}$ konverguje v normě k bodu x. Existuje-li $p \geqq 1$ a konstanta C nezávislá na n, pro něž platí:

$$\lim_{n \to +\infty} \frac{\|x^{k+1} - x\|}{\|x^k - x\|^p} = C, \tag{60}$$

řekneme, že *řád konvergence* $\{x_n\}$ je roven p.

Speciálně pak:

- pro $p = 1$ a $C \in (0, 1)$ hovoříme o *lineární konvergenci*
- pro $p = 1$ a $C = 0$, nebo $1 < p < 2$ a $C > 0$, pak o *superlineární konvergenci*
- a nakonec pro $p = 2$ hovoříme o *kvadratické konvergenci*

2.1. Policy iteration

První algoritmus je založen na hledání stacionárního rozdělení, poté co jej najdeme pro dané rozhodovací pravidlo, napočítáme jeho funkci odměn. Načež hledáme, jestli existuje jiné rozhodovací pravidlo, které by vedlo k vylepšní.

Začněme s konvergenční větou:

Věta 13. *Nechť S a A jsou konečné množiny, pak*

Algoritmus 1 Policy iteration

Zvol libovolné $d^0 \in D$
$n = 0$
while true do
  Spočti v^n pomocí: $(I - \lambda P_{d_n})v^n = r_d$
  Zvol $d_{n+1} \in \arg\max_{d \in D}\{r_d + \lambda P_d v^n\}$, pokud je to možné $d_n = d_{n+1}$
  if $d_{n+1} = d_n$ **then**
    return d_n
  else
    $n = n + 1$
  end if
end while

- *Posloupnost* $\{v^n\}$ *generovaná pomocí algoritmu Policy iteration konverguje v normě k optimu a to monotonně.*

- *Algoritmus najde v konečném počtu kroků optimální strategii.*

existuje-li navíc K, $0 < K < +\infty$, *splňující pro každé* $n \in \mathbb{N}$:

$$\|P_{v^n} - P_{d_{v_\lambda^*}}\| \leqq K\|v^n - v_\lambda^*\|$$

kde P *jsou matice pravděpodobností přechodu pro dané rozhodovací pravidla, pak platí:*

$$\frac{\|v_\lambda^{n+1} - v_\lambda^*\|}{\|v_\lambda^n - v_\lambda^*\|^2} \leqq \frac{K\lambda}{1-\lambda} \tag{61}$$

Konvergence algoritmu je tedy kvadratická.

Pokud by platilo $\lim_{n\to+\infty} \|P_{v^n} - P_{d_{v_\lambda^*}}\| = 0$, pak algoritmus konverguje dokonce superlineárně.

Největší problém tohoto algoritmu spočívá v řešení $(I - \lambda P_{d_n})v^n = r_d$. I přes to že se jedná o soustavu lineárních rovnic, tato soustava může být velká a tudíž její řešení náročné. Zjednodušeně řečeno algoritmus potřebuje

poměrně málo kroků, každý krok je ovšem náročný. Tento problém pak řeší poslední algoritmus v této kapitole 2.3, díky jehož existenci se Policy iteration téměř nepoužívá.

Pokud by čtenáře zajímal důkaz věty 14 nebo více k tomuto algoritmu, může nahlédnout do [2, Chapter 6.4.].

2.2. Value iteration

V tomto algoritmu se na problém podívám z hlediska operátoru $\mathcal{L}$, definovaný zde 46. Hlavní myšlenka algoritmu spočívá v tom, že v každém kroku aplikujeme tento operátor, dokud nedosáhneme pevného bodu.

Algoritmus 2 Value iteration

Zvol libovolné $v^0 \in V$ a $\epsilon > 0$
$n = 0$
while true do
  for all $s \in S$ **do**
    $v^{n+1}(s) = \max_{a \in A_S}\{r(s,a) + \sum_{j \in S} \lambda p(j|s,a)v(j)\}$
  end for
  if $\|v^{n+1} - v^n\| < \frac{\epsilon(1-\lambda)}{2\lambda}$ **then**
    for all $s \in S$ **do**
      $d_\epsilon(s) \in \arg\max_{a \in A_S}\{r(s,a) + \sum_{j \in S} \lambda p(j|s,a)v^{n+1}(j)\}$
    end for
    return d_ϵ
  else
    $n = n + 1$
  end if
end while

Podívejme se na konvergenční větu:

Věta 14. *Nechť $\epsilon > 0$, zvolme libovolnou funkci odměn v_0*

- *Posloupnost $\{v^n\}$ generovaná pomocí algoritmu Value iteration konverguje v normě k optimu.*

- *Algoritmus najde v konečném počtu kroků $\epsilon-$optimální strategii.*
- $\frac{\|v_\lambda^{n+1}-v_\lambda^*\|}{\|v_\lambda^n-v_\lambda^*\|} \leqq \lambda$. *Konvergence algoritmu je tedy lineární.*

Konvergenční vlastnosti se dají zlepšit pomocí různých implementací. Velice přirozená, alespoň mně to tak přijde, je Gaus-Seidelova verze algoritmu. Hlavní myšlenkou této implementace je aktualizace funkce odměn složku po složce, a pokud je to možné používáme hodnoty, které již byly aktualizovány v této iteraci.

Pokud by čtenáře zajímal důkaz věty 14 nebo více k tomuto algoritmu, může nahlédnout do [2, Chapter 6.3.].

2.3. Modified policy iteration

Předchozí algoritmy měli každý své výhody a nevýhody, co kdybychom se tedy pokusili zkombinovat oba tyto přístupy? Zlepšení spočívá v nahrazení naší soustavy lineárních rovnic a jejího řešení, $(I - \lambda P_{d_n})v^n = r_d$, aplikací našeho operátoru L, viz 46.

Takto získáme nějakou vylepšenou strategii, kterou pak použijeme jako startovní v následujícím kroku.

Můžeme nahlédnout, že pokud položíme $m_n = 0$, získáme Policy iteration, naopak pokud povolím m_n jít do nekonečna, dostaneme Value iteration.

Nyní ke konvergenční větě:

Věta 15. *Nechť $\epsilon > 0$, zvolme libovolnou funkci odměn v_0, pak pro libovolnou posloupnost $\{m_n\}$ platí:*

- *Posloupnost $\{v^n\}$ generovaná pomocí algoritmu Modified policy iteration konverguje v normě k optimu a to monotonně.*
- *Algoritmus najde v konečném počtu kroků $\epsilon-$optimální strategii.*

Algoritmus 3 Modified policy iteration

Zvol libovolné $v^0 \in V$, $\epsilon > 0$ a (neklesající) posloupnost kladných čísel $\{m_n\}_{n=0}^{+\infty}$
$n = 0$
while true do
  Zvol $d_{n+1} \in \arg\max_{d \in D}\{r_d + \lambda P_d v^n\}$, $d_n = d_{n+1}$, je-li to možné $d_n = d_{n+1}$.
  $u_n^0 = \max_{d \in D}\{r_d + \lambda P_d v^n\}$
  if $\|u_n^0 - v^n\| < \frac{\epsilon(1-\lambda)}{2\lambda}$ **then**
    return $d_\epsilon = d_{n+1}$
  else
    for k in range $(0, m_n)$ **do**
      $u_n^{k+1} = r_{d_{n+1}} + \lambda P_{d_{n+1}} u_n^k$
    end for
    $v^{n+1} = u_n^{m_n}$
    $n = n + 1$
  end if
end while

- $\frac{\|v_\lambda^{n+1} - v_\lambda^*\|}{\|v_\lambda^n - v_\lambda^*\|} \leqq \lambda^{m_n+1} + \epsilon$

Konvergence algoritmu je tedy lineární.

Modified policy iteration tedy konverguje rychleji než původní Policy iteration, původní konverguje kvadraticky zatímco tento lineárně, což z původního algoritmu udělalo pouhý příklad pro skripta, který se nepoužívá v praxi.

Logická otázka je pak jak volit m_n? Tento problém není ještě zcela vyřešen. Zajímavým výsledkem zkoumání tohoto problému je ovšem to, že pokud dovolíme m_n růst v závisloti na n nade všechny meze, bude algoritmus teoreticky konvergovat superlineárně.

Pokud by čtenáře zajímal důkaz věty 14 nebo více k tomuto algoritmu, může nahlédnout do [2, Chapter 6.5.].

3. Praktická část

Konečně se dostáváme k samotné praktické části této práce. V té se podíváme jednak na problém v energetice, na nějž by se daly použít právě markovské rozhodovací procesy. Podíváme se i na jednotlivé algoritmy řešení a srovnáme jejich vlastnosti na reálných datech.

3.1. Popis situace

Celý náš problém se točí okolo takzvaného principu „Vylij, Nalij“. Konkrétněji jde o to, abychom v přenosové soustavě udržovali stabilní napětí. Aby tomu tak bylo je třeba, aby objem elektřiny dodané do sítě („Nalij“), byl stejný jako objem odebrané („Vylij“).

