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 je PPO – Proximal Policy Optimization (optimalizace proximální politiky).

Práce se zabývá jeho teoretickými základy a následně jeho aplikací na hru Scotland Yard. Výsledky jsou porovnány s jinými algoritmy a je provedena analýza výhod a nevýhod zhotovené implementace.

[Výsledky jsou . . .]

Abstract

asd Přeložit asds

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

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.

Rozšířený abstrakt

Úvod

Hra Scotland Yard

Experimenty

- *Porovnání s náhodnou politikou*
- *Porovnání s jinými algoritmy - implementace DQN - v knihovně rlib*
- *Porovnání s monte carlo algoritmem - zdroj na internetu*

Heuristiky pro hraní hry Scotland Yard

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Cejpek
27. března 2024

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

Seznam obrázků

Kapitola 1

Úvod

Umělá inteligence je obor, který nás postupem let všechny obkopuje čím dál tím více. Dokonce je navždy spjata i s naším českým národem, když Karel Čapek dal zrodu slova robot.

Pokrok umělé inteligence je často měřen aplikací v oblasti her. Hry jsou vhodným ukazatelem pokroku v oblasti umělé inteligence, protože mají jasná pravidla, výkon je snadno měřitelný a pokrok dokáže vidět i lajk. Umělá inteligence již dokázala porazit nejlepší hráče v šachu [10], Dota 2 [11] a Go [4].

Hra studovaná v této práci je hra Scotland Yard. Je to hra 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. Musí tedy odhadovat jeho pozici a spolupracovat mezi sebou, aby ho mohli polapit. Pozice Pana X je odhalena pouze v určitých kolech. Hra končí, když je Pan X chycen (vyhrávají policisté), nebo když je dosažen maximální počet kol (vyhrává Pan X). Scotland Yard je ideální hrou ke studování umělé inteligence, protože se jedná o hru s nedokonalou informací a k vítězství policistů je zapotřebí spolupráce a strategie.

Zaměření této práce jsem si vybral jelikož mi vždy byla umělá inteligence blízká a vždy jsem chtěl začít tomuto odvětví více věnovat i po praktické stránce.

Tato práce blíže zkoumá algoritmy posilovaného učení, konkrétně algoritmu PPO (Proximal Policy Optimization) a jejich použití na hry s neurčitostí. Algoritmus PPO je často používán k řešení problémů se spojitými veličinamy a ve 3D prostoru. Často se využívá ve hrách. Dle provedených studií je vhodný k řešení problémů s nedokonalou informací [1] a je vhodný pro hry na schování a hledání [2]. Proto je pro mě zajímavý a zkušenost s tímto algoritmém by se dala využít v mém pracovním životě.

Pro zpracování práce byly využity tyto hlavní knihovny:

- *Ray.Rlib* - knihovna s implementací algoritmu PPO, použita k učení agenta
- *PyTorch* - podpůrná knihovna Ray.Rlib
- *TensorFlow* - podpůrná knihovna Ray.Rlib
- *Gym* - knihovna sloužící k vytvoření prostředí
- *Pygame* - knihovna pro vytváření uživatelského rozhraní

Kapitola 2

Shrnutí dosavadního stavu

Tato kapitola není encyklopedickým výčtem celého tématu bakalářské práce. Jedná se avšak o shrnutí nejdůležitějších relevantních informací a pojmů, duležitých pro tuto práci.

