VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

STRATEGICKÁ HRA S NEURČITOSTÍ ZALOŽENÁ NA DESKOVÉ HŘE SCOTLAND YARD

A STRATEGY GAME WITH UNCERTAINTY BASED ON THE BOARD GAME SCOTLAND YARD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ROSTISLAV HUSA**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání diplomové práce

Student: **Husa Rostislav, Bc.**
Program: Informační technologie
Obor: Inteligentní systémy
Název: **Strategická hra s neurčitostí založená na deskové hře Scotland Yard**
A Strategy Game with Uncertainty Based on the Board Game Scotland Yard
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s pravidly deskových her, kde aktuální stav hry bývá pro hráče po většinu času utajený. Inspirujte se hrou "Scotland Yard", kdy pozice jedné z figur bývá protihráčům ukázána jen v některých kolech hry.
2. Navrhněte obdobnou hru, ve které herní pole může být oproti hře Scotland Yard zjednodušené, ale zachovejte princip skrývání figury ve většině tahů.
3. 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.
4. Pro jednotlivé role figur ve hře, to znamená jak pro figuru, která je hledána, tak pro figury, které ji hledají, implementujte algoritmy řízení a ověřte jejich schopnost plnit zadané cíle.
5. Vyhodnoťte úspěšnost obou stran hry pro různé míry zapojení metod strojového učení a diskutujte zjištěné výsledky.

Literatura:

1. Russel, S., Norvig, P.: Artificial Intelligence, A Modern Approach, Pearson, 2009
2. J.P.A.M. NijssenMark H.M. WinandsMark H.M. Winands: Monte-Carlo Tree Search for the Game of Scotland Yard,Computational Intelligence and Games (CIG), 2011

Při obhajobě semestrální části projektu je požadováno:

- První tři 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. listopadu 2021
Datum odevzdání: 18. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Tato práce řeší implementaci vlastní hry na principu her typu Scotland Yard. Součástí je několik verzí umělé inteligence pro obě strany hry s využitím strojového učení. Především neuronové sítě a Monte Carlo Tree Search. Obě jsou vyzkoušeny v několika variantách a porovnány vůči sobě navzájem.

Abstract

The subject of this thesis is creation of custom game using same principles as the game of Scotland Yard. Realization is including few versions of artificial intelligence for each player of the game using machine learning. Most importantly neural net and Monte Carlo Tree Search. Both are tested in several variants and compared against each other.

Klíčová slova

umělá inteligence, strojové učení, Monte Carlo Tree Search, MCTS, hluboké neuronové sítě, Scotland Yard, strategické hry, stolní hry, hry s neurčitostí

Keywords

artifical intelligence, machine learning, Monte Carlo Tree Search, MCTS, deep neural nets, Scotland Yard, strategy games, board games, games with uncertainty

Citace

HUSA, Rostislav. *Strategická hra s neurčitostí založená na deskové hře Scotland Yard.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, Ph.D.

Strategická hra s neurčitostí založená na deskové hře Scotland Yard

Prohlášení

Prohlašuji, že jsem tuto diplomovou 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.

.....
Rostislav Husa
17. května 2022

Poděkování

Chtěl bych poděkovat svému vedoucímu doc. Ing. Františku Zbořilovi, Ph.D za odbornou pomoc.

Obsah

Kapitola 1

Úvod

Stolní hry jsou součástí lidské civilizace už od antiky a slouží jako oblíbená kratochvíle i platforma pro poměřování důvtipu soupeřících hráčů. Právě to z nich dělá vděčný předmět zájmu umělé inteligence. Jasně definovaná pravidla a vymezené možnosti jednotlivých tahů se rozvíjejí do velkého množství konkrétních průběhů hry. Dávají také jasný cíl nalezení těch správných rozhodnutí, která povedou k vítězství. Příkladem, který se asi nejsilněji zapsal do obecného povědomí, jsou šachové programy. Ty se od svých skromných počátků zdokonalovaly až po porážku mistra světa Garryho Kasparova v roce 1997. I přes dosažené úspěchy ale tomuto oboru stále zbývá mnoho výzev.

