VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

HEURISTIKY PRO HRANÍ HRY SCOTLAND YARD

HEURISTICS FOR THE SCOTLAND YARD BOARD GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL CEJPEK**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. FRATIŠEK ZBOŘIL, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

156221

Ústav: Ústav inteligentních systémů (UITS)
Student: **Cejpek Michal**
Program: Informační technologie
Název: **Heuristiky pro hraní hry Scotland Yard**
Kategorie: Umělá inteligence
Akademický rok: 2023/24

Zadání:

1. Seznamte se s pravidly deskové hry typu "Scotland Yard", kdy pozice jedné z figur bývá protihráčům ukázána jen v některých kolech hry. Také nastudujte výsledky, které pro automatické hraní této hry byly dosaženy.
2. Určete metody, které by měly být důvodně vhodné pro realizaci systému, který bude hrát tuto hru autonomně. Zaměřte vedle klasických metod hraní her i metody pro počítačové učení, jako jsou například metody posilovaného učení a hlubokého učení.
3. Pro jednotlivé role figur ve hře implementujte algoritmy řízení a ověřte jejich schopnost plnit zadané cíle.
4. Vyhodnoťte úspěšnost obou stran hry pro různé míry zapojení metod strojového učení a diskutujte zjištěné výsledky.

Literatura:

- Nijssen, J., A., M., Winands, H., M.:"Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard", IEEE Transactions on Computational Intelligence and AI in Games 4(4):282 - 294, 2012
- Norvig, P., Russel, S. :"Artificial Intelligence, A Modern Approach", Prentice Hall, 2020
- Daniel Borák: "Heuristic Evaluation in the Scotland Yard Game", Bakalářská práce, 2021, ČVUT

Při obhajobě semestrální části projektu je požadováno:
První dva body zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Zbořil František, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 6.11.2023

Abstrakt

Tato práce se zabývá možností použití algoritmů hlubokého a posilovaného učení pro řešení problémů s neúplnou informací. Konkrétně je hlavním zkoumaným algoritmem PPO – Proximal Policy Optimization (optimalizace proximální politiky). K účelu otestování vhodnosti algoritmu PPO, byla vytvořena zjednodušená implementace hry Scotland Yard a také prostředí pro trénování a testování algoritmů.

Z provedených experimentů této práce vzešlo, že algoritmus PPO je velmi vhodný na řešení problémů s neúplnou informací. Agenti při trénování velmi rychle získali pojem o cílech hry a vybudovali vhodné strategie pro naplnění těchto cílů.

Abstract

This thesis explores the possibility of using deep and reinforcement learning algorithms to solve problems with incomplete information. The main algorithm under investigation is PPO – Proximal Policy Optimization. In order to test the suitability of the PPO algorithm, a simplified implementation of the Scotland Yard game was created as well as an environment for training and testing the algorithms.

From performed experiments, it emerged that the PPO algorithm is very suitable for solving problems with incomplete information. The agents very quickly gained a sense of the game's goals and built appropriate strategies to meet those goals through training.

Klíčová slova

DQN, Posilované učení, Proximální optimalizace politiky, Scotland Yard, Umělá inteligence ve hrách

Keywords

Artificial Intelligence in Games, DQN, Proximal Policy Optimization, Reinforcement Learning, Scotland Yard

Citace

CEJPEK, Michal. *Heuristiky pro hraní hry Scotland Yard.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, Ph.D.

Heuristiky pro hraní hry Scotland Yard

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Františka Zbořila, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Cejpek
6. května 2024

Poděkování

Tímto bych rád poděkoval vedoucímu práce, panu doc. Ing. Františku Zbořilovi, Ph.D za jeho cenné rady, trpělivost a čas při vedení této práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V dnešní době, je obor umělé inteligence (AI), postupem let všudypřítomnější a její působení ovlivňuje spoustu aspektů našeho života.

Pokrok umělé inteligence je často a nejlépe měřen její aplikováním v oblasti her. A to proto, že právě hry nabízí jasně definovaná pravidla a cíle, čímž se stávají ideálním testovacím prostorem pro algoritmy umělé inteligence. Výkon AI je ve hrách snadno měřitelný a pokrok dokáže vidět i lajk bez žádných složitých grafů, tabulek a výpočtů. Umělá inteligence již dokázala porazit nejlepší hráče v *šachu* [16], *Dota 2* [17], *Go* [5] a spoustě dalších her.

Hra studovaná v této práci se jmenuje *Scotland Yard*. Jedná se o hru pro tři až šest hráčů. V této hře obvykle hraje jeden hráč jako Pan X, který se snaží uniknout policistům, ovládanými ostatními hráči. Policisté, avšak nevědí, kde na herním poli se Pan X nachází. A musí tedy odhadovat jeho pozici a spolupracovat mezi sebou, aby ho mohli polapit. Pozice Pana X je policistům odhalena pouze v určitých kolech. Scotland Yard je ideální hrou ke studování neurčitosti ve hrách hranými systémy umělé inteligence, protože se jedná o hru s nedokonalou informací a k vítězství policistů je zapotřebí spolupráce, strategie a odhadování neznámé informace.

Tato práce zkoumá algoritmy posilovaného učení, jejich použití na hry s neurčitostí a jejich porovnání s klasickými metodami hraní her. Algoritmus proximální optimalizace strategie (Proximal Policy Optimization–PPO) je hlavním zkoumaným algoritmem této práce. Pro srovnání s jinými algoritmy slouží algoritmus Deep–Q–Learning (DQN). Výslední agenti byli také testování proti hráči, který náhodně vybírá své akce. PPO je často používán k řešení problémů se spojitými veličinami a ve 3D prostoru. Algoritmus PPO byl vybrán z důvodu jeho úspěšnosti při použití ve složitých hrách s neurčitostí. Tyto tvrzení podporuje experiment OpenAI [18], podrobně popsán zde [2]. V tomto experimentu byla proximální optimalizace strategie použita pro učení strategie ve hře typu *schovávaná* s několika agenty a neúplnou informací. Jelikož tento typ hry přesně odpovídá i hře Scotland Yard, zdá se být vhodný i pro řešení této hry. Také další studie [1] a [17] tento výrok dále podporují a dokazují vhodnost algoritmu PPO pro tuto práci.

Zaměření této práce jsem si vybral, jelikož mi vždy byl obor umělé inteligence blízký a vždy jsem se chtěl začít tomuto odvětví více věnovat i po praktické stránce. Algoritmus je pro mě zajímavý a zkušenost s tímto algoritmem by se dala využít v mém dalším pracovním životě.

Pro zpracování práce byly využity tyto hlavní knihovny:

- `Ray.Rlib` - framework pro posilované učení. Obsahuje již zhotovenou implementací algoritmu PPO a DQN, které byly použity v této práci.
- `PyTorch` - podpůrná knihovna `Ray.Rlib`
- `Gymnasium` - knihovna použita k vytvoření prostředí hry Scotland Yard
- `Pygame` - knihovna pro vytváření uživatelského rozhraní

Práce byla zhotovena pro operační systémy Windows a Python verze 3.10

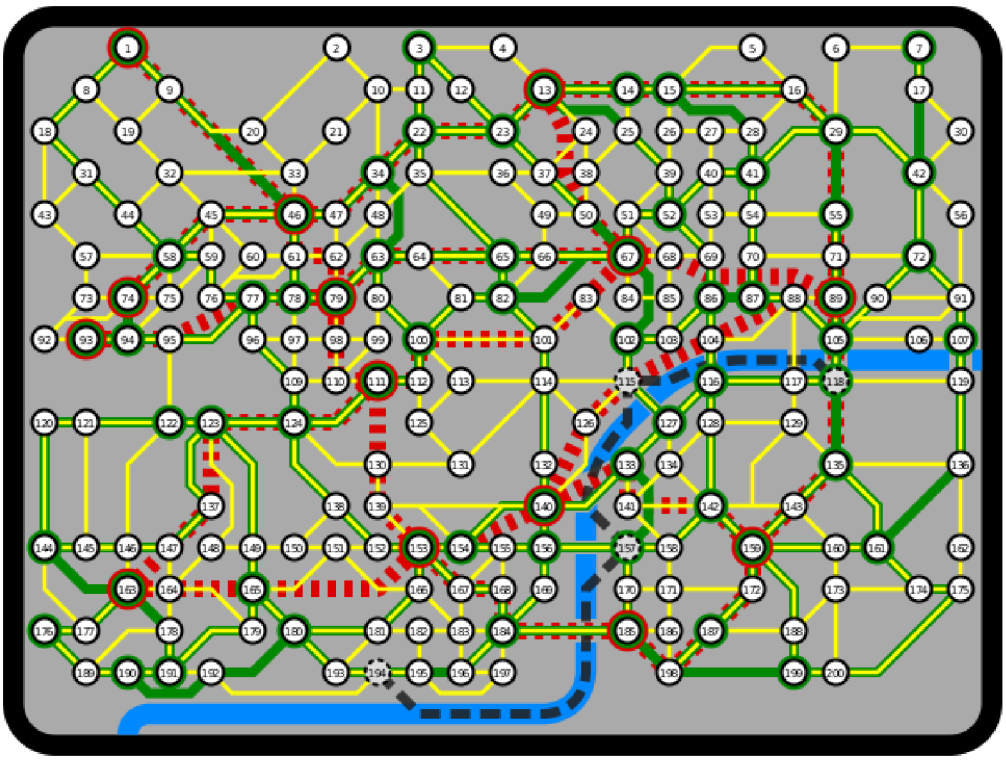
Kapitola 2

Shrnutí dosavadního stavu

Tato kapitola není encyklopedickým přehledem celého tématu bakalářské práce. Nýbrž jedná se o shrnutí nejdůležitějších relevantních informací a pojmů, důležitých pro tuto práci.

2.1 Desková hra Scotland Yard

Scotland Yard je populární hra pro tři a více hráčů, která kombinuje prvky schovávané a hry na honěnou. Jeden hráč hraje za Pana X, který se snaží uniknout policistům a ti jsou ovládáni zbylými hráči. Hra je u konce, pakliže je Pan X úspěšně chycen policisty (vyhrávají policisté), nebo když je dosažen maximální počet kol, bez toho aniž by byl Pan X chycen (vyhrává Pan X). Originální hra se odehrává na mapě Londýna.



Obrázek 2.1: Ukázka herní mapy hry Scotland Yard. Zdroj [14]:

Na této herní mapě nachází 200 polí, které jsou vzájemně propojené cestami. Každá z těchto cest povoluje pouze určitý způsob dopravy (např. pouze taxíkem, pouze autobusem atd.). Jednotliví hráči využívají prvky veřejné dopravy k pohybu po herní ploše, kterými jsou:

- *Taxi*
- *Autobus*
- *Metro*
- *Trajekt*

Každému hráči je na začátku hry přidělen pouze určitý počet jízdenek na tyto typy dopravních prostředků. K využití dopravy je použita právě tato jízdenka. Pokud již hráč nemá některou z typů jízdenek, nemůže nadále tento způsob přepravy využívat. Hra se dělí na kola, ve kterých se hráči postupně střídají. Jakmile Pan X provede svůj tah, je odhaleno pouze jaký typ dopravy využil, ale neodhaluje na které místo se přemístil.

Hlavní myšlenkou hry je, že po většinu kol je pozice Pana X policistům utajena. Odhaluje se jim pouze určená kola. To znamená, že policisté pro polapení Pana X musí odhadovat jeho pozici a spolupracovat mezi sebou. Tímto se ze hry Scotland Yard stává hra s nedokonalou informací, jelikož policisté nevidí přesnou pozici Pana X. Tento fakt ji činí vhodnou pro studování metod hraní her s neurčitostí.

Pan X má také možnost použít speciální akci, kterou je *dvojitý tah*. Také vlastní speciální jízdenku, která mu umožňuje použít jakýkoliv druh dopravy Pokud Pan X použije tutu speciální jízdenku, není policistům odhalen jaký typ dopravy využil.

2.2 Další hry s nedokonalou informací

V oblasti umělé inteligence hraje důležitou roli modelování a řešení her. Hry představují formalizaci konfliktních interakcí mezi aktéry. Klasická teorie her se zaměřuje na hry s úplnou informací, kde mají všechny strany v daném okamžiku přístup ke všem relevantním informacím z herního prostředí. Znají strategie a cíle ostatních hráčů. V praxi se však častěji setkáváme se situacemi, kde jednotlivým stranám chybí některé informace, ať už se jedná o informace z prostředí či cíli soupeře.

2.2.1 Bayesovské hry

Tyto případy lze modelovat pomocí her s neúplnou či nedokonalou informací, kde hráči nemají úplné znalosti o prostředí nebo soupeřích. Neúplnou informaci můžeme sledovat například u Bayesovských her.