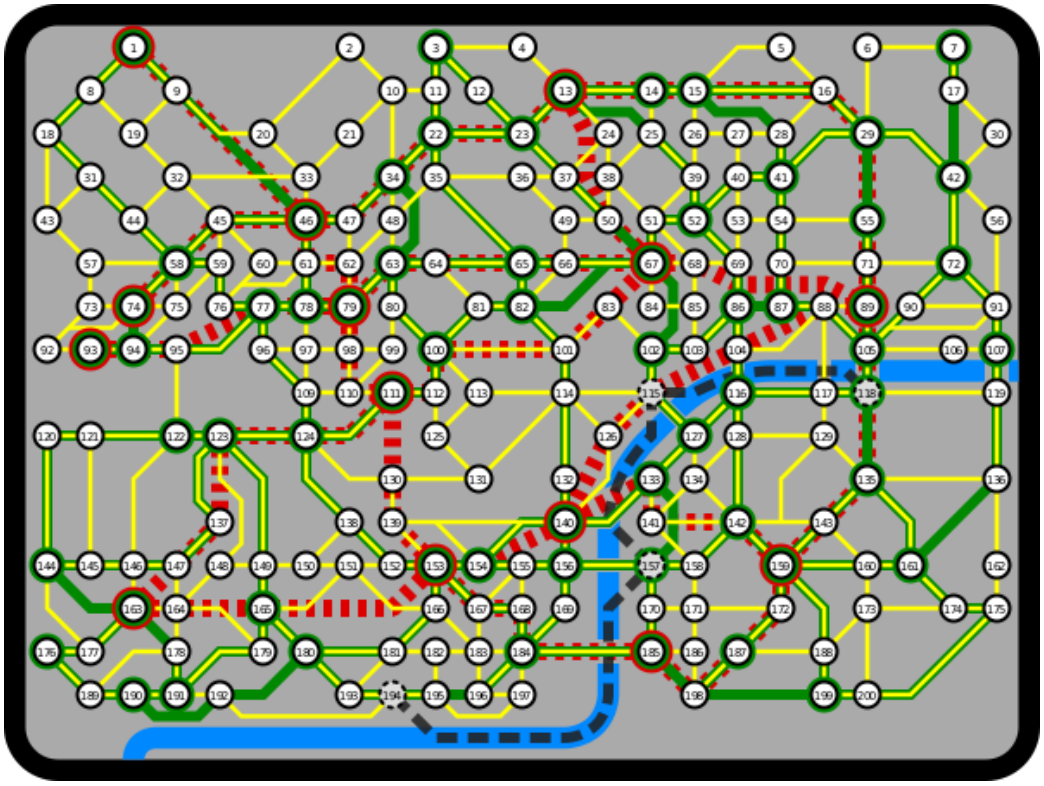
- *40-50 % rozsahu práce*
- *Hodně citovat literaturu*
- *Vysvětlit všechno, už ne pro plebíky*
- *Je vhodné na začátku této části uvést, co obsahuje a proč a taky že „není encyklopedickým přehledem“*
- *Asi tak ze 2 kapitol?*
- *Existující řešení (implementace scotlandu, říct že se implementuje pomocí tamtoho algoritmu a proč jsem vzal PPO)*

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, ovládanými ostatními hráči. Hra končí, když je Pan X chycen (vyhrávají policisté), nebo když je dosažen maximální počet kol (vyhrává Pan X). Originální hra se odehrává v Londýně. Na herní mapě se nachází 200 polí, které jsou vzájemně propojené náhodnými cestami. Každá cesta povoluje určitý způsob pohybu (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 dopravní prostředky. K využití dopravy je potřebná právě tato jízdenka. Pokud ji hráč nemá, nemůže již tento způsob přepravy použít. Hra se dělí na kola, ve kterých se hráči střídají.



Obrázek 2.1: Ukázka herní mapy hry Scotland Yard. Zdroj:[8]

Hlavní myšlenkou hry je, že po většinu kol je pozice Pana X je policistům utajena. Odhaluje se jim pouze určená kola. To znamená, že policisté musí odhadovat další kroky Pana X aby ho mohli polapit. 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 a rozvíjení oboru umělé inteligence.

2.2 Další hry s nedokonalou informací

V oblasti umělé inteligence hraje důležitou roli modelování a řešení her. Hry představují abstraktní formalizaci konfliktních interakcí mezi aktéry, tzv. hráči. 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. V praxi se však častěji setkáváme se situacemi kde jednotlivým stranám chybí určité informace.

2.2.1 Bayesovské hry

[PŘEPSAT]

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

- *Ekonomii* - kde se jedná o nedokonalou informaci o trhu, cenách, situačních výkyvech, atd.

- *Armáda* - kde se jedná o neúplnou informaci o pozici nepřítele, jeho vybavení, strategii, cíli, atd.
- *Sportovní hry* - kde se jedná o nedokonalou informaci o taktice soupeře, jeho schopnostech, atd.

Definice 1 (Bayesovská hra) *[9] je definována pěticí* $(N, A_i, \theta_i, p(\theta_i), u_i)$*, 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 her s nedokonalou informací.