Tato práce je inspirována asymetrickou stolní hrou Scotland [23]. V této hře jedna strana reprezentuje skupinu pronásledujících s početní převahou herních figur, ale pouze neúplnými informacemi o tazích druhé strany. Druhá strana reprezentuje prchajícího, který se snaží, aby i přes zanechávané stopy nebyl dostižen. Herním plánem je neorientovaný graf s více druhy hran a pravidly omezujícími možnosti pohybu figur. Vzhledem k rozsahu hry a elementu neurčitosti se jako vhodné nabízí využití strojového učení, především v podobě hlubokých neuronových sítí a metody Monte Carlo Tree Search (MCTS) [8], [20].

Cílem této práce je tedy navrhnout hru na principu Scotland Yardu a vytvořit několik variant umělé inteligence pro obě strany hry. Po jejich vytvoření a naučení pak tato práce jejich výsledky porovná vůči sobě navzájem, vůči existujícím řešením obdobné problematiky, a v neposlední řadě také proti živému hráči.

V kapitole 2 jsou vysvětleny obecné principy stolních her, přiblížen vliv neurčitosti a skrytých informací v nich. Dále jsou podrobněji rozebrána specifika hry Scotland Yard a navržena vlastní hra na těchto principech založená.

V kapitole 3 je představeno použití umělé inteligence a popsáno několik jejích technik se zhodnocením jejich vhodnosti pro problematiku, kterou tato práce řeší. S přihlédnutím ke specifikům hry jde především o metody využívající strojové učení.

V kapitole 4 je navržena reprezentace hry jako takové s několika konkrétními variantami umělé inteligence pro každou ze stran. Pro použití strojového učení jsou specifikovány metody získávání zpětné vazby z průběžných výsledků.

Kapitola 5 popisuje konkrétní implementaci jádra hry a řešení jednotlivých variant hráčské inteligence včetně případných úprav vedoucích k jejich zlepšení.

Kapitola 6 pak představuje a zhodnocuje testy různých řešení hráčské inteligence, porovnání jejich kvality v různých variantách a úspěšnost vůči sobě navzájem.

Kapitola 2

Stolní hry s neurčitostí

Společenské stolní hry zažívají především v posledních době velký rozmach popularity a s ním spojené snahy o neustálé inovace. Přesto lze říct, že v základu stále staví na podobných principech. Každá hra je definována systémem pravidel, která vytyčují možnosti jednotlivých hráčů a cíl, kterého se snaží dosáhnout, aby zvítězili (případně za jakých okolností remizují). Hráči se pak rozhodují, kterou z nabízených možností využít, aby vítězství dosáhli.

U některých her je každým z hráčů učiněno právě jedno rozhodnutí, ta jsou provedena oběma (všemi) hráči skrytě, nezávisle na sobě, a po odhalení vyhodnocena na základě stanovených pravidel. Tomuto říkáme hry v normální formě. Příkladem může být hra kámen-nůžky-papír.

Další běžnou variantou stolních her jsou hry sekvenční, kde se hráči v tazích střídají. Hra se tak ze svého výchozího stavu tahem prvního hráče posouvá do stavu nového. Zde přebírá rozhodnutí hráč následující a po vyhodnocení jeho tahu se předávání iniciativy opakuje dokud jeden z hráčů nevyhraje. Průběhy tohoto typu her lze vyjádřit rozhodovacím stromem, jehož rozsah se bude lišit podle komplexnosti konkrétní hry. Zatím co u piškvorek na mřížce 3x3 je s přihlédnutím k symetričnostem pouze 765 možných unikátních stavů hry, u šachu je to po odehrání dvou tahů každým z hráčů přes 70000.

Kromě samotného počtu možných stavů mohou hry využívat ještě další mechanismy. Jedním z nich je vyšší počet hráčů, který už sám o sobě zvyšuje variabilitu hry, protože každý hráč nepočítám jen s jedním tahem soupeře ale s vektorem kombinujícím tahy všech ostatních hráčů. Umožňuje také prvek kooperace. Ta přináší možnost dynamického vytváření koalic v průběhu hry, kdy hráči kromě možností vlastních tahů posuzují také možnosti spolupráce s dalšími hráči. Hry více hráčů, kde jsou kritéria kooperace od začátku pevně dána a v průběhu hry neměnná, mohou přispět k pospolitosti kolektivu. Ale z hlediska herních mechanismů vlastně komplexitu nezvyšují, protože na takovou kooperující skupinu se společným cílem lze pohlížet jako na jednoho hráče.