Tuto rovnováhu na území České republiky hlídá společnost pod názvem Česká elektroenergetická přenosová soustava, zkráceně ČEPS. Aby ji zajistila, používá systém pokut a odměn, jak již to tak bývá, pokuty jsou větší než odměny. Společnosti musí dopředu udat kolik energie do sítě dodají a kolik jí ze sítě odeberou. Toto je pak pro ně nějaká ta nulová hladina, kterou když přesně naplní tak se nemusí o nic víc starat.

Ne vždy je toto ovšem možné. Je těžké s naprostou jistotou dopředu odhadnout kolik energie odebere rodiný dům v danou hodinu, nebo jestli bude svítit slunce a sluneční elektrárna vyrobý přesně dané množství elektřiny. V praxi se tedy téměř nikdy nestane, že by stav sítě byl přesně takový, jak se slíbilo.

Ovšem samotný fakt, že nedojde k naplnění tohoto slibu nevede automaticky k pokutě. Naopak může dokonce vést i k odměně. Například pokud dodá elektrárna více energie do sítě během hodiny, v níž elektřina v síti chybí, dostane odměnu. Pokud si nazveme rozdíl mezi dodaným a slíbeným množ-

Výchylka sítě \Moje výchylka	Dodám víc	Dodám míň
Přebytek	POKUTA	odměna
Nedostatek	odměna	POKUTA

Tabulka 1: Schema pokut ČEPS

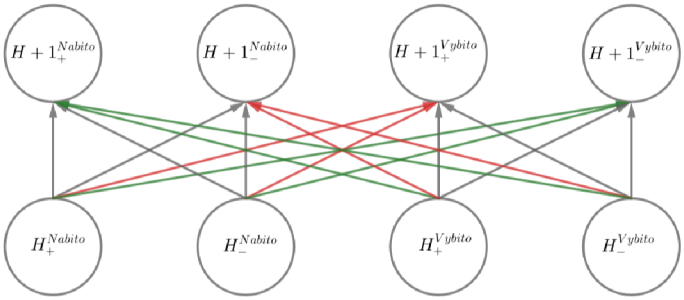
stvím energie společností jako odchylka firmy, tedy odchylka firmy = dodaný objem energie − slíbený objem energie, a totéž uděláme pro celou přenosovou soustavu, odchylka soustavy = celkový objem dodané energie − očekávaný stav. Můžeme jednoduše říct, že pokud má odchylka firmy a odchylka sítě stejné znaménko, pak bude firma platit pokutu, protože její akce vedou k nerovnováze v síti. Pokud mají tyto odchylky opačné znaménko, dostane naopak odměnu, protože její akce podporují stabilitu sítě. Tento systém je možné nahlédnout v tabluce 3.1. Tyto odchylky byly v době zaznamenávání používaných dat měřeny co hodinu, ovšem tyto intervaly se mají zkracovat. Byla vyslovena teorie, že odchylky sítě splňují Markovovu vlastnost, tato domněnka je v této práci považována za pravdivou.

Pokud bychom tedy byli schopni předpovídat výchylky sítě v následující hodině a měli k tomu baterii, dokazali bychom generovat peníze jen tím, že bychom přispívali ke stabilitě sítě a to je přesně to, o co se nyní pokusíme.

3.2. Řešení

Nejdříve si něco řekneme k možným akcím. Budou konkrétně dvě akce *Nabít* a akce *Vybít*. Na první pohled by někdo mohl hledat akci *Nedělej nic*. Ta je ovšem nyní schovaná uvnitř již uvedených akcí, předpokládáme totiž, že baterii buď celou vybijeme nebo nabijeme. Nabíjení nabité baterie tedy můžeme překládat jako „Nedělej nic“.

Zpracovávat budeme data za půl roku, konkrétně od 1.7.2022 do 31.12.2022. Měření jsou prováděna co hodinu, dohromady máme tedy celkově 4416 po-



Obrázek 1: Schéma stavové struktury

zorování.