Definice 1 (Bayesovská hra) *[15] je definována pětic*í $(N, A_i, \theta_i, p(\theta_i), u_i)$*, kde:*

- N *je konečná množina hráčů,* $N = \{1, 2, \ldots, n\}$.
- A_i *je neprázdná množina strategií hráče* i.
- θ_i *je neprázdná množina typů hráče* i.
- $p(\theta_i)$ *je apriorní pravděpodobnostní rozdělení typu hráče* i *na* θ_i.
- $u_i : A_1 \times \cdots \times A_n \times \theta_1 \times \cdots \times \theta_n \to \mathbb{R}$ *je výplatní funkce hráče* i.

Bayesovské hry představují formální rámec sloužící k modelování a popisu her s neúplnou informací.

2.2.2 Stratego

Stratego je desková strategická hra pro dva hráče. Vychází z dřívějších her, jako jsou Šachy a Go. Kombinuje strategické plánování se snahou obelstít soupeře. Hra se odehrává na herní ploše o velikosti 10x10 polí. Na herním poli jsou umístněny sekce s různými terény, které znemožňují pohyb jednotek. To vytváří různé strategicky uzká místa, kde je možno zabarikádovat protivníka jen pomocí dvou figur. Tím se hra stává zajímavější a je možné tyto body využívat k donucení oponenta aby zaútočil jako první.

Na této herní mapě má každý hráč svoji sadu figur reprezentujících armádu. Figury jsou ale vyobrazeny tak, že jejich hodnost lze přečíst pouze z jedné strany. Tím druhý hráč nezná, rozestavění protivníkovy armády. Každý hráč má 40 figur, rozdělených do 11 hodností (generál, plukovník, skaut atd.).

Cílem hry je porazit soupeře nalezením a obsazením jeho vlajky. S vlajkou, stejně tak s minou nemohou hráči po umístnění pohnout. Tyto speciální miny mohou zničit jakoukoliv figuru se kterou se utkají v boji. Minu může zničit pouze horník.

Hra začíná tím, že každý hráč rozmístí své figury na herní pole. Toto rozmístnění je samozřejmě tajné. Hráči se střídají v tazích, kdy se pokouší najít oponentovu vlajku. Pokud hráč táhne na pole, kde se nachází oponentova figura, nastává souboj. Souboj spočívá v odkrytí hodnosti obou figur a vyhrává ta s vyšší hodností. Figura, která vyhrála zůstává, poražená figura je odstraněna z hry.

Ve hře Stratego je důležité blafování a odhadování soupeřových tahů. V této hře velmi důležitá informace, tudíž je důležité informace získávat. To můžou hráči učinit například obětováním slabších figur, aby odhalily soupeřovy figury se silnější hodností. I tato hra je modelovatelná metodou Bayesovských her [22].

Obrázek 2.2: Ukázka rozestavěných figur ve hře Stratego. Zdroj [10]:

Z pohledu umělé inteligence je Stratego zajímavý problém. Nejenže je hra s nedokonalou informací a je tedy zapotřebí odhadovat oponentovy tahy a blafovat. Hra ale také má ob-

rovský stavový prostor 10^{535} [19], čímž je prakticky neřešitelná klasickými metodami, které prochází celý stavový prostor. Až do roku 2022 nebyla umělá inteligence v této hře velmi úspěšná a nedokázala se rovnat expertním hráčům. To se změnilo příchodem *DeepNash* [19], kdy se tento agent umístil mezi třemi nejlepšími hráči světa.

2.2.3 Dota 2

Dota 2 je velmi komplexní strategická hra v odehrávající se v reálném čase. Hraje se ve dvou týmech po pěti hráčích. Každý tým má svoji základnu, kterou se snaží chránit a zároveň zničit základnu soupeře. Hra je u konce v okamžiku, kdy je zničena základna jednoho z týmů a tento tým prohrává. Herní mapa se skládá ze tří hlavních cest, které jsou spolu navzájem propojeny mnoha menšími stezkami.

V každém týmu hraje 5 hráčů, kteří ovládají postavy s různými unikátními schopnostmi. Hráči získávají zkušenosti jak za porážení nepřátelských postav, tak za zabití jednotek ovládaných počítačem. Zkušenosti se využívají k zvyšování úrovně postavy a odemykání nových schopností. Schopnosti jsou hlavní součástí hry. Díky schopnostem mohou hrdinové zranit nepřítele, udělat silnější sebe či své spojence a mohou mít i spousty dalších efektů. Co jim navíc přidává na komplexnosti je to, že každá schopnost nemůže být po použití znovu použita po určitou dobu. Zlato používají hráči jako herní měnu, za kterou si hrdinové mohou koupit různé předměty, které je posílí.

Ve hře se vyskytuje válečná mlha, která způsobuje že hráči vidí pouze část mapy ve svém okolí a okolí spřátelených jednotek. To znamená, že hráči nemají úplnou informaci o pozici nepřátelských jednotek.

Dota 2 je pro umělou inteligenci náročnou výzvou z těchto důvodů:

- Složitost hry

  Dota 2 je komplexní hra s mnoha proměnnými a možnostmi. Její stav se mění velmi rychle a je tedy těžké předpovědět vývoj hry. Agenti tedy musí být schopni rychle reagovat na změny a tím přizpůsobit svoji strategii.

- Nedokonalá informace

  Právě kvůli válečné mlze nemají hráči úplnou informaci o pozici nepřátelských jednotek a musí tedy pouze odhadovat kde na mapě se nachází. Neznají ani jestli jsou nepřátelé schopni použít své schopnosti a předměty. Ve hře dota 2 vykonávají hráči několik rozhodnutí někdy i několikrát za sekundu. Je to doopravdy dynamická hre, kde je třeba rychle reagovat na strategie soupeřů, rychle a správně tyto strategie odhadovat a vytvářet úplně nové strategie.

Dota 2 sama o sobě obsahuje implementaci chování inteligentních agentů, kteří jsou schopni hrát hru na velmi vysoké úrovni. Jejich výkon však není dostatečný a nebyli schopni ani zdaleka porazit nejlepší hráče světa. Použité metody pro vytvoření těchto agentů bohužel nejsou zveřejněny.

Inovaci umělé inteligence v této hře opět přinesla společnost OpenAI a jejich projekt *OpenAI Five* [17]. Tento projekt se zaměřil na vytvoření modelu, který by byl schopen porazit nejlepší hráče světa ve hře Dota 2. OpenAi nepřišel se žádným novým algoritmem, ale pouze s novým přístupem k trénování a využití algoritmu **PPO** v doposud neviděném měřítku. Algoritmus PPO a posilované učení bodou blíže popsáni v následujících sekcích 2.6 a 2.3.

OpenAi Five porazil tehdejší nejlepší tým světa během exhibice na nejprestižnějším turnaji Dota2 *The International*. Hrála se klasická turnajová verze, a to tedy nejlepší ze tří. OpenAI Five byl schopen porazit nejlepší tým světa s rozhodným výsledkem 2:0. První hra trvala 38 minut a druhá již pouhých 21 minut. V celém exhibičním turnaji proti nejlepším týmům světa z celkových 24 her vyhrál 19 her a pouhých 5 prohrál. Bylo zajímavé sledovat, že v průběhu zápasu OpenAI Five veřejně vypisoval své odhady pravděpodobnosti výhry.

Agent OpenAI Five byl, kombinovaně s předchozí verzí, trénován asi *55 000* herních let hraním sama proti sobě. Pozorovací prostor je obrovský, obsahuje až 16 000 vstupů. Akční prostor je složitý, jelikož lidé hrají Dotu 2 většinou pomocí myši a klávesnice. U OpenAI Five je akční prostor rozdělen na hlavní akce a parametry těchto akcí. Hlavní akce jsou například, pohyb, útok, použití schopnosti/ předmětu, zakoupení předmětu a další situační akce. Dostupnost těchto akcí je závislá na stavu hry a je řízena maskou akcí. Tyto akce mají 3 parametry, zpoždění, vybranou jednotku a offset cíle akce. Pokud tyto akce a jejich parametry zkombinujeme získáme akční prostor o velikosti 1,837,080 dimenzí. Výsledný reakční čas agenta je okolo 33ms, oproti tomu reakční čas profesionálních hráčů her je okolo 120ms [13]

Na začátku byla většina chování agenta uměle naprogramována a postupně mu byla předávána kontrola nad vlastním chováním s využití strategie. Některé naskriptované chování však bylo zachováno i ve finální verzi OpenAi Five a nebyly odstraněny, jelikož již agent byl dostatečně efektivní a nebylo třeba je odstranit.

2.3 Klíčové koncepty posilovaného učení

Posilované učení (Reinforcement Learning, RL) je oblast strojového učení, která se zaměřuje na učení agentů v dynamickém prostředí. Agent se učí strategii chování, která maximalizuje kumulativní odměnu.

2.3.1 Agent

Je komplexní entita, která interaguje s prostředím. Prostředí poskytuje agentovi informace o stavu a agent na základě těchto pozorování vykonává akce. Tyto akce mohou ovlivnit stav prostředí a agent obdrží odměnu na základě odměnové funkce. Agent většinou volí takové akce, aby maximalizoval kumulativní odměnu.

2.3.2 Pozorovací a akční prostor

Pozorovací prostor je jednou z nejdůležitějších součástí posilovaného učení. Bez jeho vhodného zvolení, je dosažení strategie s dobrými výsledky velmi obtížné. Pozorovací prostor je množina všech možných pozorování, které může agent získat z prostředí. Je to tedy forma, kterou prostředí předává informace agentovi. Velmi často se aktuální stav pozorovacího prostoru využívá jako stav. Při vytváření prostředí je nutné definovat typ, tvar prostoru a jakých hodnot může nabývat.

Pro jeho podoby je vhodné zvolit standard z knihovny `Gymnasium` [9]. Gymnasium je pokračování knihovny `Gym` od společnosti OpenAI. Tyto knihovny nastavily standard vytváření prostředí pro posilované učení. Dodržování těchto standardů umožňuje použití libovolného algoritmu na jedno prostředí.

Akční prostor je, jak název vypovídá, množina všech možných akcí, ze kterých může strategie volit. Často se jedná o konečnou množinu celočíselných hodnot, které reprezentují

různé akce. Samozřejmě může být akční prostor i spojitého typu (blíže popsáno v další sekci).

V herním prostředí mohou nastat situace kdy se nějaká akce stane nevalidní, jelikož by její provedení vyústilo v nevalidní stav herního prostředí. V takovém případě je vhodné zvolení této akce zamezit, nejlépe pomocí akční masky. Ta definuje, které akce jsou v momentálním stavu prostředí proveditelné. Lze také tyto akce povolit, ale následně agentovi udělit sníženou odměnu za provedení nevalidní akce.

2.3.3 Diskrétní a spojitý prostor

Pokud se bavíme o **diskrétním prostoru** v oboru umělé inteligence, bavíme se o konečném prostoru. Skládá se tedy z konečného počtu prvků. Příkladem takového prostoru je například pozorovací prostor, jehož vstupem je fotka. Ta se dělí na přesně daný počet pixelů a je tedy diskrétní. Diskrétní akční prostory jsou počítatelné a často se jedná o konečnou množinu možných akcí.

Zato **spojitý prostor** je nekonečný prostor, který může obsahovat nekonečný počet prvků. Spojitý pozorovací prostor je například ten, jehož vstupem je audio stopa. Spojité akční prostory jsou nekonečné a často je definuje nekonečná množina akcí. Na prostředí se spojitými prostory se často používají jiné algoritmy, například algoritmus DDPG (Hluboký deterministický gradient strategie).

2.3.4 Prostředí

Je vše s čím agent interaguje. Prostředí je buď fyzické (entity z reálného světa, ovládání chytré domácnosti, robotické ruky, ovládání reaktoru apod.) nebo virtuální (simulace nebo hra). Prostředí reaguje na zvolené akce agenta poskytuje mu zpětnou vazbu ve formě odměny. Odměna může být i záporná, pokud agent zvolil velmi nevhodnou akci. Pokud v prostředí existuje více agentů, může mít každý agent jiné pozorování. Díky tomuto můžeme například schovat agentovi A určité informace, které agent B vidí.

2.3.5 Strategie (Policy)

Pomocí posilovaného učení vzniká strategie. Strategie je matematická funkce, která definuje agentovo chování na základě jeho pozorování (stavu). Snaží se definovat takové chování, které vede k maximální kumulativní odměně. Strategie může být deterministická nebo stochastická.

Deterministická strategie

Deterministická strategie přesně definuje cílový stav přechodu pro každý stav. Agent tedy pro jeden stav vždy volí stejnou akci. Tato strategie je vhodná, pokud je zapotřebí v každém stavu reagovat konzistentně, bez odchylek. Například, pokud agent ovládá termostat v domě a teplota je pod požadovanou hladinu. Nemůže se stát, aby byla šance, že agent zvolí akci, která teplotu ještě sníží. Další výhoda, je že je jednoduchá na interpretaci a implementaci [6].