Doposud uváděné příklady představovaly hry, kde všichni hráči hrají podle stejných pravidel, se stejnými možnostmi tahů a stejnými podmínky dosažení vítězství. Snaží se tedy dosahovat stejných cílů a tyto hry považujeme za symetrické. Případná výhoda iniciativy prvního tahu by měla být co nejmenší a dále redukována tím, že se o určení začínajícího hráče rozhodne náhodně. To ale nemusí platit vždy. Další kategorií jsou hry asymetrické, kde možnosti herních tahů a podmínky dosažení vítězství mohou být pro každého z nich jiné. Příkladem budiž například skupina starých keltských her, souhnně známých pod názvem Tafl. Zde samozřejmě vyvstává otázka, nakolik jsou šance hráčů rovnocenné.

2.1 Vliv neurčitosti ve stolních hrách

Zatím uvedený rozbor implicitně předpokládal, že všichni hráči mají o hře úplné informace. Tedy, jsou jim známy všechny aspekty stavu hry a jsou jim rovněž známy i všechny možnosti ostatních hráčů. Pokud je hra zároveň i deterministická, znamená to, že úspěch hráčů záleží pouze na jejich schopnosti herní situaci správně vyhodnotit a učinit nejlepší možné rozhodnutní. Vyčíslitelná složitost hry se bude lišit podle konkrétních pravidel.

Protože lidský mozek není nezávislou výpočetní platformou, ale cíleně se zdokonaluje ve věcech, které procvičuje, z hlediska lidsky vnímané náročnosti hry pak platí, že rozhodujícím faktor je zkušenost. To z hlediska společenského vnímání stolních her nemusí být vždy žádoucí. Je-li jeden z hráčů schopen hrát výrazně lépe než druhý bude výsledek opakovaných her pro slabšího z hráčů demotivující. Zanesení vlivu neurčitosti může tento problém překonat tím, že dále zvyšuje komplexitu hry. Zároveň dává šanci méně zkušenému hráči zvítězit i při hraní slabší strategie.

Prvek náhody

Jedním ze zdrojů neurčitosti je prvek náhody. Ten je v klasickém pojetí teorie her reprezentován tak, že pro dané rozhodnutí hráče není výsledek pevně daný. Místo toho může rozhodnutí hráče vést na několik možných výsledků s různou pravděpodobností. A tedy, za předpokladu, že hráč má stále úplné informace o stavu hry, si může dopočítat předpokládaný užitek. Toto je varianta kdy hráč nejprve provedl rozhodnutí a následně se projevil prvek náhody.

V obecném pojetí stolních her ale často nachází využití i opačný mechanismus, kdy hráčův tah začíná vygenerováním nějakých náhodných možností. Z těch pak hráč během svého tahu vybírá. Okrajovovým případem je dětská hra Snakes and Ladders, kde o pohybu figury po herním plánu rozhoduje pouze náhoda, a o hráč ve své strategii neprovádí žádný výběr. Reprezentativnějším příkladem je hra vrhcáby, kde během každého tahu hráč nejprve hází kostkami, a pak se rozhoduje, kterými ze svých kamenů pohne o vzdálenosti které padly na kostkách. Vzhledem k tomu, že pravidla hry zahrnují mechanismy blokování pohybu a vyřazování soupeřových kamenů, není důležitý jen rychlý přesun do cíle, ale i zajištění výhodné pozice na herní desce tak aby vlastní kameny byly co nejméně ohroženy a naopak hru co nejvíce ztěžovaly soupeři. Hráč se tedy musí snažit předjímat přinejmenším soupeřův bezprostředně následující tah, jehož možnosti budou opět záviset na hodu kostkami. V lepším případně by pak měl hráč plánovat i svůj následující tah, případně možné pozice svých a soupeřových kamenů na několik tahů dopředu.