Pro tuto práci budeme předpokládat, že se dny mezi sebou příliš neliší a vytvoříme řetězec pro libovolný celý den. Stavy budou dány pomocí trojice charakteristik, konkrétně počáteční hodiny (do 0 do 23), stavu baterie (nabito x vybito) a znaménka výchylky v této hodině (+ nebo -). Označíme $hodina^{Stavbaterie}_{vchylka}$, například 12_-^{Nabito} značí, že ve dvanáct hodin byla baterie nabitá a mezi dvanáctou a jednou byl v síti nedostatek energie. Máme tedy čtyři stavy na každou hodinu, dohromady 96 stavů. Stavy jsou seřazeny následovně: $0_+^{Nabito}, 0_-^{Nabito}, 0_+^{Vybito}, 0_-^{Vybito}, 1_+^{Nabito}, \ldots$. Tato zavedení nám vytvoří určitou strukturu na množině stavů, ne nepodobnou schematu neuronové sítě. Jednotlivé vrstvy budou tvořit stavy v příslušné daným hodinám, a stavy v po sobě jdoucích hodinách jsou vzájemně propojené, a pak se musí také propojit vrstvy příslušné 23. a 0. hodině. Schema přechodů mezi dvěmi vrstvami je možné si prohlédnout na obrázku: 1, kde zelené šipky značí přechod mezi stavy za který dostaneme odměnu, za červené pokutu a ty ostatní by odpovídaly tomu, že jsme nic neudělali.

Nyní se podívejme na matice přechodu jako takové. Můžeme přecházet vždy jen ze stavů v hodině h do hodiny $h+1$, respektive z 23 do 0. Většina pravděpodobností přechodů tedy bude nutně rovna nule a nenulové hodnoty

$$\begin{pmatrix} 0 & P_{0,1} & 0 & 0 & \dots \\ 0 & 0 & P_{1,2} & 0 & \dots \\ & & & \ddots & \\ 0 & 0 & 0 & \dots & P_{22,23} \\ P_{23,0} & 0 & 0 & 0 & \dots \end{pmatrix}$$

Obrázek 2: Schema matice pravděpodobností přechodu

$$\begin{pmatrix} p_{4h,4h+4} & p_{4h,4h+5} & 0 & 0 \\ p_{4h+1,4h+4} & p_{4h+1,4h+5} & 0 & 0 \\ p_{4h+2,4h+4} & p_{4h+2,4h+5} & 0 & 0 \\ p_{4h+3,4h+4} & p_{4h+3,4h+5} & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & p_{4h,4h+6} & p_{4h,4h+7} \\ 0 & 0 & p_{4h+1,4h+6} & p_{4h+1,4h+7} \\ 0 & 0 & p_{4h+2,4h+6} & p_{4h+2,4h+7} \\ 0 & 0 & p_{4h+3,4h+6} & p_{4h+3,4h+7} \end{pmatrix}$$

Obrázek 3: Tvar matice $P_{h,h+1}$ pro akci nabít (vlevo), respektive vybít(vpravo)

budou pouze v určitých blocích, viz schema 2, kde matice $P_{h,h+1} \in \mathbb{R}^{4x4}$ obsahují pravděpodobnosti přechodu mezi stavy v hodině h a $h+1$.

Další nuly v matici přechodu vzniknou v závislosti na volbě akce, nabíjením nabité baterie ji není možné vybít. I matice $P_{h,h+1}$ tedy budou mít pro každou akci specifický tvar. Pro akci nabít budou nenulové první dva sloupce, pro akci vybít pak bude nenulový třetí a čtvrtý, opět zde pro rychlejší pochopení obrázek 3.

V definici nám zbývá už jen matice odměn pro obě akce. I když teoreticky potřebujeme dvě, a implementace počítá se dvěma, my si vystačíme jen s jednou. To díky tomu, že přechody mezi stavy, za které bychom pro jednu akci dostali odměnu, nebo pokutu, mají pro druhou akci pravděpodobnost přechodu nula. Celková matice odměn bude mít stejnou strukturu jako matice

Obrázek 4: Schéma matice odměn R

pravděpodobností přechodů, viz obrázek 2, jen místo matic $P_{h,h+1}$ do tohoto schémátka dosadíme matici odměn R. Tvar matice R jsme již trošku nakousli, když jsme řešili schéma na obrázku 1. Schematicky si ji můžeme prohlédnout na obrázku 4.

3.2.1. Poznámky k implementaci

Z poznatků v minulé kapitole můžeme pro náš problém dojít k ještě několika závěrům, které by nám mohli pomoct najít lepší způsob řešení našeho problému.

První pozorování je, že matice v našem problému jsou sice velké, co se týče rozměrů, ovšem poměrně řídké. Bylo by tedy vhodné podívat se, jestli použití *řídkých matic* nebude žádoucí.