Rovnice výsledné akce deterministické strategie je:

$$\pi(s) = a \tag{2.1}$$

Stochastická strategie

Zato stochastická strategie definuje pro každý stav pravděpodobnostní rozdělení nad množinou akcí. Výsledná akce je tedy náhodná dle rozdělení pravděpodobnosti. Může tedy nastat situace kdy ve stejném stavu agent zvolí vždy jinou akci. Tato strategie je vhodná v situacích, kdy je potřeba zkoumat různé strategie a kdy agent nemá úplnou informaci o prostředí. Například tam kde by deterministická strategie zvolila jasnou akci A, stochastická strategie by mohla s malou pravděpodobností zvolit akci B. Čímž ale může odhalit, že akce B je s ohledem na komulativní odměnu lepší než akce A [6].

Rovnice výsledné akce stochastické strategie je:

$$\pi(a|s) = \mathbb{P}_\pi[A = a|S = s] \tag{2.2}$$

2.3.6 Akce

Akce je aktivita, proces či funkce kterou agent vykoná ve specifickém stavu [8]. Výsledkem provedení akce je tedy změna z aktuálního stavu, do jiného, či stejného stavu z množiny možných stavů. Zjednodušeně, je to rozhodnutí, které agent vykonává v prostředí a toto rozhodnutí ovlivňuje prostředí.

Akce může být také nedeterministická či stochastická. Výsledný stav tedy může být jiný pro stejný stav a akci.

2.3.7 Odměna

Odměna je hodnota, kterou agent obdrží od prostředí po vybrání akce. Může být kladná, záporná nebo nulová. Dle této zpětné vazby se agent učí, jak moc byla jeho zvolená akce v daném stavu vhodná.

2.3.8 Hodnotová funkce

Hodnotová funkce vyhodnocuje, jak dobrý je stav tím, že predikuje budoucí odměnu, pokud bude další tah začínat v tomto stavu. Čím vzdálenější odměna je od tohoto stavu, tím více je snížena. Protože, čím je odměna vzdálenější tím více je nejisté, že bude doopravdy získána.

Existují dva typy hodnotových funkcí:

Hodnotová funkce stavu $V(s)$

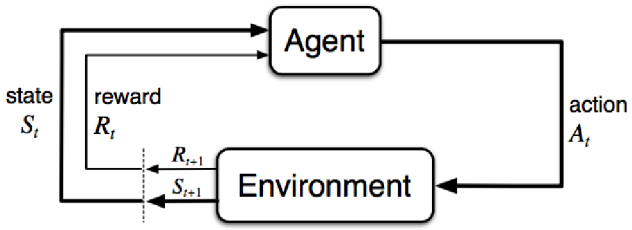
Hodnotová funkce stavu $V(s)$ vyhodnocuje očekávanou komulativní odměnu, jestliže se agent nachází v tomto stavu. Tato funkce je závislá na strategii, kterou se agent řídí. Vyhodnocuje tedy jak příznivý je daný stav pro agenta.

Hodnotová funkce akce $Q(s, a)$

Hodnotová funkce akce $Q(s, a)$ vyhodnocuje očekávanou komulativní odměnu, pokud se agent nachází v tomto stavu a zvolí tuto akci. Tato funkce je opět závislá na strategii, kterou se agent řídí. Vyhodnocuje tedy jak příznivé je zvolení dané akce v aktuálním stavu.

2.3.9 Markovský rozhodovací proces

Téměř všechny problémy, řešené posilovaným učením, mohou být označeny jako Markovy rozhodovací procesy (Markov Decision Process). Tato abstrakce je základním kamenem pro modelování algoritmů posilovaného učení. Markovský rozhodovací proces značí, že následující stav není závislý na stavech minulých, nýbrž pouze na aktuálním stavu.



Obrázek 2.3: Interakce mezi prostředím a agentem podle Markova rozhodovacího procesu. Zdroj [25]:

Definice 2 *Markovský rozhodovací proces je definován pěticí* (S, A, P, R, γ) *[25], kde:*

- S *je množina stavů.*
- A *je množina akcí.*
- P *je pravděpodobnostní přechodová funkce*
- R *je odměnová funkce*
- γ *je zlevňovací faktor pro budoucí odměny*

2.3.10 Bellmanovy rovnice

Belmanova rovnice se zaměřuje na rozložení hodnotových funkcí na menší a snadněji zpracovatelné celky. Dociluje toho tak, že rozděluje hodnotovou funkci na dvě části: *okamžitou odměnu* a postupně snižovanou *budoucí odměnu* [3].

$$V(s) = E[R_{t+1} + \gamma V(S_{t+1}) | S_t = s] \tag{2.3}$$

Tato rovnice vyjadřuje hodnotu stavu s jako očekávanou budoucí odměnu, kterou lze získat, začneme–li v tomto stavu a budeme se řídit optimální strategií [25, 3].

$$Q(s, a) = E[R_{t+1} + \gamma E_{a \sim \pi} Q(S_{t+1}, a) | S_t = s, A_t = a] \tag{2.4}$$

Tato rovnice vyjadřuje hodnotu akce a ve stavu s jako očekávanou budoucí odměnu, kterou lze získat, začneme–li v tomto stavu, zvolíme akci a a poté se budeme řídit optimální strategií [25, 3].

- $V(s)$ je hodnota stavu s.
- $Q(s, a)$ je hodnota akce a ve stavu s.
- R_{t+1} je okamžitá odměna za provedení akce a ve stavu s.

- S_t je stav v čase t.
- A_t je akce v čase t.
- π je strategie.

Tyto rovnice jsou základem pro většinu algoritmů posilovaného učení [25, 3].

2.3.11 Bellanovy rovnice optimality

$$Q_*(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} \max_{a' \in \mathcal{A}} Q_*(s', a') \tag{2.5}$$

Tato rovnice udává, že hodnota akce a ve stavu s při optimální strategii je rovna okamžité odměně za provedení akce a ve stavu s plus zlevněné hodnotě nejlepší akce ve stavu s' při optimální strategii [25, 3].

$$V_*(s) = \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P^a_{ss'} V_*(s') \right) \tag{2.6}$$

Rovnice 2.4 udává, že maximální hodnota stavu s při optimální strategii je rovna maximální hodnotě akce a ve stavu s, pokud dále budeme pokračovat v optimální strategii [25].

- $Q_*(s,a)$ je nejlepší možná hodnota akce a ve stavu s při optimální strategii.[25]
- $V_*(s)$ je nejlepší možná hodnota stavu s při optimální strategii.
- $R(s,a)$ je okamžitá odměna za provedení akce a ve stavu s.
- $P^a_{ss'}$ je pravděpodobnost přechodu ze stavu s do stavu s' po provedení akce a.
- γ je zlevňovací faktor pro budoucí odměny.
- $\mathcal{S}$ je množina stavů.
- $\mathcal{A}$ je množina akcí.
- $\max_{a' \in \mathcal{A}} Q_*(s', a')$ je maximální hodnota akce ve stavu s' při optimální strategii.

2.3.12 Rovnováha mezi explorací a exploatací

> *Kompromis mezi potřebou získávat nové znalosti a potřebou použít již nabyté znalosti k vylepšení výkonnosti je jedním z nejzákladnějších kompromisů v přírodě [4].*

Explorace a exploatace jsou dvě protichůdné strategie, které se vyskytují jak ve strojovém učení, tak i v reálném životě.

Exploatace se snaží vybrat nejlepší možnou akci na bázi známých informací. Tyto informace, nemusí být kompletní, nebo mohou být zavádějící. A to z důvodu nedostatečného trénování, či nedostatečného prozkoumávání možností prostředí.

Tomu opačná metoda *explorace* usiluje o prozkoumání možností, které nejsou známé a mohly by vést k lepší budoucí odměně. Explorace tedy často zvolí akci, která nemusí být nejlepší, ale může odhalit nové informace, které následovně povedou ke zlepšení exploatace.

2.3.13 Druhy informací v teorii her

V rámci umělé inteligence se potýkáme s různými druhy informací. Dělí se na tyto hlavní typy:

Dokonalá informace znamená, že agent ví o prostředí a o ostatních hráčích vše. Například ve hře šachy. Hráč vidí všechny figury na herní ploše, i ty soupeřovy.

Kompletní informace značí, že agent je obeznámen se strukturou hry a jsou mu také odhaleny odměnové funkce ostatních hráčů. Hráč tedy ví, jakou hru hraje je obeznámen s jejími pravidly. A rozumí, jaké jsou podmínky výhry a je obeznámen s taktikou ostatních hráčů.

Nedokonalá informace znamená, že agent nemá všechny relevantní informace o prostředí a ostatních hráčích. Například, tedy všechny hry, ve kterých hrají hráči zároveň jsou hry s nedokonalou informací. Jelikož hráč v daném okamžiku nezná informaci o tahu ostatních hráčů. Další příklad je například hra poker, kde hráč nezná rozdané karty ostatních hráčů. Také hra Scotland Yard, kde policisté neznají pozici Pana X.

Neúplná informace znamená, že hráč nezná strukturu odměn, podstatu hry nebo její pravidla. Hráč tedy nezná výchozí informace o hře. Všechny hry s neúplnou informací se dají považovat za hry s nedokonalou informací.

Soukromá informace je informace, která není dostupná ostatním hráčům.
Společná informace je informace, která je dostupná všem hráčům.

2.4 Vhodné algoritmy k řešení her s nedokonalou informací

Tato kapitola se zaměřuje na algoritmy, které jsou vhodné pro řešení her s nedokonalou informací, s důrazem na metody posilovaného učení a srovnáním s klasickými metodami jako Monte Carlo.

2.4.1 Monte Carlo tree search

Metoda Monte Carlo tree search (MCTS) je heuristický algoritmus prohledávání. Kombinuje stromové vyhledávání s principy posilovaného učení. Je často využíván, je–li stavový prostor řešeného problému příliš velký a složitý na to, aby byl prohledán kompletně jinými metodami, jako například minimax, či alfa–beta prořezávání. Tyto „tradiční“ algoritmy nelze na mnoho problémů použít, jelikož by byly příliš pomalé a náročné na výpočet.

Tato metoda se také potýká s rovnováhou mezi explorací a exploatací (viz. sekce Rovnováha mezi explorací a exploatací). Explorací se strom rozrůstá do šířky, zatímco exploatací se strom prohlubuje [20].

MCTS se skládá z několika fází:

- *Selekce*

  Na základě aktuálního stavu se vybere další stav k prozkoumání. Pro tento výběr se využívají dvě strategie:

  Strom s horní mezí spolehlivosti (Upper confidence bounds applied to trees, UCT) kombinuje průměrnou hodnotu uzlu a odměnu za exploraci.

  Chamtivá strategie ϵ (ϵ–greedy strategy) vybírá s pravděpodobností ϵ náhodný uzel, jinak volí uzel s nejvyšší hodnotou. Tato strategie se používá méně často než UCT.

  Obě tyto strategie se snaží o rovnováhu mezi explorací a exploatací.

- ***Expanze***

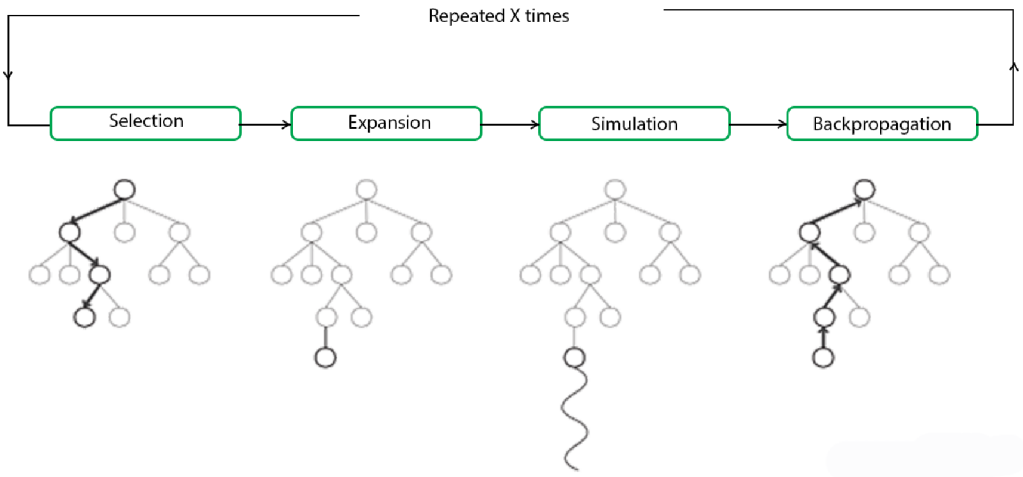
  V tomto kroku se vyhledávací strom rozšíří o nový uzel, který je výstupem z předchozího kroku.