2.2.2 Stratego

Stratego je desková strategická hra pro dva hráče, která se odehrává na hracím plánu rozděleném do políček a využívá se k ní sada figurek reprezentujících armádu. Vychází z dřívějších her, jako je Šachy a Go, a kombinuje strategické plánování, taktické manévry.

Cílem hry je porazit soupeře nalezením a obsazením jeho vlajky. Hráči to dělají tak, že se navzájem utkávají se svými figurkami na herním plánu.

Každý hráč má 40 figur, rozdělených do 11 hodností (generál, plukovník, skaut, atd.). Hráči mají k dispozici také bomby, které mohou zničit jakoukoliv figuru, avšak nemohou se hýbat. Minu může zničitt pouze horník. Figury lze rozeznat jen z jedné strany, proto oponent neví o jakou figuru se jedná.

Hra začíná tím, že každý hráč rozmístí své figury na herní pole. Hráči se střídají v tazích, kdy se snaží najít oponentovu vlajku. Pokud hráč táhne na pole, kde se nachází oponentova figura, nastává souboj. Souboj spočívá v odkrytí 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.

Z pohledu umělé inteligence je Stratego zapeklitý problém. Nejenže je hra s nedokonalou informací a je tedy zapotřebí odhadovat oponentovy tahy a blafovat. Hra ale také má obrovský stavový prostor 10^{535}[12]. Až do roku 2022 nebyla umělá inteligence v tétp hře moc úspěšná nedokázala se rovnat expertnímu hráče. To se změnilo příchodem *DeepNash*[12], kdy se tato metoda umístila mezi třemi nejlepšími hráči světa.

2.2.3 Poker

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.

Obrázek 2.2: Ukázka rozestavěných figur ve hře Stratego. Zdroj:[7]

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 volí takové akce aby maximalizoval kumulativní odměnu.

2.3.2 Prostředí

Je vše s čím agent interaguje. Prostředí je buď fyzické (entity z reálného světa, ovládání chytré domácnosti, ovládání reaktoru, atd.) nebo virtuální (simulace, například hra). Prostředí reaguje na akce agenta poskytuje mu zpětnou vazbu ve formě odměny či trestu (záporná odměna). Pokud v prostředí existuje více agentů, může mít každý agent jiné pozorování. Diky tomuto můžeme například schovat agentu A určité informace, které agent B vidí.

2.3.3 Model

Je matematická funkce, která popisuje chování prostředí v závislosti na agentových akcích. Model může být známý nebo neznámý, to následně rozděluje metody učení posilovaného na 2 základní kategorie: metody *s modelem* a metody *bez modelu*.

2.3.4 Politika

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

Deterministická politika

Deterministická politika přesně definuje cílový stav přechodu pro každý stav. Agent tedy pro jeden stav vždy volí stejnou akci. Tato politika 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.[5]

Rovnice deterministické politiky je:

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

Stochastická politika

Zato stochastická politika 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 politika je vhodná v situacích, kdy je potřeba zkoumat různé strategie a kdy agent nemá úplnou informaci o prostředí. Například tam kde by deterministická politika zvolila jasnou akci A, stochastická politika by mohla s malou pravděpodobností zvolit akci B. Čímž ale může odhalit, že stav B je s ohledem na komulativní odměnu lepší než stav A.[5]

Rovnice stochastické politiky je:

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

2.3.5 Akce

Akce je přechod z aktuálního stavu, do následujícího stavu z množiny možných stavů. Zjednodušeně, je to rozhodnutí, které agent vykonává v prostředí a toto rozhodnutí ovlivňuje prostředí. Akce zvolena dle politiky a je závislá na pozorování agenta.

2.3.6 Odměna

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

2.3.7 Hodnotová funkce

Hodnotová funkce vyhodnocuje, jak dobrý je stav tím, že predikuje budoucí odměnu. Čím vzdálenější odměna je, tím více je snížena. Jelikož, čím je odměna vzdálenější tím více je nejistá.