I kdyby jejich použití nevedlo ke zrychlení algoritmu, mohli bychom pomocí takovéto implementace umožnit využití těchto algoritmů i v období,

kdy se budou výchylky měřit každých 5 minut, to by bylo $12 * 24 * 4 = 1152$ stavů, pokud bychom nic jiného nezměnili. Mohli bychom narazit na limitace hardwaru.

Co to vůbec jsou řídké matice? Velice jednoduše se dá říct, že matice je řídká, pokud vetšina jejích prvků je rovna 0. A na co jsou dobré? V softwaru je typicky možné takovéto matice ukládat specifickým způsobem. Zaznamenány nejsou celé matice, ale pouze nenulové hodnoty a jejich souřadnice. Každý nenulový prvek m_{ij} libovolné matice M se dá vyjádřit jako uspořádaná trojice (i, j, m_{ij}). Tyto trojice se pak uspořádají podle daného klíče, typicky po řádcích (tzv *CSR formát*). Ukažme si na příkladu jak matici uložit ve formátu CSR:

Příklad 3. Podívejme se na matici $P_{0,1}$ z našeho problému pro akci Nabít:

$$P_{0,1} = \begin{pmatrix} 0.65 & 0.35 & 0 & 0 \\ 0.27 & 0.73 & 0 & 0 \\ 0.65 & 0.35 & 0 & 0 \\ 0.27 & 0.73 & 0 & 0 \end{pmatrix}$$

tuto matici můžeme jako řídkou napsat jako:

$$\text{Data} = (0.65, 0.35, 0.27, 0.73, 0.65, 0.35, 0.27, 0.73)$$

$$\text{Sloupec} = (0, 1, 0, 1, 0, 1, 0, 1)$$

$$\text{Řádek} = (0, 2, 4, 6)$$

Kde pole Data obsahuje zaznamenané hodnoty a jeho délka odpovídá počtu nenulových prvků. Pole Sloupec má stejnou délku a obsahuje index

sloupce. Nakonec pole Řádek má vždy délku rovnou počtu řádků a obsahuje index prvního prvku v poli Data obsahující prvek z daného řádku.

★

Čeho dosáhneme použitím tohoto formátu? Mějme libovolnou matici $M \in \mathbb{R}^{n \times n}$ a v ní k nenulových prvků. Řekněme že na uložení jednoho čísla použijeme 8 bajtů, to odpovídá pythonovskému floatu. Na uložení této matice ve standardním formátu použijeme $8 * n^2$ bajtů. Pro uložení stejné matice jako řídké jen jako uspořádané trojice potřebujeme celkem $8 * 3 * k$ bajtů. Pro formát CSR pak potřebujeme dokonce jen $8 * (2 * k + n)$. Lehce nahlédneme, že pokud platí $n^2 > 2 * k + n$, dojde k úspoře paměti uložíme-li matici ve formátu CSR. Pro náš příklad 3 je lepší tradiční implementace, protože $16 < 20$, ale pokud bychom se podívali na celou matici přechodu dostaneme $96^2 = 9216 > 2 * 2 * 96 + 96 = 480$. Takováto implementace by tedy měla být výhodná. Více o řídkých maticích najdete například v [3].

Druhé pozorování je, že specifická struktura stavů nám nabízí ještě jednu možnost jak nemuset používat tak obrovské matice. Podíváme-li se na algortimus Value iteration, z kapitoly 2.2, zjistíme, že abychom aktualizovali hodnoty funkce odměna příslušné stavům v hodině $h + 1$, matici pravděpodobností přechodů z hodiny h do hodiny $h + 1$, odměny příslušící těmto přechodům a hodnoty funkce odměn příslušné stavům hodiny h.

V našem příkladě to znamená, že nebudeme aktivně používat matice 96×96 ale 4×4. Navíc velikost těchto matic nebude závislé na frekvenci měření výchylek. Samozřejmě je otázkou, jestli takováto implementace bude lepší než tradiční algoritmy s řídkými maticemi.

3.2.2. Řešení problému

Podívejme se nyní, jak se nám povede vyřešit náš problém s baterkou. Na to jsme nejdříve museli získat z dat pravděpodobnosti přechodu. Ty jsou získané z celého datasetu, nebyly tedy použity testovací a trénovací skupiny. Důvodem bylo, že přechody mezi stavy s kladnou a zápornou výchylkou jsou v datasetu zastoupeny poměrně řídce, často jsou v hodnotách jednotek. Omezení se na testovací a trénovací data často vedlo k naprosté ignoraci těchto přechodů. I takto některé přechody nikdy nenastaly například pokud byla výchylka mezi 1:00 a 2:00 kladná, tak tomu tak bylo i v následující hodině, ani jednou za půl roku se nestalo jinak.