- ***Simulace***

  Po této fázi je provedena náhodná simulace od nového uzlu až do konečného stavu.

- ***Aktualizace***

  Díky nově nabitým informacím ze simulace, se zpětnou propagací aktualizují hodnoty uzlů ve stromě.



Obrázek 2.4: Diagram jednotlivých fází MCTS. Zdroj: [20]

Tento algoritmus se skvěle hodí na hry s nedokonalou či neúplnou informací, jelikož se spoléhá na vzorkování pomocí simulací.

2.4.2 Q-learning

Q-learning je jedním z nejznámějších algoritmů posilovaného učení. Učení tohoto algoritmu probíhá bez modelu a mimo strategii.

Jak už název vypovídá, Q-learning se zaměřuje na určení hodnoty hodnotové funkce akce $Q(s, a)$, viz. 2.3.8. Tyto hodnoty se uchovávají v Q–tabulce, která na každou kombinaci stavu a akce uchovává hodnotu $Q(s, a)$. Hodnoty v Q–tabulce se iterativně aktualizují na základě získaných odměn. Kladná odměna po vykonání akce s, zvýší se hodnota $Q(s, a)$. Naopak, dostane–li po vykonání této akce zápornou odměnu, hodnota $Q(s, a)$ se sníží.

Řádky Q-tabulky tedy reprezentují stavy a sloupce akce. Po vytvoření jsou všechny Q hodnoty v tabulce inicializovány na nulu. Následně jsou tyto hodnoty iterativně aktualizovány dle zpětné vazby udělené od prostředí ve formě odměny.

Jako u většiny algoritmů je zde potřeba dohlédnout na rovnováhu mezi explorací a exploatací. K tomuto se využívá ϵ–greedy strategie viz. 2.4.1.

Hlavní nevýhodou tohoto algoritmu je právě jeho závislost na Q–tabulce. Tabulka mapuje hodnoty pro každou kombinaci stavu a akce, Je–li avšak stavový, či akční prostor příliš velký, nebo dokonce nekonečný je tento algoritmus nevhodný, skoro až nepoužitelný. Výsledná Q–tabulka by byla neefektivní kvůli své velikosti. Mohlo by se stát, že s nekonečným množstvím kombinací by velikost tabulky rostla do „nekonečna“. Jako řešení byl navrhnut algoritmus Deep Q-learning.

2.4.3 Deep Q-learning (DQN)

Tato metoda je rozšířením algoritmu Q-learning které, nahrazuje Q–tabulku neuronovou sítí určené k aproximaci hodnotové funkce akce $Q(s, a)$. Díky této aproximaci je možné použít tento algoritmus na problémy s velkým, či nekonečným množstvím kombinací akcí a stavů.

Avšak tímto vzniká nový problém, *nestabilita učení*. Ten je řešen dvěma mechanizmy:

- **Přehrání zkušenosti (experience replay)**

  Během trénování se ukládají všechny zkušenosti do paměti. Ať už to jsou akce, stavy, odměny atd. Při trénování se poté náhodně vybírají náhodně zkušenosti z této paměti a aktualizují se podle nich váhy sítě. Tímto se snižuje rozdíl mezi jednotlivými aktualizacemi a tím pádem se zvyšuje stabilita učení.

- **Periodická aktualizace** Síť je naklonována a změny se provádí pouze na duplikátní verzi. Do hlavní originální sítě se změny klonují po určitém počtu kroků.

Při vytváření neuronové sítě jsou váhy inicializovány náhodně.

2.4.4 Gradient strategie

Oproti předchozím zmiňovaným algoritmům, které usilují o naučení hodnotové funkce či prohledávají stavový prostor, algoritmy gradientu strategie se snaží naučit strategii přímo.

V diskrétním prostoru je odměnová funkce definována jako:

$$\mathcal{J}(\theta) = V_{\pi_\theta}(S_1) = \mathbb{E}_{\pi_\theta}[V_1] \tag{2.7}$$

Ve spojitém prostoru je odměnová funkce definována jako:

$$\mathcal{J}(\theta) = \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) V_{\pi_\theta}(s) = \sum_{s \in \mathcal{S}} \Big(d_{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi(a|s, \theta) Q_\pi(s, a) \Big) \tag{2.8}$$

- $\mathcal{J}(\theta)$ je odměnová funkce strategie π_θ.
- θ je parametr strategie π_θ.
- $V_{\pi_\theta}(S_1)$ je hodnota stavu S_1 dle strategie π_θ.
- $\mathbb{E}_{\pi_\theta}[V_1]$ je očekávaná hodnota stavu S_1 dle strategie π_θ.
- $d_{\pi_\theta}(s)$ je rozdělení pravděpodobnosti stavů s dle strategie π_θ.

Věta o gradientu strategie

Výpočet gradientu matematicky je zaznamenán touto rovnicí:

$$\frac{\partial \mathcal{J}(\theta)}{\partial \theta_k} \approx \frac{\mathcal{J}(\theta + \epsilon u_k) - \mathcal{J}(\theta)}{\epsilon} \tag{2.9}$$

Tento výpočet je velmi pomalý a náročný na výpočet. Avšak tento vzorec lze zjednodušit na rovnici zvanou *věta o gradientu strategie*:

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s,\theta) Q_\pi(s,a)] \tag{2.10}$$

2.5 Trust Region Policy Optimization (TRPO)

Trénované strategie jsou často velmi náchylné na změny. Kde i malá náhlá změna v jednom kroku může způsobit velké změny v chování agenta a zamezit dalšímu učení správné strategie. Během učení totiž chceme, aby učení probýhalo plynule a ne skokově. Pokud se strategie změní příliš rychle protože následovala nejstrmější směr růstu, může se stát, že mine cestu vedoucí k optimální strategii. Může tedy uváznout v lokálním optimu.

TRPO je optimalizační algoritmus, který se snaží tento problém řešit tím, že definuje omezení rozdílu mezi novou aktualizovanou strategií **p** a starou strategií z předešlého kroku **q**. Tento rozdíl mezi dvěma pravděpodobnostními rozděleními je definován jako Kullback-Leiblerova divergence (K-L divergence)[24]. Vzorec pro K-L divergenci ve spojitém prostoru je následující:

$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx \tag{2.11}$$

Vzorec v diskrétním prostoru:

$$D_{KL}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{2.12}$$

Vzniká tak region důvěry, ve kterém musí nová strategie setrvat. Strategie se tedy nemůže skokově změnit na lokální maximum, ale pouze následuje směr lokálního maxima.

K tomu využívá Hesenská matice. Kdy pomocí těchto matic hledá optimální směr změny strategie. Tím, že se hledá lokální maximum pouze v oblasti důvěry, řeší problém se zaseknutím v těchto lokálních optimech. Hesenská matice je vždy čtvercového tvaru. Její hodnoty obsahují druhé parciální derivace funkce a popisuje tím její lokální zakřivení. Jelikož je strategie pravděpodobnostní funkce, jsou tyto matice ideální pro hledání optimálního směru změny strategie.

Díky tomuto řešení se učení stává robustním a stabilním. Avšak výměnou za to, je náročnost aproximace pomocí Hesenských matic a s tím související náročnost na implementaci. Výpočet buňky v Hesenské matici je definován jako:

$$H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} \tag{2.13}$$

Kde i značí řádek a j sloupec matice. Výsledná matice tak poté může vypadat takto:

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{bmatrix} \tag{2.14}$$

2.6 Proximální optimalizace strategie (PPO)

Metoda proximální optimalizace strategie (Proximal policy optimization - PPO) byla představena roku 2017 [21]. Tato metoda je vylepšeným následníkem algoritmu TRPO, popsaný v předchozí sekci 2.5.

Co výrazné odlišuje tuto metodu od předchůdců, je její jednoduchost a efektivita. Dosahuje lepších výsledků než metoda TRPO a srovnatelných výsledků s metodou ACER. Je avšak mnohem jednodušší na implementaci, má nižší nároky na výkon a je mnohem lepší co se týče efektivity dat pro trénování [21]. K dosažení těchto výsledků, využívá PPO techniku ořezávání náhradních cílů vloženou do algoritmu TRPO. Vynucuje, aby K-L divergence staré a nové strategie byla v rozmezí $[1-\epsilon, 1+\epsilon]$ kde ϵ je modifikovatelný parametr.

Nechť $r_t(\theta)$ je pravděpodobnostní poměr mezi novou a starou strategií.

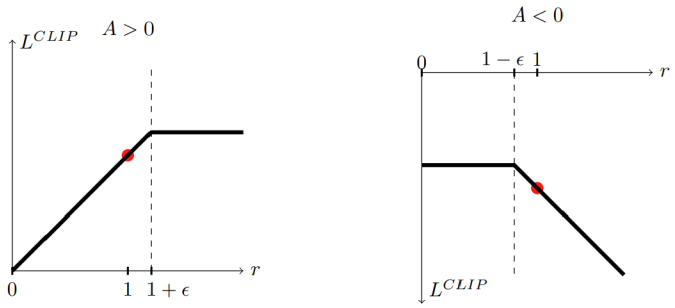
$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2.15}$$

Pokud tento vzorec dosadíme do rovnice pro výpočet gradientu strategie TRPO. Kde TRPO využívá K-L divergenci a maximalizuje vedlejší cíle [26], vznikne následnovná rovnice:

$$L^{CLIP}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big] \tag{2.16}$$

$$L^{CLIP}(\theta) = \mathbb{E}_t\Big[\min\Big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big] \tag{2.17}$$

- θ je parametr strategie.
- $\mathbb{E}_t$ značí empirickou očekávání v časových krocích
- r_t relativní podobnost nové a staré strategie [21]
- $\hat{A}_t$ je odhad výhody akce v čase t.
- ϵ je hyperparametr, obvykle nastaven na hodnotu 0.1 nebo 0.2.



Obrázek 2.5: Grafy zobrazují 1 časový krok funkce L^{CLIP}. Červeným bodem je označen počáteční stav omptimalizace. Levý graf zobrazuje kladnou změnu a pravý zápornou změnu. Zdroj: [25]

Tyto změny tvoří PPO oproti TRPO kompatibilní s metodou stochastického gradientního sestupu. Odstraňuje K–L omezení a nahrazuje je ořezáváním.

PPO má však i své nedostatky. Dle studie [12] se ukázalo, že PPO nefunguje optimálně za 3 podmínek:

1. V prostředí se spojitým prostorem akcí je nemodifikované PPO nestabilní, pokud odměna náhle zmizí mimo ohraničenou podporu.

2. V diskrétním akčním prostoru s řídkými a vysokými odměnami PPO volí neoptimální akce.

3. V době těsně po inicializaci je náchylné k předčasné volení strategie, pokud je některá z optimálních akcí po inicializaci blízko a snadno dosažitelná.

Tato studie také navrhla tyto řešení a prokázala jejich účinnost. Bod 1 a 3 je řešen buď převedením spojitého akčního prostoru na diskrétní nebo zavedením *Beta parametrizace strategie*. *KL regulováním cíle* je řešen bod 2.

Podmínka číslo 1 je v mé implementaci prostředí hry Scotland Yard implicitně neplatná, jelikož akční prostor je diskrétní.
Podmínka číslo 2 je řešena přídáním menších dílčích odměn. Agenti tak získávají odměny i za menší kroky, nejen například za vítězství/ prohru či blízkost, viz. podsekce Systém odměn.

2.6.1 Experiment OpenAI

Důležitým zdrojem pro tuto práci byla studie [2], od společnosti OpenAI. Z ní vyplývá, že algoritmus je vhodný pro řešení problémů s nedokonalou informací. V dané studii byl algoritmus implementován na komplexní hru typu schovávaná. Ve hře proti sobě hrají dva typy hráčů. Modří hráči se snaží schovat a utéct před červenými. Po herní ploše byly rozestavěny objekty, se kterými hráči mohou interagovat. Mohou je přesouvat a následně „zamrazit“ na místě, takže s objektem nelze pohybovat. Modří hráči se na začátku hry objevují ve své pevnosti, která má několik děr. Červeným hráčům je na začátku hry znemožněn pohyb, což dává modrým hráčům čas připravit se na jejich útok.

Po více než 8 miliónech epizodách učení se modří hráči naučili efektivně blokovat vstup do jejich pevnosti, takže je červení hráči nebyli schopni dostihnout. Červení hráči se poté adatpovali a naučili se využívat rampy a přelézt opevnění modrých hráčů. Jako finální, a ultimátní strategii se modří naučili po 43 miliónech epizodách. Modří hráči na začátku kola ukradli červeným všechny rampy a zabarikádovali se i s nimi v pevnosti. Červení hráči tak neměli žádnou šanci na výhru.