Existují dva typy hodnotových funkcí:

Hodnotová funkce stavu $V(s)$

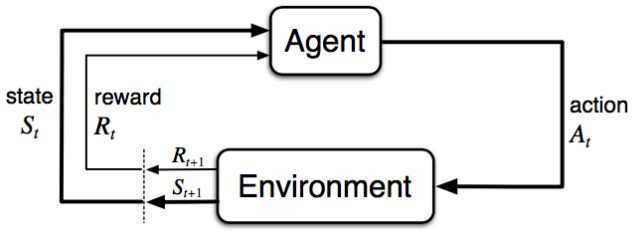
Hodnotová funkce stavu $V(s)$ vyhodnocuje očekávanou komulativní odměnu, pokud se agent nachází v tomto stavu. Tato funkce je závislá na politice, kterou se agent řídí. Vyhodnocuje tedy jak příznivý je daný stav pro agenta.

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 politice, kterou se agent řídí. Vyhodnocuije tedy jak příznié je zvolení dané akce v aktuálním stavu.

2.3.8 Markovský rozhodovací proces

Teměř 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:[14]

Definice 2 *Markovský rozhodovací proces je definován pěticí* (S, A, P, R, γ)*[14], kde:*

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

2.3.9 Bellmanova rovnice

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

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

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

2.3.10 Rovnováha mezi explorací a exploatací (exploration-exploitation dillemma)

[3, Kompromis mezi potřebou získávat nové znalosti a potřebou použít již nabyté znalosti k vylepšení výkonnosti je jedním z nejzákladnějších kompromisů v přírodě]. 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* se snaží prozkoumat možnosti, které nejsou známé a mohly by vést k lepší budoucí odměně.

2.3.11 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 hračích vše. Například ve hře šachny. Hrač 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 jejímy pravidly. A rozumí jaké jsou podmínky výhry a je obeznámn 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.

Příklady

- Nedokonalá ale kompletní informace - poker Karty ostatních hráču jsou skryté [DODĚLAT PŘÍKLADY]

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 heruistický algoritmus prohledávání. Kombinuje stromové vyhledávání s principy posilovaného učení. Je často využíván, je–li stavový prostor řešeného problém příliš velký a složitý na to, aby byl prohledán kompletně jinými

metodamy, 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 (exploration-exploitation dillemma)). Explorací se strom rozrůstá do šířky, zatímco exploatací se strom prohlubuje.

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 $(\epsilon - -greedystrategy)$ vybíra 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***

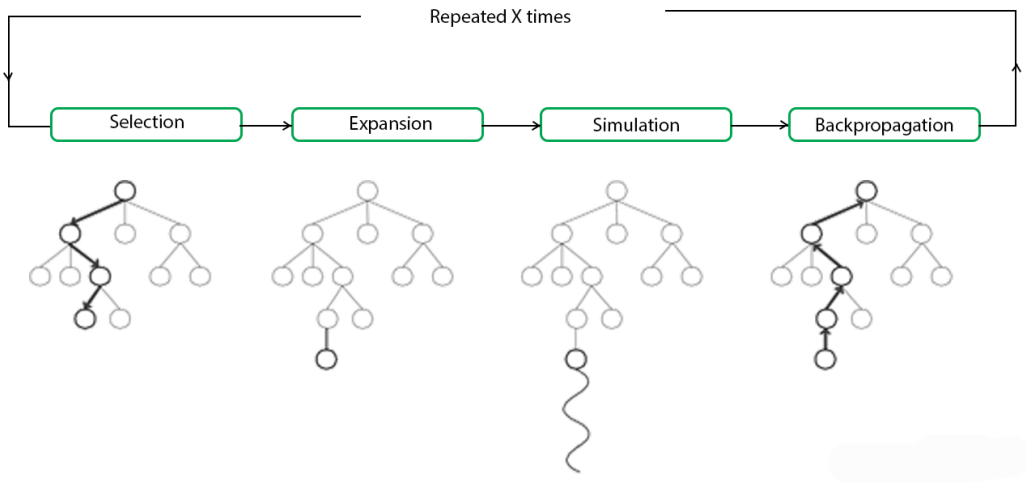
  V tomto kroku se vyhledávací strom rozšíří o nový uzel, který je výstupem z předchozího kroku.

- ***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:[**?**]

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 politiku.