Velikost odměn a pokut jsem naškáloval jako 1:5. Navíc předpokládáme, že naše baterie není dost velká na to, aby zcela obrátila výchylku. Diskontní faktor λ jsem položil roven 0.9. V iniciační strategii platí, že pro sudé stavy a nulu jsem zvolil akci nabít pro liché pak vybít.

Pro takovouto situaci, jsem byli schopni nalézt strategii, která vede ke kladné očekávané odměně. A která je stejná pro všechny algoritmy, tedy optimální strategie získaná pomocí Policy iteration je stejná jako $\epsilon-$ optimální strategie získané ostatnímy algoritmy.

Vzhledem k tomu že tato stategie σ, která nám v průměru přinese téměř 0.75*velikost odměny, je odpovědí na naši původní otázku, napišme si ji. Pomůžeme si značením N = nabít, V = vybít a I = „nic nedělej“:

$\sigma = (I, V, I, I, I, I, I, I, I, V, N, I, I, V, N, I, I, V, N, I, I, V, N, I,$
$I, I, N, I, I, V, I, I, I, V, I, I, I, V, I, I, I, I, I, I, I, V, I, I, I, I, N, I, I,$
$V, I, V, I, V, I, I, I, V, N, I, I, V, N, I, I, V, I, I, I, V, I, I, I, I, N, I, I,$
$V, I, I, I, V, I, I, I, I, N, I, I, I, N, I, I, I, I, I).$

Ještě poznámka k interpretaci: pokud ve stavu s v hodině h přísluší akce a, znamená to, že pokud se ocitneme v tomto stavu s tak během příští hodiny

	CČ (ms)	CČ s ŘM (ms)	PČ (ms)	PČ s ŘM (ms)
Policy iteration	2.7982	14.1817	22.0156	112.8906
Value iteration	3.4431	180.5281	4.7031	179.9844
MPI	4.8172	624.376	37.2812	1908.4063
Value iteration blocks	1.5431	-	1.53125	-

Tabulka 2: Porovnání rychlosti jednotlivých algoritmů

se máme řídit akcí a. Odpovídá to situaci kdy se podíváme co se stalo minulou hodinu, tam uvidíme doporučenou akci a tou se budeme řídit v této.

3.2.3. Porovnání algoritmů

Nyní se ještě podíváme na jednotlivé algoritmy a porovnáme, jak rychle jsou schopny nalézt žádané řešení. Respektive se podíváme na jejich implementace v Pythonu. K tomu použijeme dvojici statistik. Konkrétně se podíváme na celkový čas (dále CČ) potřebný k nalezení řešení a procesorový čas (dále PČ).

Každý algoritmus byl spuštěn tisíckrát a pak byla vzána průměrná doba exekuce, abychom zabránili problémům s vlivem náhodně puštěných programů. Je třeba si být vědom, že tyto informace jsou značně závislé na kvalitách hardwaru. Časy jsou pak uvedeny v tabulce 3.2.3.

Pokud se čtenář na první pohled vyděsil, stejně jako autor, že čas na procesoru je typicky delší než výpočetní čas, může se opět uklidnit. Moderní procesory jsou schopné provádět na více vláknech ve stejný okamžik, a tím zrychlit výpočet jako takový.

Zdá se, že co se týče rychlosti algoritmu, je implementace pomocí bloků výhodná. Zdá se také, že Value iteration na procesoru tráví poměrně málo času, což odpovídá tomu, že operace potřebné pro tento algoritmus jsou na úrovni násobení vektorů. Zajímavý je i fakt, že MPI je ze všech algoritmů nejpomalejší. Pravděpodobně jsme splnili předpoklady superlineární konver-

gence pro PI.

Samozřejmě další zajímavost je, jak dlouho jsou zpracovávány algoritmy s řídkými maticemi. Přiznám, že tyto algoritmy je možné zrychlit. Problém spočívá v omezeních které plynou ze způsobu uložení matic, konkrétně ve formát CSR, transponovat takovouto matici je poměrně náročné. Je možné zvolit jiné způsoby uložení, problém s nimi je, že se mi nepodařilo najít verzi, která by zároveň zrychlila algoritmus a zároveň nezvýšila paměťovou náročnost.