Dokonce se objevili i fascinující strategie které zneužívaly chyby v prostředí. Například, v kódu obsluhující kolize byla chyba, která způsobovala, že při najetí rampy na zeď arény, pod určitým úhlem, byla rampa rapidní rychlostí vymrštěna do vzduchu. Toho červení hráči zneužili a za pomocí této chyby se vymršťovali do vzduchu, aby překonali zdi pevnosti modrých.

Kapitola 3

Zhodnocení současného stavu a plán práce (návrh)

V této kapitole je popsáno navrhnutí postup implementace řešení.

3.1 Zkoumaná modifikovaná verze hry Scotland Yard

Tato práce využívá modifikovanou verzi hry Scotland Yard, ve které se hráči pohybují po mřížkové herní ploše ve tvaru čtverce. Na mřížce se nachází 15×15 polí. Hráčí se po těchto polích mohou pohybovat ortogonálně i diagonálně, vždy o však maximálně 1 pole. Hráč se může rozhodnout nezměnit pozici a zůstat na svém aktuálním poli. K pohybu nejsou potřebné žádné jízdenky.

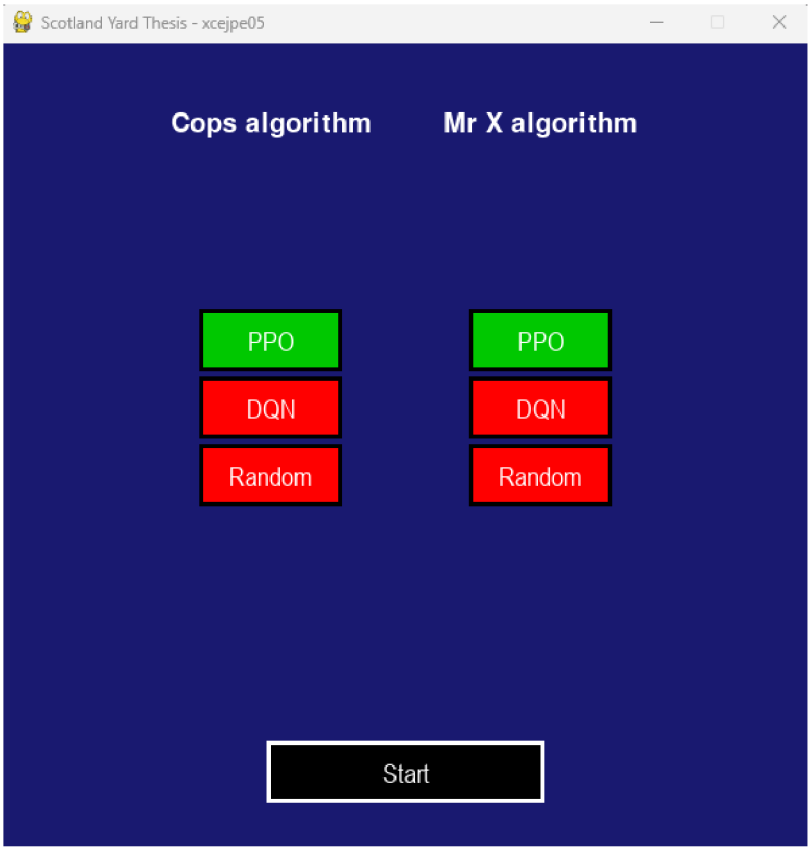
Předtím nežli začne hra, se vyberou náhodné startovací pozice Pana X a policistů. Z těchto možných pozic se následně náhodná pozice přidělí jednotlivým hráčům. Poté začíná hra.

Hra se dělí na jednotlivá kola, ve kterých se hráči ve svých tazích střídají. Pro hru byli zvoleni 3 policisté, z toho důvodu, že je herní pole velké a 2 policisté by nemuseli mít možnost ho celé pokrýt. V kole hraje jako první Pan X a poté policisté, již podle jejich očíslování, které jim bylo náhodně přiděleno při vytváření. Hra končí v okamžiku, kdy policisté chytí Pana X nebo když Pan X zůstane nepolapen až do konce.

Z původní verze hry byly odstraněny některé elementy jako jsou jízdenky a různé druhy dopravních prostředků. Tato úprava vede k tomu, že se zjednodušila strategie agentů, jelikož se nemusí starat o svoje jízdenky a mohou se pohybovat po herní ploše libovolně. Změny, avšak nemění základní podstatu hry, zachovává neurčitost, ale značně zjednodušuje implementaci. Stavový prostor je tedy zjednodušený a tím se i snižují nároky na výkon.

3.2 Implementace uživatelského rozraní a herních mechanizmů

Uživatelské rozhraní bylo vytvářeno pomocí knihovny `Pygame`. Hra začíná v menu, kde je momentálně možné vybrat pouze možnost sledování hry mezi dvěma agenty. Je zde ale možnost vybrat jaký algoritmus rozhodování je použit pro jednotlivé agenty. K dispozici jsou algoritmy *PPO*, *DQN* a náhodné chování.



Obrázek 3.1: Menu hry

Samotný kód hry je rozdělen na 3 částí. Jednotlivé vrstvy hry jsou tak izolovány a mohou být snadno vyměněny za jiné implementace.

- *GameController* [7]

  Tato třída je zodpovědná za řízení hry. Je zde spuštěna univerzální herní smyčka, která zpracovává uživatelské vstupy. A následně provádí aktualizaci stavu aktuální scény a překreslení dané scény.

- *Scény* [7]

  Jednotlivé scény následně definují své chování při aktualizaci a překreslení. Manipulace a přepínání mezi nimi je zajištěno pomocí zásobníků scén. Do tohoto zásobníku se ukládají nově otevřené scény a obsluhována je vždy ta nejnovější.

- *Hra*

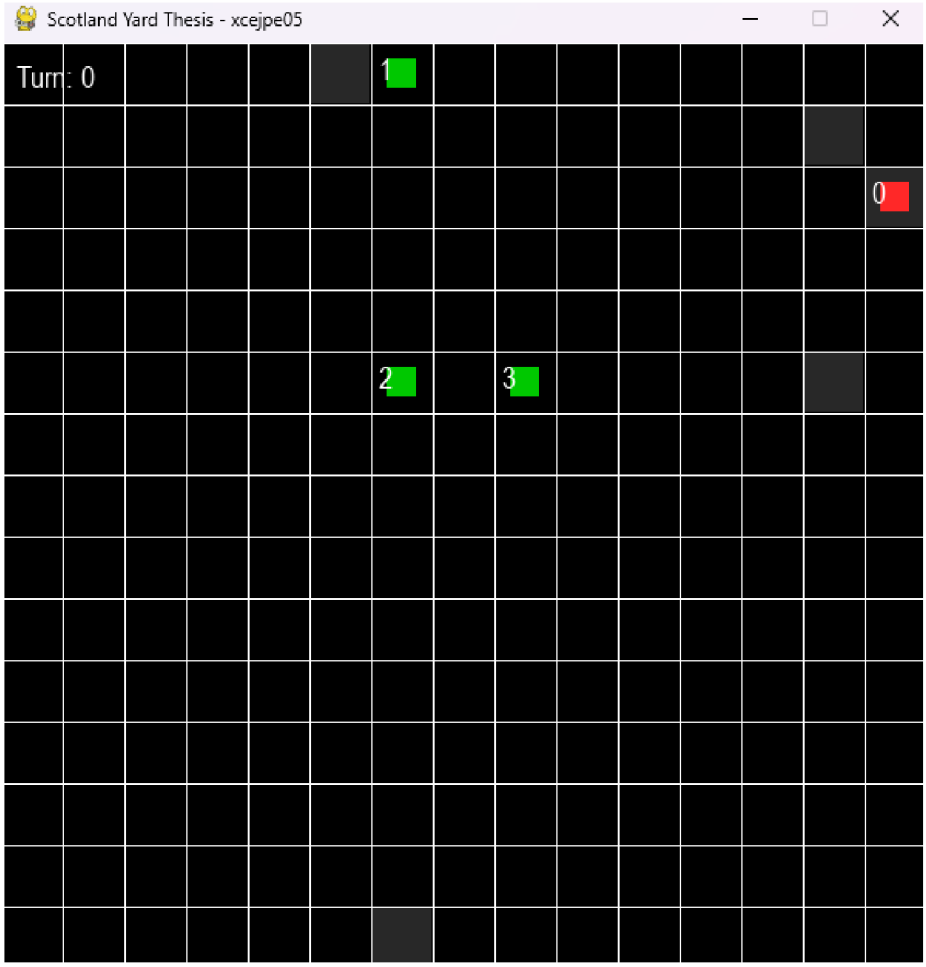
  Samotná hra je následovně rozdělena na 2 další části která každá zpracovává jinou část hry.

  - *Herní logika* — src/game/scotland_yard_game_logic.py Zpracovává herní mechanizmy, jako je pohyb, zpracování výherních podmínek atd. Zprostředkovává informace pro prostředía a to poté pro učící se agenty.
  - *Herní vizualizace* — src/game/scotland_yard_game_visual.py Vykreslování herních elementů (herní pole, figury atd.).

Použití herní smyčky, která obsluhuje aktuální scénu bylo inspirován výukovým projektem z platformy GitHub [7]. Kód byl, avšak značně upraven a vylepšen, aby vyhovoval integrování do této práce.

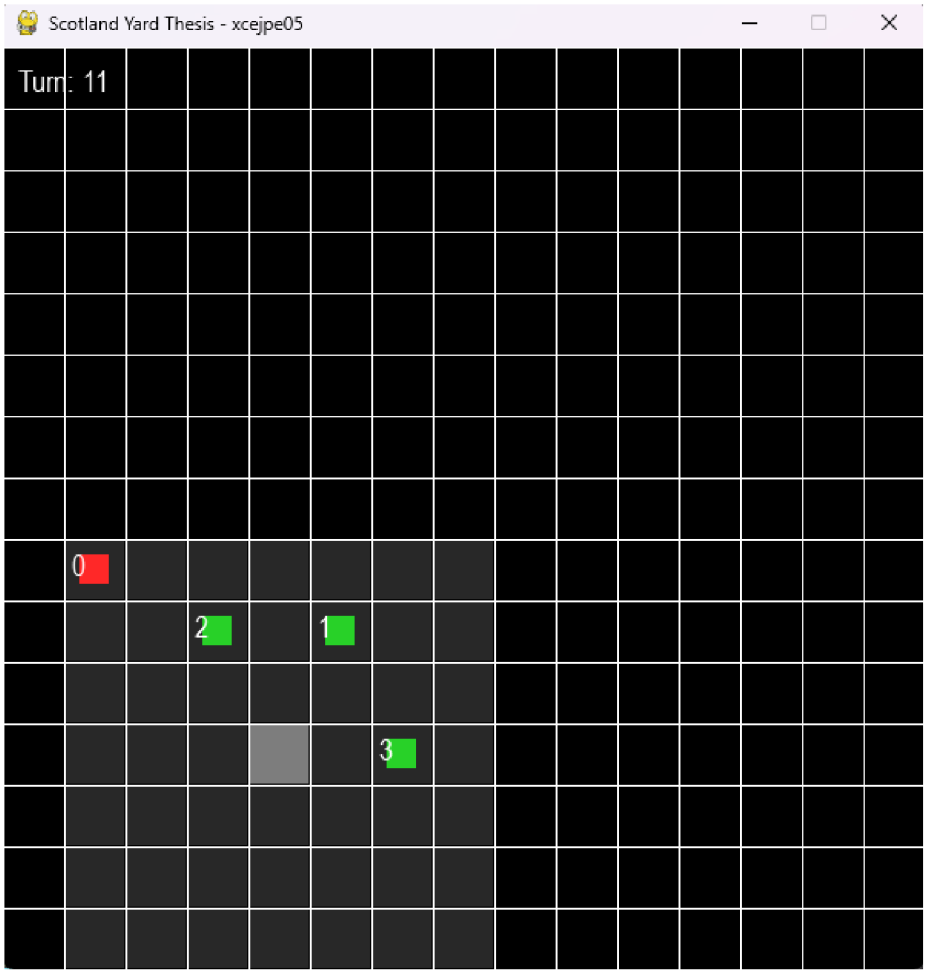
Hra se zapne v pozastaveném stavu, aby bylo možné lépe pozorovat počáteční rozložení hráčů. Pokud je hra pozastavena, je možné pokračovat stisknutím klávesy Space (mezerník). Jakmile hra běží, je možné ji opětovně pozastavit stisknutím klávesy Space (mezerník).

Samotná hra je zobrazena na herní ploše, kde se nachází 3 policisté a Pan X. Je rozdělena do mřížky 15×15 polí. V mřížce jsou zobrazeny 4 figury ve formě barevného čtverce. Červený čtverec značí Pana X a zelené čtverce značí policisty. V levém horním rohu okna je zobrazen aktuální stav hry. Pakliže hra stále běží, je zobrazeno číslo aktálního kola. Pokud hra skončila, je zobrazeno, kdo hru vyhrál.