Jak už název vypovídá, Q-learning se zaměřuje na určení hodnoty hodnotové funkci akce $Q(s, a)$, viz. Hodnotová funkce akce $Q(s, a)$. Pokud dostane kladnou odměnu 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. Hlavním prvkem tohoto algoritmu je Q–tabulka, která na každou kompbinaci stavu a akce uchovává hodnotu $Q(s, a)$. Řádky jsou tedy stavy a sloupce akce. Na začátku 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. Monte Carlo tree search.

Protože tato metoda mapuje Q hodnoty pro každou kombinaci stavu a akce nastává zde problém. Je–li stavový, či akční prostor příliš velký, nebo dokonce nekonečný je tento algoritmus nevhodný, skoro až nepoužitelný. Jako řešení byl navrhnut algoritmus Deep Q-learning.

2.4.3 Deep Q-learninggg

Tato metoda je rozšířením algoritmu Q-learning. Místo mapovací Q–tabulky využívá hluboké neuronové sítě k aproximaci hodnotové funkce akce $Q(s, a)$. Díky této aproximaci je možné použít tento algoritmus na problémy s velkým, či nekonečným stavovým prostorem.

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

- ***Přehrání zkušenosti (experience replay)***
- *Periodická aktualizace*

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

2.4.4 Gradient politiky

Oproti předchozím zmiňovaným algoritmům, které se snaží naučit hodnotovou funkci či prohledávají stavový prostor, algoritmy gradientu politiky se snaží naučit politiku přímo.

V diskrétním prostoru je odměnová funkce definována jako:

$$\mathcal{J}(\theta) = V_{\pi_\theta}(S_1) = \mathbb{E}_{\pi_\theta}[V_1] \tag{2.5}$$

V spojitém prostoru je odměnová funkce definována jako:

$$\mathcal{J}(\theta) = \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) V_{\pi_\theta}(s) = \sum_{s \in \mathcal{S}} \left(d_{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi(a|s, \theta) Q_\pi(s, a) \right) \tag{2.6}$$

Věta o gradientu politiky

Výpočet gradientu matematicky je zaznamenán touto rovnicí:

$$\frac{\partial \mathcal{J}(\theta)}{\partial \theta_k} \approx \frac{\mathcal{J}(\theta + \epsilon u_k) - \mathcal{J}(\theta)}{\epsilon} \tag{2.7}$$

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

$$\nabla \mathcal{J}(\theta) = \mathbb{E}_{\pi_\theta}[\nabla \ln \pi(a|s, \theta) Q_\pi(s, a)] \tag{2.8}$$

2.5 Zkoumaný algoritmus proximální optimalizace politiky

Kapitola 3

Zhodnocení současného stavu a plán práce (návrh)

- *Kritické zhodnocení dosavadního stavu*
- *Návrh, co by bylo vhodné vyřešit na základě znalostí dosavadního stavu*
- *Co jste konkrétně udělal s teorií popsanou výše*
- *Volba OS, jazyk, knihovny*
- *Detailní rozbor zadání práce, detailní specifikace a formulace cíle a jeho částí*
- *Popis použití řešení, situace/problémy, které projekt řeší*
- *Postup práce/kroky vedoucí k cíli, rozdělení celku na podčásti*
- *Návrh celého řešení i jeho částí, s odkazy na teoretickou část*

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 [15x15] polí. Hrací se po těchto polích pohybují ortogonálně i diagonálně, vždy o maximálně 1 pole. Hráč se může rozhodnot nezměnit pozici a zůstat na svém aktuálním poli. K pohybu nejsou potřebné žádné jízdenky. Toto zjednodušení herní plochy nijak nemění základní podstatu hry, zachovává neurčitost, ale značně zjednodušuje implementaci.

Předtím nežli začne hra, že se vyberou náhodné možné pozice Pana X a policistů. Z těchto možných pozíc se následně náhodná pozice přidělí jednotlivým hráčům. Poté začíná hra. Hra se dělí na jednotlivá kola, ve kterých se hráči střídají. Pro hru byli zvoleni 3 policisté z toho důvodu, že je herní pole velmi velké a 2 policisté by nemuseli mít možnost ho celé pokrýt. V kole hraje jako první Pan X a poté policisté, již podle jejich očíslení, které jim bylo přiděleno při vytváření. Hra končí v okamžiku, kdy policisté chytí Pana X nebo když Pan X zůstane nepolapen až do konce.