Samostatnou kategorii pak tvoří hry s balíkem karet, který je ve své podstatě nástrojem sekvence podmíněných pravděpodobností. Asi nejilustrativnějším příkladem je známý balík pokerových karet tvořený čtyřmi barvami, každá s třinácti hodnotami. Každá tažená karta je z balíku odebrána a nemůže být v rámci probíhající hry tažena znovu. Hráč tak na základě toho, jaké karty drží, případně karet, které sám odhodil, nebo byly odhozeny otevřeně, ví, které karty už v balíku nejsou. Na základě této informace může průběžně upravovat svoje očekávání ohledně pravděpodobnosti tažení konkrétní karty nebo jedné ze zbývajících karet které b požadovanými vlastnostmi. Díky tomu může vyhodnotit, jaké karetní kombinace má šanci dosáhout. Ohodnocení výsledné kombinace karet, které hráč získá, je pak nastaveno tak, že rámcově reflektuje pravděpodobnost jejich dosažení [1].

Skrytá informace

Dalším zdrojem neurčitosti ve stolních hrách je skrytá informace. To znamená, že stav hry je sice pevně daný, ale hráči o něm mají pouze neúplné informace. Příkladem může být výše zmíněný balík karet. Zde hráč ví, které karty táhl a případně i které byly zahozeny, nemá ale informace o tom, které karty drží ostaní hráči. Pokud hráč tento faktor nezohlední, může marně doufat v dobrání karty, která už v balíku není. Hráč se na základě dostupných informací samozřejmě může snažit vydedukovat skutečný stav hry. Například, že protihráč zahazující jednu sedmičku nemá žádné další, protože by si porušil dvojici. Zároveň ale takováto hráčova dedukce může být zavádějící. Například protihráč může mít čtyři karty stejné barvy včetně jedné sedmičky, a pátou kartu sedmičku jiné barvy zahazovat ve snaze dobrat pátou v barvě, což mu zajistí silnější ruku než by byla dvojice sedmiček [1].

Zajímavou variantou hry se skrytou informací je pak kooperativní karetní hra Hanabi, ve které je cílem odehrát náhodně rozdané karty ve vzestupném pořadí v jednotlivých barvách. Hráči ale drží karty lícem od sebe a sami hodnoty vlastních karet nevidí. Musí si tak navzájem vhodně napovídat indiciemi barvy a hodnoty ve snaze odehrát všechny sekvence s omezeným množstvím nápověd.

U her s balíkem karet dochází ke kombinaci skryté informace (v podobě karet, které byly rozdány, ale hráč je nevidí) s prvkem náhody (v podobě karet, které teprve budou z balíku taženy). Mechanismus skryté informace ovšem může být využit i samostatně u her které jsou zcela deterministické. Ty lze dále rozdělit do tří kategorií: nejjednodušší v podobě statické skryté pozice figur, jak představuje hra Námořní bitva. V té hráči rozmístí své figury na mřížce dané velikosti a následně se střídají v tazích, ve snaze odhalit pozice figur protihráče. Další možností je skrytý nebo částečně skrytý stav herní desky a s ním i skrytí pohybu soupeřových figur na ní. Jako příklad může posloužit šachová varianta "Dark chess" kde má hráč informace pouze o polích na kterých se jeho figury nacházejí nebo se na ně mohou v následujícím tahu pohnout. Ostatní pole mu zůstávají skryta. Zde je vhodné podotknout, že před zapojením výpočetní techniky byly fyzické realizace tohoto typu her krajně nepraktické. Často vyžadovaly přítomnost dalšího účastníka v roli arbitra či rozhodčího, který si udržoval přehled o skutečném stavu hry a zprostředkovával každému z hráčů právě jen ty informace, které má vedět.