Obrázek 3.2: Hra před spuštěním

V obrázku 3.2 je hra v pozastaveném stavu a je zobrazeno počáteční rozložení hráčů. Na obrázku lze vidět tvavě šedé čtverce, které značí možnou pozici Pana X. Na začátku hry, je těchto pozic mnoho, jelikož Pan X doposud svoji pozici neodhalil.



Obrázek 3.3: Jedenácté kolo hry

Na obrázku 3.3 je zobrazeno jedenácté kolo hry. Světlé šedý a nejvýraznější čtverec značí poslední známou pozici Pana X. Tmavě šedé čtverce opět značí možné pozice Pana X. Na tomto obrázku lze pozorovat, že policisté jsou v blízkosti Pana X a snaží se ho chytit. Všichni policisté se z pozic, zobrazených na obrázku 3.2, přesunuli na možné pozice Pana X.

3.3 Prostředí a učení agentů

3.3.1 Použité technologie pro učení

K učení strategie pro hru Scotland Yard byl využit open-source framework `Ray` [23]. Konkrétně byla využita knihovna `Ray.Rllib`. Tato knihovna poskytuje nástroje pro posilované učení a samotné implementace jednotlivých algoritmů, včetně zkoumaného algoritmu *PPO*. Rllib dokáže využívat obě populární knihovny pro strojové učení `Tensorflow` a `Pytorch`. Pro tuto práci byla vybrána knihovna `Pytorch`.

Před finálním rozhodnutím pro využití frameworku `Ray` byly zkoumány i další možnosti. Knihovna Stable Baselines 3, která byla zavrhnuta, jelikož neumožňuje učení více strategií souběžně při hraním agentů proti sobě. Knihovna *CleanRL* byla také zavrhnuta, jelikož nemá dostatečně implementované trénování více strategií souběžně.

Vytvořit prostředí pro platformu `Ray.Rlib` bylo složité. Dokumentace vytváření prostředí není příliš komplexní, byl tedy problém zajistit správnou komunikaci prostředí a agentů. Prostředí a následné učení je komplikované, jelikož zde figuruje více agentů a naráz se učí dvě rozdílné strategie.

3.3.2 Trénování agentů pomocí algoritmu DQN a PPO

Agenti se již od začátku učí zásadně pouze hraním proti sobě. Důvodem pro toto rozhodnutí bylo, že se jedná o zajímavější experiment, kdy jsou agenti vpuštěni do zcela neznámého prostředí a nemají se od koho učit strategii. Jejich učení tedy není nijak podporováno a musí se naučit strategii sami. Tím je i pozorováno, jestli jsou agenti vůbec schopni rozpoznat cíl hry, co je po nich požadováno a případně jak rychle k dosáhnou dobré strategie.

Implementace algoritmu PPO i DQN je obsažena v knihovně `Ray.Rlib`. PPO využívá základní parametry trénování bez žádné modifikace parametrů jelikož se ukázalo, že agenti se chovají optimálně a hru pochopili i bez těchto zásahů do výchozích hodnot hyperparametrů. Zato DQN agenti se chovali *ne*optimálně. Agenti se naučili pouze stát na místě, přestože tato taktika byla velmi nevýhodná. Pro vyřešení tohoto problému, je nyní agentům udělována penalizace pokud se 5 kroků po sobě nepohnou. Penalizace je rovna zápornému počtu kol nečinnosti. Postupně tedy roste a pohybem je resetována. Tato expanze odměn vedla k vylepšení strategie agentů DQN a nadále nestáli pouze na jednom místě. Bohužel však přes veškeré snahy, agenti DQN nedosahují ani zdaleka tak dobrých výsledků jako agenti PPO.

S konfiguračními parametry `Ray.rlibu` a hyperparametry DQN jsem dlouho experimentoval. Měnila se velikost vnitřních vrstev, počet neuronů, rychlost učení, velikost paměti, velikost dávky učení a další.

Výsledná konfigurace je následující. Obsahuje 2 vnitřní skryté vrstvy, první vrstvá má 128 neuronů a druhá vrstva má 64 neuronů. Používá verzi *double DQN* Algoritmus k vyvážení explorace a exploatace při učení využívá metodu ϵ–greedy, kde je *epsilon* postupem trénování lineárně snižováno z hodnoty 1 na 0.05. Tato hodnota udává pravděpodobnost náhodné volby akce. Po natrénování modelu je ve hře explorace vypnuta, *epsilon* je tedy nastaveno na 0. K posílení explorace během trénování je k vahám sítě přidán „šum“. Pro zlepšení stability je povoleno přehrávání zkušeností. Avšak i přesto jsou výsledky agentů DQN neuspokojivé, viz.Experimenty.

Řešení zvolení nevalidních akcí

Hra Scotland Yard má omezené herní pole. Tím pádem je jasné, že pokud se hráči vyskytují na okraji herního pole, nemohou zvolit takovou akci, která by je posunula mimo toto herní pole. Policisté také nemohou zvolit akci, která by je posunula na stejné pole, ve kterém již je jiný policista. K vyřešení tohoto problému se nejčastěji využívá tzv. maska akcí (*Action mask*). Framework Ray tuto možnost v omezené míře podporuje. Vytvořené prostředí je, avšak potřeba obalit v obalové třídě, která tuto funkcionalitu zprostředkuje. Bohužel se mi nepodařilo tuto funkcionalitu spojit s dalšími požadavky systému, které také vyžadují zabalení do jiné třídy. Mezi těmito požadavky je fungování více aktérů s různými strategiemi v jednom prostředí a různé pozorovací prostory agentů.

strategie, tedy může zvolit nevalidní akci, ale je za jejich zvolení velmi penalizována. Agenti ji tedy volí opravdu zřídka, a to jen na základě velmi malé explorační šance a pouz na počátku trénování. Pokud se i tak stane, je ve hře implementovaná funkcionalita, která se pokusí znovu vygenerovat další akce, dokud vygenerovaná akce není validní. Aby

se zamezilo nekonečnému cyklu čekání na validní akci, je po 100 pokusech vygenerována náhodná validní akce. Takto uměle generovaná akce, nebyla za celou dobu testování agentů PPO potřeba, avšak ze statistického hlediska může tato situace nastat.

Zavedení tohoto omezení pomocí systému odměn vedlo k positivním výsledkům a agenti se naučili jaké akce jsou nevalidní a nepoužívají je.

Systém odměn

Jednotlivým agentům jsou udělovány odměny na základě jejich chování. Jelikož policisté a Pan X mají odlišné a protichůdné strategie i jejich odměny jsou odlišné. Odměny které agenti dostávají byly navrženy odhadem a laděny postupným experimentováním. Konečné odměny jsou vypočítány následovně:

$$R_{\rho_p} = \rho_{p_c} - \rho_{pref} \tag{3.1}$$

$$R_{\rho_l} = \rho_l * 0.2 \tag{3.2}$$

$$R = R_{\rho_p} + R_{\rho_l} \tag{3.3}$$

- R_{ρ_p} - odměna za vzdálenost od policistů
- R_{ρ_l} - odměna za vzdálenost Pana X od jeho poslední známé pozice
- R - celková odměna
- ρ_l - vzdálenost Pana X od jeho poslední známé pozice
- ρ_{p_c} - vzdálenost nejbližšího policisty od Pana X
- ρ_{pref} - pomezí mezi kladnou a zápornou odměnnou. Pokud je vzádelenost od policisty menší než tato hodnota, Pan X obdrží záporno odměnu a naopak kladnou.

Bylo také experimentováno s variantou kde byla odměna za vzdálenost od policistů počítána pro všechny policisty zvlášť a následně byla zprůměrována:

$$R_{\rho_p} = \sum_{i=1}^{3} [(\rho_{p_i} - \rho_{pref})] \div 3 \tag{3.4}$$

Což ale vedlo k tomu, že Panu X tolik „nevadilo“ když byl blízko jednoho policisty, ale daleko od zbylých dvou. Výpočet této odměny se tedy změnil na výše uvedený vztah 3.1. Což vedlo ke zlepšení výsledků Pana X.

Výpočet odměny policistů je složitější. Policisté znají poslední odhalenou pozici Pana X. Okolo tohoto bodu je vytvořená oblast zájmů, ve které se Pan X může nacházet. Policisté znají bod této oblasti, který je jim nejbližší, neboli vstupní bod. Čím blíže je policista k tomuto bodu, tím větší odměnu získává. Pokud je od oblasti velmi vzdálený obdrží penalizaci. Jakmile policista docílí oblastí zájmů dostává konstantní odměnu za to, že se pohybuje v lokaci ve které se může objevovat Pan X.

PPO algoritmus může začít v prostředích s diskrétními akcemi volit velmi neoptimální akce. Toto nastává pokud má příliš málo podstatných odměn[12]. To se dokonce ukázalo i během tohoto experimentu, kdy v raných fázích vývoje nebyla udělována odměna za vzdálenost od oblasti zájmu. Učení nebylo příliš úspěšné, ale přídáním této dílčí odměny

za interakci s touto oblastí se výsledná strategie značně zlepšila a agenti dosahovali lepších výsledků.

Tento styl odměňování policistů se ukázal jako velmi efektivní. Další varianta byla přidat do pozorování policistů pozice všech pozic, kde se může Pan X nacházet. Toto by ale bylo méně efektivní, jelikož by se pozorovací prostor několikanásobně zvětšil. Tím by se zvýšila náročnost výpočtů a zpomalilo by se učení.

Pozorovací a akční prostor agentů

V této práci vzájemně v prostředí vystupují dva agenti s různými strategiemi a různými pozorovacími prostory. Akční prostor je stejný jak pro policisty, tak pro Pana X.

Pozorovací prostor agentů je velmi důležitý pro úspěšné naučení dobré strategie. Struktury jednotlivých pozorovacích prostorů jsou následující:

- Shodné položky
  - Pozice X, Y sama sebe
  - Pozice X, Y a vzdálenost od poslední známé pozice pana X
  - Číslo aktuálního kola
  - Maximální počet kol
  - Číslo kola, kdy dojde k dalšímu odhalení pozice Pana X
- **Pan X**
  - Pozice X, Y policistů
  - Vzdálenost od policistů
- **Policisté**
  - Pozice X, Y zbylých policistů
  - Vzdálenost od zbylých policistů
  - Pozice X, Y nejbližšího bodu oblasti zájmu
  - Vzdálenost od nejbližšího bodu oblasti zájmu
  - Pravdivostní hodnota, zda je policista v oblasti zájmu

Velikost pozorovacího prostoru tedy není příliš velká a je snadněji zpracovatelná, než kdyby obsahoval všechny herní políčka. Celkově má tedy pozorovací prostor Pana X 17 rozměrů a pozorovací prostor policistů 18 rozměrů. Tento prostor je diskrétní s typovou hodnotou *numpy.float32*, jelikož se v něm vyskytují hodnoty vzdálenosti a ty nejsou celočíselné.

Pokud informace obsažená v těchto pozorovacích prostorech není v daný moment definována, je nahrazena hodnotou -1. Konkrétně se jedná pouze o hodnoty týkající se poslední veřejně známé pozice Pana X, jelikož v prvních několika kolech není známa.

Akční prostor agentů reprezentuje možné akce, které mohou agenti provést. Tyto akce jsou stejné jak pro policisty, tak pro Pana X. Jedná se ortogonální i diagonální pohyb. Agent také může zůstat na místě a neprovést žádnou akci. Celková velikost akčního prostoru je tedy 9. Jedná se o diskrétní prostor, jehož výstupem je hodnota 1 má–li být akce zvolena.

Kapitola 4

Experimenty

Pro ověření vlastností algoritmu PPO byly provedeny dva experimenty. Experimenty byly prováděny na systému s následujícími parametry:

- Operační systém: Windows 10 (64-bit)
- Procesor: Intel Core i5–9300H
- Paměť: 16 GB DDR4
- Grafická karta: NVIDIA GeForce GTX 1660 Ti

Agenti DQN i PPO se učili ve stejném prostředí, za stejných počátečních podmínek, měli totožný stavový i akční prostor a od prostředí získávaly stejné odměny.

Pro uskutečnění těchto experimentů byly vytvořeny skripty v jazyce Python. Jeden pro sběr dat a druhý pro jejich následné vyhodnocení a zpracování do grafů a textové podoby.

4.1 Experiment 1: Pozorování vývoje chování během tréninku

Experiment trénování spočívá v tom ukázat, jak se agenti postupem trénování zlepšují. U agentů DQN a PPO je očekáván nárůst průměrné odměny a snížení průměrné vzdálenosti policistů od Pana X. Experiment v průběhu tréninku periodicky provádí simulace her mezi agenty a zaznamenává jejich průběh.