Implementace umělé inteligence do hry Scotland Yard pomocí algoritmu PPO

3.0.1 Použité technologie

K učení politiky pro hru Scotland Yard byla využit open-source framework *Ray* [13]. Konkrétně byla využita knihovna *Ray.Rllib*. Tato knihovna poskytuje nástroje pro posilované učení a samotné implementace jednotlivých algoritmů, včetně zkoupameného 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. [Dopsat] Knihovna Stable Baselines 3. Knihovna byla zavrhnuta, jelikož neumožňuje učení více politik se hraním agentů proti sobě.

3.0.2 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 dvoumy agenty. Je zde ale možnost vybrat jaký algoritmus rozhodování je použit, jestli algoritmus PPO či náhodné chování.

Samotný kód hry je rozdělena na 3 částí.

- *GameController* [6]

  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* [6]

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

- *Hra* Samotná hra je následovně rozdělena na 2 podčástí která každá zpracovává jinou stránku hry.

  - *Herní logika* src/game/scotland_yard_game_logic.py Zpracovává herní mechanizmy, jako je pohyb, zpracování výherních podmínek, atd. Zprostředkovává informace pro prostředí 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 byl inspirován tutoriálovým projektem z platformy GitHub [6]. Kód byl avšak značne upraven a vylepšen aby vyhovoval integrování do této práce.

Jednotlivé vrstvy hry jsou tak izolovány a mohou být snadno vyměněny za jiné implementace.

3.0.3 Prostředí

Vytvořit prostředí pro platformu Ray.Rlib byl velký oříšek. Dokumentace vytváření prostředí není příliš komplexní, byl tedy problém zajistit správnou komunikaci prostředí a agentů. Prostředí je komplikovanější, jelikož zde figuruje více agentů a dvě rozdílne strategie (politiky).

Řešení zvolení nevalidních akcí

Hra Scotland Yard má omezené herní pole. Tím pádem je jasné, že pokud jsou hráči na okraji herního pole, nemohou zvolit takovou akci, která by je posunula mimo herní pole. K vyřešení tohoto problému se využívá tzv. maska akcí (*Action mask*). Framework Ray tuto možnost v omezené míře podporuje. Je avšak potřeba vytvořené prostředí obalit v obalové třídě, která tuto funkcionalitu zprostředkuje. Avšak nepodařilo se mi tuto funkcionalitu zkloubit s dalšímy požadavky systému. Mezi těmito požadavky je fungování více aktérů s různými politikami v jednom prostředí a různé pozorovací prostory agentů.

Politika, tedy může zvolit nevalidní akci, ale je velmi penalizována. Agenti ji tedy zvolí opravdu zřídka. Pokud se tak stane, je ve hře implementovaná funkcionalita, která se pokusi znovy vygenerovat akci, dokud 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 potřeba.

Systém odměn

Jednotlivým agentům jsou udělovány odměny na základě jejich chování. Jelikož policisté a Pan X mají odlišné protichůdné i jejich odměny jsou odlišné.

$$R_{\rho_p} = \sum_{i=1}^{3} [(\rho_{p_i} - \rho_{pref}) \div 3] \tag{3.1}$$

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

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

- 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_i} - vzdálenost policisty i 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.

Výpočet odměny policistů je složitější. Policisté znají poslední odhalenou pozici Pana X. Okolo tohoto bodu je vytvořená oblast, ve které se Pan X může nacházet. Policisté znají nejbližší bod této oblasti. Pokud je policista v této oblasti dostáva odměnu. Pokud je mimo tuto oblast zájmu, je penalizován. Čím blíže je policista k této oblasti tím menší penalizaci obdrží.

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

Pozorování agentů

Kapitola 4

Experimenty

Kapitola 5

Závěr

Kapitola 6

Přílohy

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.

[3] 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. DOI: 10.1371/journal.pone.0095693. Dostupné z: https://doi.org/10.1371/journal.pone.0095693.

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

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

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

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

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

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

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

[11] OpenAI. *OpenAI Five defeats Dota 2 world champions* [online]. 2019 [cit. 2024-03-18]. Dostupné z: https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer).

[12] 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. DOI: 10.1126/science.add4679. ISSN 1095-9203. Dostupné z: http://dx.doi.org/10.1126/science.add4679.

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

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