Zopakujeme také poznatky o paměťové náročnosti, které jsme započaly v kapitole 3.2.1. Pro uložení všech dat pro klasické algoritmy bez použití řídkých matic potřebujeme dvě matice odměn a dvě matice pravděpodobností přechodu každá o rozměrech 96×96, dohromady $4 * 96^2 = 36864$ hodnot do naší paměti. Pro blokovou implementaci pak potřebujeme místo každé této matice dvacetčtyři matic o rozměru 4×4, dohromady 7680 hodnot tedy asi pětinu. Pokud bychom ukládali matice 96×96 jako řídké potřebovali bychom dohromady $4 * 480 = 1920$ hodnot. Co se týče paměťové náročnosti můžeme tedy s klidem říct, že řídké matice jsou s přehledem nejzajímavější.

4. Závěr

Co se nám tedy v této práci povedlo? Definovali a motivovali jsme pojem markovských rozhodovacích procesů a dokázali si jejich nejdůležitější vlastnosti. Ukázali jsme si algroritmy určené k řešení takovýchto problémů a implementovali jsme je v jazyce python.

Tyto implementace jsme použili k řešení problému z reálného světa, konkrétně v oblasti energetiky a ten se nám povedlo vyřešit. Zaměřili jsme se na způsoby, které by mohli vést k zefektivnění řešení tohoto problému což se nám povedlo specifickou implementací algoritmu Value iteration, která si vedla lépe než jsem očekával.

Rychlost algoritmů pro takovýto problém jsme pak empiricky porovnali pomocí času potřebného pro výpočet a procesorového času, výsledky je možné nahlédnout v tabulce 3.2.3. Některé výsledky nejsou úplně očekávané, především pomalost algoritmů s řídkými maticeli stejně jako fakt, že Policy iteration je rychlejší než její modifikovaná verze.

Podívali jsme se i na problém vzniku velkých matic, které budou potřeba pro řešení podobných problémů v budoucnu, a kromě již zmíněné specifické implementace jsme se pokusili problém vyřešit i použitím řídkých matic, konkrétně jsme zvolili formát CSR, které nám tento problém dá se říci vyřešily. Původní matice zabíraly v paměti téměř dvacetkrát více místa než ty řídké.

Nastaly ovšem i některé problémy. První spočívá ve faktu, že i přesto, že jsem původně myslel, že mám dostatek dat, bych nyní upřednostnil, kdybych měl k dispozici ještě jeden rok navíc, abych jej mohl použít jako testovací set. Stejně tak implementace řídkých matic vedla ke zpomalení algoritmů, což by se ideálně dít nemělo, čemuž napomáhá i má volba formátu CSR. Stejně tak může být na vině má ipmlementace algoritmů.

Náš model by šlo vylepšit, například bychom mohli povolit více akcí.

Bylo by možné povolit nabíjet a vybíjet baterii jen částečně. Zavedli bychom k tomu akce „Nabij o $x\%$“ a „Vybij o $x\%$“ a navíc bychom se při této implementaci neobešli bez akce „Nic nedělej“. Stejně tak by nám přibyly stavy. Místo nabito a vybito bychom uváděli informaci o stavu baterie.

Jen pro představu se ještě zamyslíme, jak velký by tento problém byl. Povolíme-li změnu stavu baterie o násobky deseti, tedy $+0$, $\pm 10\%$, $\pm 20\%$, $\ldots$, $\pm 100\%$, dostaneme 21 akcí, tedy 21 matic přechodu a stejný počet matic odměn. Navíc matice budou násobně větší, každá tato matice bude o rozměrech 528×528, to jest 22 stavů pro každou hodinu. Pro blokovou implementaci z kapitoly 3.2.1 bychom potřebovali matice 22×22. Matice přechodu by byly ještě řidší, protože nadále bychom mohli pomocí jedné akce přejít z jednoho stavu do nejvýše 2 jiných. Ze stavu $0_{-}^{20\%}$ se můžeme pomocí akce „Nabij o 20%“ dostat jen do stavů $1_{+}^{40\%}$ a $1_{-}^{40\%}$.

Podobně by bylo možné zakomponovat i další informace, které by mohli mít vliv na chování sítě. Dá se očekávat, že chování energetické sítě v zimě se bude lišit od jeho chování v létě, i kdyby jen vlivem solárních elektráren. Stejně tak se dá očekávat rozdíl mezi víkendy a pracovními dny. Tyto problémy by měli být řešitelné pomocí stejných algoritmů jen bychom museli „nafouknout“ stavy o další proměnné, která by je definovala.