Hlavními sledovanými parametry jsou:

- Průměrná odměna Pana X
- Průměrná odměna policistů
- Průměrná vzdálenost policistů od Pana X
- Počet vyhraných her Pana X
- Počet vyhraných her policistů

Experiment sleduje vývoj těchto hodnot po **1000 iterací**. Což znamená, že na konci experimentu, podstoupili všichni agenti přesně 1000 iterací tréninku

Simulační hry byly prováděny ve všech možných kombinací algoritmů. Zkoušeny tedy byly všechny algoritmy jak v roli Pana X, tak v roli policistů. Simulace byly spouštěny po různém počtu trénovacích iterací v závislosti na aktuálním počtu již provedených iterací.

Počet simulací byl následovný:

- 50 simulací každých 10 iterací, do 100 trénovacích iterací
- 50 simulací každých 20 iterací, od 100 do 500 trénovacích iterací
- 50 simulací každých 50 iterací, od 500 do 1000 trénovacích iterací

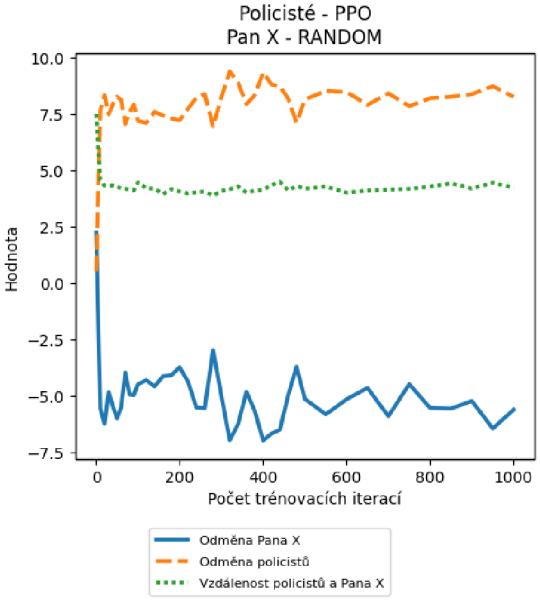
Při učení je nejzajímavější sledovat vývoj chování agentů na začátku trénování. Proto byly simulace prováděny s větší četností právě na začátku a postupem trénování byla četnost snižována pro snížení výpočetní a časové zátěže simulací. Tím bylo zajištěno, že experiment správně a efektivně vyobrazuje jak rychle a jestli, agenti byly schopni pochopit cíl hry. Sběr dat pro tento experiment trval přibližně 18 hodin. Při četnějším sběru dat by rapidně narostla časová náročnost experimentu a dle testovacích spuštění a výpočtů by trval i několik dní.

4.1.1 Výsledky experimentu

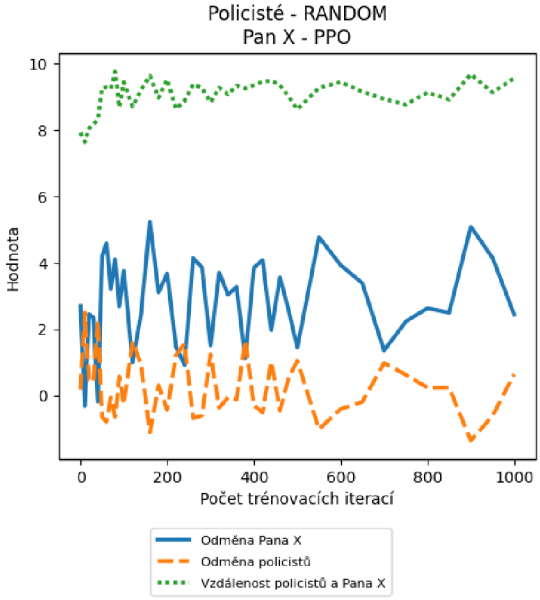
Při sledování výsledků tohoto experimentu je důležité si uvědomit že díky základní premise hry Scotland Yard je pro policisty vyhrát mnohem složitější než pro Pana X. A to z toho důvodu, že policisté dokáží vyhrát pouze pokud chytí Pana X, zato Pan X může vyhrát jenom na základě uplynutí maximálního počtu kol.

Všechny následující grafy vyobrazují průměrnou odměnu Pana X (modá, nepřerušovaná linie) a policistů (oranžová, přerušovaná linie), průměrnou vzdálenost policistů od Pana X (zelená, tečkovaná linie) a to vše v závislosti na počtu trénovacích iterací.

Srovnání s náhodným agentem



Obrázek 4.1: Graf simulace her mezi policisty trénovanými pomocí PPO a Panem X volícím náhodné akce



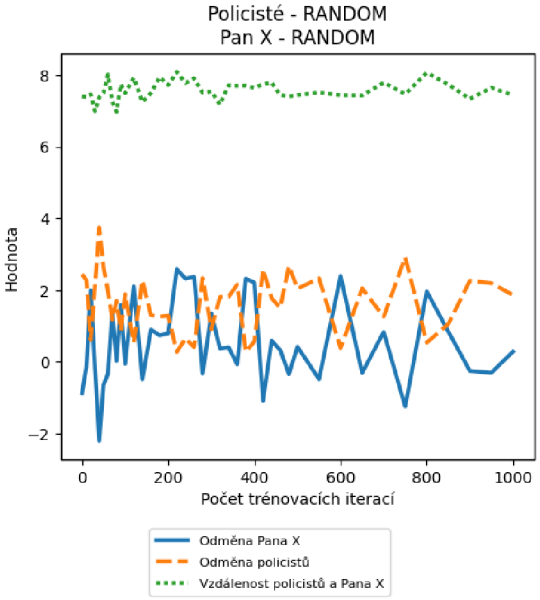
Obrázek 4.2: Graf simulace her mezi Panem X trénovaným pomocí PPO a policisty volícími náhodné akce

Jak jde vidět na grafu 4.1, policisté jsou velmi úspěšní ve své strategii. Vzdálenost mezi policisty, trénovanými pomocí PPO a Panem X, který volí náhodné akce, je po celou dobu

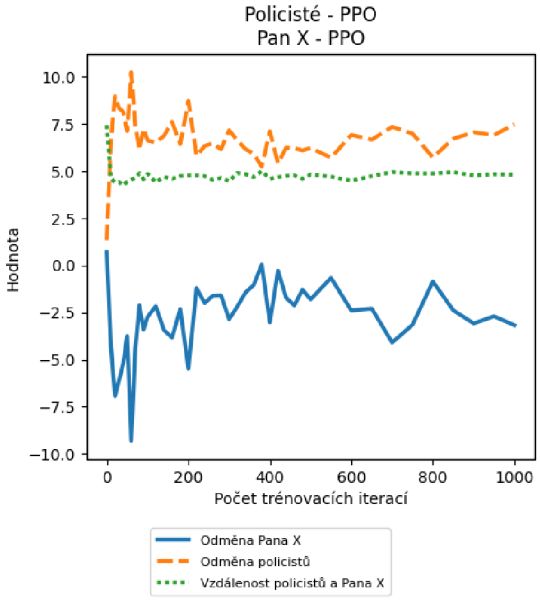
velmi malá. Odměna kterou policisté získávají je velmi vysoká. Odměna Pana X je naopak velmi nízká.

Na grafu lze také sledovat že průměrná vzdálenost policistů od Pana X na začátku trénování rapidně klesá, stejně tak jako klesá odměna Pana X tím, že se zlepšuje strategie policistů. Po tomto rapidním poklesu se vzdálenost policistů od Pana X stabilizuje a odměna Pana X se drží na nízké hodnotě.

Naopak z grafu 4.2 lze pozorovat, že je odměna Pana X kladná, a až na odchylky vyšší než odměna policistů, kteří se chovají náhodně. Odměna policistů se pohybuje kolem nuly a odměna Pana X je stále kladná. Také vzdálenost policistů od Pana x je skoro dvojnásobná než v předchozím případě.



Obrázek 4.3: Graf simulace her mezi policisty a Panem X, kde oba volí náhodné akce



Obrázek 4.4: Graf simulace her, kde oba agenti volí akce dle modelu PPO

Obrázek 4.5: Srovnání PPO agenty s agenty volící náhodné akce

Na srovnání 4.5 lze vidět, že na jak PPO, tak náhodný algoritmus udržují velice stabilní vzdálenost. U náhodného chování je to kvůli základní pravděpodobnosti, která musí být průměrně stejná. U grafu agentů PPO, to avšak vypovídá o tom, že policisté dokáží stabilně odhadovat pozici Pana X a přinejmenším se pohybovat v jeho blízkém okolí. Z grafu4.4 lze pozorovat, že odměna policistů a Pana X kolísá. Frekvence tohoto kolísání se postupně jemně zmenšuje. Z toho odvozuji, že toto kolísání vypovídá o tom, že se agenti postupně učí strategii protivníka a snaží se podle ní upravit svoji strategii, aby byli opět lepší. Což značí, že se obě strategie pomalu přibližují k optimu. Předpokládám, že dalším trénováním, by se tento trend potvrdil. Avšak k tomu by bylo potřeba zapotřebí větší výpočetní výkon a spoustu času. Ale jak lze vidět z výsledků experimentu, i přes relativně malý počet trénovacích iterací se objevuje očekávané chování a rozdíl s náhodným chováním je markantní.

Závěr experimentu pro algoritmus PPO

Každá možná kombinace využití algoritmů pro policisty a Pana X byla během experimentu testována celkově 2050×. Z tohoto testování vzešly kromě grafů vývoje chování, také údaje zaznamenávající výsledky jednotlivých her. Tyto výsledky byly zpracovány do tabulek, které zobrazují procentuální počet výher jednotlivých stran v závislosti na použitém algoritmu.

Policisté	Pan X		
	PPO	Random	DQN
PPO	38 %	67 %	65 %
Random	11 %	20 %	18 %
DQN	10 %	17 %	19 %

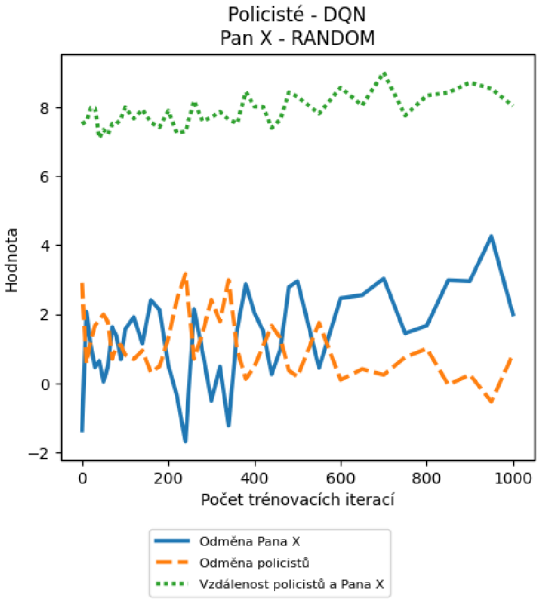
Tabulka 4.1: Zobrazuje procentuální počet výher policistů proti Pánovi X s vybraným algoritmem během experimentu

Pan X	Policisté		
	PPO	Random	DQN
PPO	62 %	89 %	90 %
Random	33 %	80 %	83 %
DQN	35 %	82 %	81 %

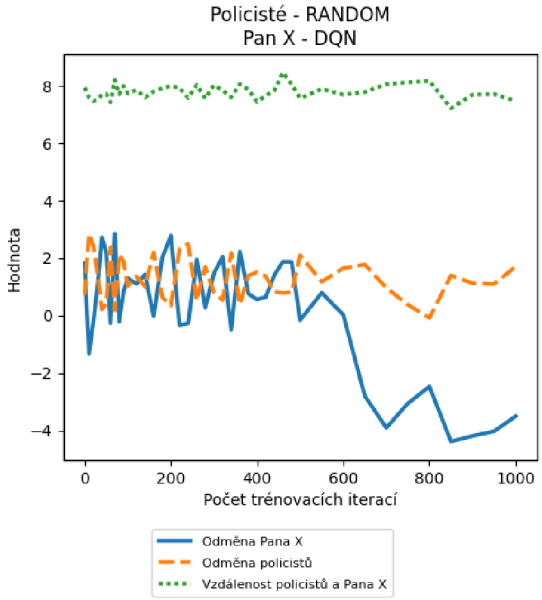
Tabulka 4.2: Zobrazuje procentuální počet výher Pana X proti policistům s vybraným algoritmem během experimentu

Pokud zde opět porovnáme výsledky, kdy proti sobě hráli dva náhodní agenti a agenti trénovaní pomocí PPO, lze vidět, že agenti trénovaní pomocí PPO mají výrazně lepší výsledky.

Ostatní výsledky



Obrázek 4.6: Graf simulace hry mezi policisty trénovanými pomocí DQN a Panem X volícím náhodné akce



Obrázek 4.7: Graf simulace hry mezi Panem X trénovaným pomocí DQN a policisty volícími náhodné akce

Jak již jsem zmiňoval, výsledky trénování agentů pomocí algoritmu DQN bohužel nemají dobré výsledky. Výsledné chování se velmi podobá náhodnému agentovi. Z předchozích prací je známo, že by algoritmus DQN měl mít znatelně lepší výsledky než náhodná metoda [11].

Algoritmus DQN je velice citlivý na hyperparametry a je možné, že bylo zvoleno nevhodné nastavení. Potřebuje také mnohem více trénovacích iterací než PPO. Je tedy možné, že 1000 iterací není dostatečné pro naučení dobré strategie pro algoritmus DQN. Jelikož jsou ale trénovací iterace PPO časově náročnější než iterace DQN, nebylo možné sladit jejich počet.

Trénování z tohoto experimentu poslouží jako základ pro další experiment, kde budou simulovány hry s již natrénovanými agenty.

4.2 Experiment 2: Simulace hry s již natrénovanými agenty

Tento experiment byl proveden za účelem ověření, zda jsou agenti trénováni pomocí algoritmu PPO schopni pochopit cíl hry a naučit se optimální strategii. Porovnává, jestli již natránovaní agenti z experimentu 1, trénování tímto způsobem, výrazně liší od agenta který volí pouze náhodné akce a tím prokázat že se chová způsobem, který vede k vítězství.

Pro ověření zkůtečného výkonu agentů byly provedeny simulace hry mezi již natrénovanými agenty. Ačkoli již předchozí experiment dokazuje, že agenti trénovaní pomocí algoritmu PPO jsou schopni pochopit cíl hry, je důležité získat skutečná data o výkonu jednotlivých metod, bez zkreslených výsledků kvůli trénování.

Následující tabulky byly získány ze 10000 simulací pro každou kombinaci již natrénovaných algoritmů.

Policisté	Pan X		
	PPO	Random	DQN
PPO	26 %	72 %	64 %
Random	10 %	19 %	18 %
DQN	8 %	17 %	17 %

Tabulka 4.3: Zobrazuje procentuální počet výher policistů proti Pánovi X s vybraným algoritmem

Pan X	Policisté		
	PPO	Random	DQN
PPO	74 %	90 %	92 %
Random	28 %	81 %	82 %
DQN	36 %	82 %	83 %

Tabulka 4.4: Zobrazuje procentuální počet výher Pana X proti policistům s vybraným algoritmem

Z tabulky lze vyčíst, že policisté trénováni pomocí PPO vyhráli proti Pánovi x s náhodným chováním 72 % her. V porovnání s policisty volící náhodné akce je to výrazně lepší výsledek. Ti proti Pánovi x s náhodným chováním vyhráli pouze 19 % her.

Z her vzešly také tyto informace:

- Průměrná vzdálenost mezi Panem X a policisty (oba PPO): 4.779692656488397
- Průměrná vzdálenost mezi Panem X a policisty (oba náhodné chování): 7.631191103120987
- Průměrná vzdálenost mezi Panem X a policisty (oba DQN): 7.869671633109524

Z tohoto experimentu lze tedy vyvodit, že agenti trénovaní pomocí algoritmu PPO prokázali, že se byli schopni naučit optimální strategii i ve hře s neúplnou informací.

Kapitola 5

Závěr

Provedené experimenty ukázaly, že algoritmus PPO je skutečně vhodný na hry s neúplnou informací. Jelikož policisté trénování pomocí PPO dokázali odhadovat přibližnou pozici Pana X a měli mnohem větší úspěšnost než náhodní agenti. Agenti se učili formou hraní proti sobě a dokázali pochopit koncept a cíl hry Scotland Yard. Naučili se v ní chovat optimálně, přestože začínali s nulovým věděním o prostředí. Bylo příjemným překvapením, že již po pouhých 10 iterací trénování byli agenti schopni projevovat náznaky pochopení cíle hry.

Možná vylepšení

Bohužel se během experimentu nepovedlo separovat výslednou strategii od frameworku `Ray.Rlib` a extrahovat ji do podoby kde by již `Rlib` nebyl potřeba pro její využití. Při spouštění hry se tedy načítá i celý framework `Ray.Rlib`, toto velmi zpomaluje načítání hry kdy prvotní načtení trvá několik sekund. Toto a spoustu dalších potíží jsou důvody proč zvolení knihovny Ray.Rllib zpětně lituji. Měl jsem spíše algoritmus PPO implementovat svépomocí s využitím `Pytorch` či `Tensorflow`. Nebo využít knihovnu `CleanRl` a učení více strategií naráz vyřešit jiným způsobem.

Původním plánem bylo také realizovat možnost hraní hry mezi lidmy a agenty. Tuto implementaci jsem, avšak přes problémy s experimenty a algoritmem DQN nestihl dokončit. Mám za to, že by tato možnost nadále prohloubila výsledky této studie o zajímavé experimenty, kde by bylo možné sledovat jak si agent PPO vede proti lidskému protivníkovi.

Agenti trénování pomocí algoritmu DQN měli velmi špatné výsledky. Je vyloučeno, že by byla chyba v prostředí, udělování odměn či v pozorování jelikož se agenti trénovaní pomocí PPO učili za stejných podmínek a jejich výsledky byly dobré. Je možné, že i přes hledání optimálních parametrů s pomocí `Ray.Tune` byla zvolena nevhodná konfigurace. Byť jsem s těmito parametry dlouhosáhle experimentoval, nebylo dosaženo uspokojivých výsledků. Navrhuji změnu stavového a akčního prostoru, aby byl kompatibilní s verzemi studentů z minulých let, kteří realizovali bakalářskou práci na podobné téma. A následně využít jejich implementaci pro další experimenty [11].

Dále je zde možnost rozšířit hratelnost hry o jednotlivé druhy dopravy a k nim příslušící jízdenky. Tím by výrazně vzrostla složitost hry a stavový prostor. Jak ale ukázala práce OpenAI [17], algoritmus PPO nemá problém s obrovským stavovým i akčním prostorem a naučit se i komplikované hry.

Literatura

[1] Baes, J. *Application of Reinforcement Learning Algorithms to the Card Game Manille*. 2022. Diplomová práce. UGent. Faculteit Ingenieurswetenschappen en Architectuur.

[2] Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G. et al. *Emergent Tool Use From Multi-Agent Autocurricula*. 2020. Dostupné z: https://arxiv.org/abs/1909.07528.

[3] BELLMAN, R. a Dreyfus, S. *Dynamic Programming*. 1. vyd. Princeton University Press, 2010. ISBN 9780691146683. Dostupné z: http://www.jstor.org/stable/j.ctv1nxcw0f.

[4] Berger Tal, O.; Nathan, J.; Meron, E. a Saltz, D. The Exploration-Exploitation Dilemma: A Multidisciplinary Framework. *PLOS ONE*. 1. vyd. Public Library of Science, Duben 2014, sv. 9, č. 4, s. 1–8. Dostupné z: https://doi.org/10.1371/journal.pone.0095693.

[5] Borowiec, S. *AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol* online. 2019. Dostupné z: https://www.theguardian.com/technology/2016/mar/15/googles-alphago-seals-4-1-victory-over-grandmaster-lee-sedol. [cit. 2024-03-18].

[6] Carr, T. *Policies in Reinforcement Learning* online. March 2024. Dostupné z: https://www.baeldung.com/cs/rl-deterministic-vs-stochastic-policies. [cit. 2024-03-20].

[7] CDcodes), C. Y. channel:. *Game states* online. June 2021. Dostupné z: https://github.com/ChristianD37/YoutubeTutorials/tree/master. [cit. 2024-03-20].

[8] Cognilytica. *Action - Cognilytica* online. Unknown. Dostupné z: https://www.cognilytica.com/glossary/action/. [cit. 2024-05-02].

[9] Foundation, F. *Gymnasium*. 2024. Dostupné z: https://gymnasium.farama.org/.

[10] Guinness, H. *Here's how a new AI mastered the tricky game of Stratego* online. 2022. Dostupné z: https://www.popsci.com/technology/ai-stratego/. [cit. 2024-03-20].

[11] Hrkľová, Z. *Metody hlubokého učení pro strojové hraní hry Scotland Yard [online]*. Brno, CZ, 2023 [cit. 2024-05-04]. Bakalárska práca. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://theses.cz/id/811r0e/.

[12] Hsu, C. C.-Y.; Mendler Dünner, C. a Hardt, M. *Revisiting Design Choices in Proximal Policy Optimization*. 2020. Dostupné z: https://arxiv.org/abs/2009.10897.

[13] Lindner, J. *Statistics About The Average Reaction Time* online. February 2024. Dostupné z: https://gitnux.org/average-reaction-time/. [cit. 2024-03-25].

[14] Mliu92. *Scotland Yard schematic* online. 2023. Dostupné z: https://commons.wikimedia.org/wiki/File:Scotland_Yard_schematic.svg. [cit. 2024-03-20].

[15] Narahari, Y. *Game Theory* online. 2012. Dostupné z: https://gtl.csa.iisc.ac.in/gametheory/ln/web-ncp13-bayesian.pdf. [cit. 2024-03-19].

[16] Newborn, M. *Kasparov versus deep blue: computer chess comes of age*. 1. vyd. Springer-VerlagBerlin, Heidelberg, 1996. ISBN 978-0-387-94820-1.

[17] OpenAI; :; Berner, C.; Brockman, G.; Chan, B. et al. *Dota 2 with Large Scale Deep Reinforcement Learning*. 2019. Dostupné z: https://arxiv.org/abs/1912.06680.

[18] OpenAI. Emergent tool use from multi-agent interaction. online, September 2019. Dostupné z: https://openai.com/research/emergent-tool-use. [cit. 2024-03-27].

[19] Perolat, J.; De Vylder, B.; Hennes, D.; Tarassov, E.; Strub, F. et al. Mastering the game of Stratego with model-free multiagent reinforcement learning. *Science*. 1. vyd. American Association for the Advancement of Science (AAAS), prosinec 2022, sv. 378, č. 6623, s. 990–996. ISSN 1095-9203. Dostupné z: http://dx.doi.org/10.1126/science.add4679.

[20] Roy, R. *ML | Monte Carlo Tree Search (MCTS)* online. May 2023. Dostupné z: https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts. [cit. 2024-03-25].

[21] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A. a Klimov, O. *Proximal Policy Optimization Algorithms*. 2017. Dostupné z: https://arxiv.org/abs/1707.06347.

[22] Stankiewicz, J. a Schadd, M. *Belgian/Netherlands Artificial Intelligence Conference*. Leden 2009. Dostupné z: https://www.researchgate.net/publication/237395013_Opponent_Modelling_in_Stratego.

[23] Team, T. R. *Ray Documentation*. 2024. Dostupné z: https://docs.ray.io.

[24] Weng, L. *From GAN to WGAN*. 2017. Dostupné z: https://lilianweng.github.io/posts/2017-08-20-gan/#kullbackleibler-and-jensenshannon-divergence.

[25] Weng, L. *A (Long) Peek into Reinforcement Learning* online. February 2018. Dostupné z: https://lilianweng.github.io/posts/2018-02-19-rl-overview/#key-concepts. [cit. 2024-03-20].

[26] Weng, L. *Policy Gradient Algorithms*. 2018. Dostupné z: https://lilianweng.github.io/posts/2018-04-08-policy-gradient/.

Příloha A

Obsah přiloženého paměťového média

- `src/` – Složka se zdrojovými kódy aplikace.
- `text/` – Složka s zdrojovými kódy textu práce v jazyce LaTeX.
- `trained_policies_dqn/` – Složka s natrénovaným modelem DQN.
- `trained_policies_ppo/` – Složka s natrénovaným modelem PPO
- `thesis.pdf` – Soubor s textem práce.
- `thesis_print.pdf` – Soubor s textem práce pro tisk.
- `README.md` – README soubor pro tuto práci.
- `simulations/` – Složka s výsledky simulací.
- `arial.ttf` – Písmo pro grafické rozhraní
- `main.py` – Soubor, jejož spuštěním se spustí grafické rozhraní hry Scotland Yard.
- `TrainerDQN.py` – Soubor obstarávající trénování modelu DQN.
- `TrainerPPO.py` – Soubor obstarávající trénování modelu PPO.
- `tune_dqn.py` – Tune byl použit pro hledání nejlepších hyperparametrů pro model DQN.
- `tune_ppo.py` – Tune byl použit pro hledání nejlepších hyperparametrů pro model PPO.
- `requirements.txt` – Soubor s python balíčky potřebnými pro spuštění aplikace.