S tímto ovšem narazíme na něco, co považuji za omezení tohoto způsobu řešení. A to konkrétně náročnost na množství dat, která budou potřeba. Jen to, že budeme měřit výchylku sítě každých pět minut nám vytvoří 1152 stavů za každý den. Navíc si dovolím vyslovit doměnku, že získat dostatek relevantních dat nebude lehké, protože trh s energetikou poslední dobou prochází značnými turbulencemi ať už způsobenými válkou, nebo opatřeními v rámci Green dealu.

Apendix

Zavedení markovských řetězců

V této sekci si ukážeme jak se dá také dospět k definici markovských rozhodovacích procesů. Začneme od nejjednoduššího konstruktu a to *Markovův řetězec*:

Definice 8. Řekneme, že posloupnost diskrétních náhodných veličin $\{X_n\}_{n\in\mathbb{N}_\not}$, které nabývají hodnot z množiny $S \subseteq \mathbb{N}_0$, tvoří *Markovův řetězec*, jestliže platí:

$$P(X_{n+1} = j|X_n = i_n, X_{n-1} = i_{n-1}, X_{n-2} = i_{n-2}, \ldots, X_0 = i_0) = P(X_{n+1}. = j|X_n = i_n) \tag{62}$$

pro každé $n \in \mathbb{N}_\not$ a $i_0, \ldots, i_n, j \in S$ pro které:

$$P(X_n = i_n, X_{n-1} = i_{n-1}, X_{n-2} = i_{n-2}, \ldots, X_0 = i_0) > 0. \tag{63}$$

K Markovovským řetězcům si uvedeme dvě důležité vlastnosti:

Definice 9. Označme pro libovolné $i, j \in S$ a $n \in \mathbb{N}$: $p_{ij}^{(n)}$ pravděpodobnost, že se za n kroků dostaneme ze stavu i do stavu j. Řekneme, že řetězec je *nerozložitelný*, pokud pro každou dvojici stavů $i, j \in S$ existuje $n \in \mathbb{N}$ takové, že $p_{ij}^{(n)} > 0$.

Definice 10. Nechť $\{X_n, n \in \mathbb{N}\}$ je homogenní Markovův řetězec s množinou stavů S a maticí pravděpodobností přechodu P. Nechť $\alpha = \{\alpha_j, j \in S\}$ je rozdělení pravděpodobností na množině S. Potom α nazveme *stacionárním*

rozdělením daného rozdělení, platí-li:

$$\alpha = \alpha P \tag{64}$$

Stacionární rozdělení α nám udává pravděpodobnost, že po ustálení procesu se bude řetězec nacházet v daném stavu.

Často by nás zajímalo, co chování řetězce přinese nám. K tomuto slouží takzvané *Markovovy řetězce s výnosy*, kde nám přibudou, jak již název napovídá, odměny.

Mějme matici $R = (r_{ij})$, kde r_{ij} je odměna za přechod ze stavu i do stavu j. Tuto matici nazveme *matice odměn*. Markovovský řetězec spolu s jemu příslušnou maticí odměn nazveme právě *Markovovův řetězec s výnosy*.

Zatím jsme je analyzovali nějaký proces, nad kterým jsme neměli žádnou kontrolu. Co když můžeme ovlivňovat chování procesu? To nám dovolí právě *markovské rozhodovací procesy*, kterými jsme se zabývali a jejichž definicí jsme začínali, konkrétně definice 1.

Pokud by se čtenář chtěl dozvědět o tomto tématu více, může nahlédnout například do [1].

Přílohy

V příloze pojmenované Suransky_DP_Prilohy.zip je možná najít kódy slíbené v kapitole 2, stejně jako data použitá v praktické části 3.

Literatura

[1] FAČEVICOVÁ, Kamila; HRON, Karel a KUNDEROVÁ, Pavla. Univerzita Palackého v Olomouci Přírodovědecká fakulta MARKOVOVY ŘETĚZCE A JEJICH APLIKACE. 2nd ed. Univerzita Palackého v Olomouci, 2018. ISBN 78-80-244-5432-0.

[2] PUTERMAN a MARTIN. Markov decision processes discrete stochastic dynamic programming. New York : John Wiley, 2005. ISBN 0-471-72782-2.

[3] STOER, Josef; BULIRSCH, Roland. Introduction to Numerical Analysis. 3rd ed.. Berlin, New York: Springer-Verlag. 2002. ISBN 978-0-387-95452-3.