Poslední možností je hra se známou informací o pozici ale skrytými typy figur. Toto dobře ilustruje hra Stratego (u nás známá jako Maršál a špión). V této hře mají oba hráči k dispozici stejný počet figur s hierarchií vojenských hodností, několik zvláštních figur (miny, sapéry, špióna) a jednu figuru vlajky. Typ figury je skrytý tak, že ho vidí pouze ovládající hráč. Po úvodním rozmístění figur, které může být provedeno v rámci startovního prostoru libovolně, se hráči střídají v pohybu figurami. Pokud hráčova figura vstoupí na pole, kde už je soupeřova figura, jsou obě odhaleny a střetnou se. Vyšší vojenská hodnost vyřazuje nižší, případně se aplikují pravdila zvláštních figur. Střetnou-li se dvě stejné figury jsou vyřazeny obě. Cílem je vyřadit soupeřovu vlajku které je nehybná. Při nejběžnější verzi hry na desce velikosti 10×10 se 40 figurami na stranu je velikost stavového prostoru hry přibližně 10^{115} [2]. Hráč se o konkrétních typech soupeřových figur dozvídá pouze ze střetů, a vliv skryté informace pro plánování vhodné strategie je tedy zcela zásadní.

2.2 Hry typu Scotland Yard

Nejvhodnějším příkladem hry typu Scotland Yard je samozřejmě hra Scotland Yard [23]. Na základě klasifikací uvedených v předchozí sekci může být definována jako hra sekvenční a asymetrická, kde jednu stranu tvoří pevně daný tým se společným cílem. Druhá strana je reprezenotvána jedním hráčem, který do hry vnáší prvek neurčitosti ve formě skryté informace. Početnější strana reprezentuje skupinu pronásledovatelů a hráč hrající skrytě reprezentuje prchajícího. Fyzická realizace hry tedy nenaráží na problém výměny právě jen těch nutných informací, který mívají některé symetrické hry se skrytou informací jak bylo zmíněno dříve v sekci 2.1. Toho je dosaženo tím, že jedna strana hraje otevřeně, a může tedy vše reprezentovat přímo na herním plánu. Prchající se naopak, až na níže zmíněné výjimky, na herním plánu nereprezentuje. Prchající skrývá jedinou informací, a to svoji pozici, která je zaznamenávána v jeho cestovním deníku. Případně dopadení přizná sám prchající a může být ověřeno z jeho cestovního deníku. Na straně pronásledujících pak vzhledem k jejich alianci platí, že sledují společný cíl a snaží se při jeho dosažení co nejlépe spolupracovat. To může ze společenského pohledu do hry přinášet jisté komplikace, z technického hlediska ale můžeme uvažovat, jako by celá strana pronásledujících byla jeden hráč s více figurami.

Tématicky je hra zasazena do Londýna, herní plán tedy zjednodušeně reprezentuje ulice v jeho středu a klíčové zastávky na nich, viz obrázek 2.1. To hernímu plánu dává netriviální strukturu s malým množstvím symetrií nebo opakujících se podsekcí. Pohyb figur po herním plánu je limitován tím, že hrany spojující jednotlivá pole jsou zařazeny do kategorií odpovídající různým typům dopravy, a každá figura pronásledovatelů disponuje pouze omezeným množstvím lístků pro použití každého z nich. Po vyčerpání lístků pro daný typ dopravy už ho tato figura nemůže dále používat, a v krajním případě už se nemůže pohnout vůbec. Prchající začíná s podobným omezením, u něj ovšem není tak zásadní, neboť navíc přebírá zdroje použité pronásledovateli. Na druhou stranu je prchající znevýhodněn tím, že ve stanovených okamžicích musí svoji pozici odhalit. Informace o tom, jaké dopravní prostředky prchající používá, je rovněž veřejná. Pronásledovatelé tedy mohou na základě poslední známé pozice a druhů dopravy které prchající použil usuzovat na jeho možné pozice.

Od prvního uvedení hry v roce 1983 vzniklo několik dalších edic. U nás nejznámější je asi Fantom staré Prahy která zasazuje pronásledování na zjednodušený plán Prahy počátku 20. stoletní. Kromě změny herního plánu a rozdílné terminologie je hra velice podobná [14]. Existuje také varianta N.Y. Chase zasazená na Newyorský Manhattan, která ale poněkud degraduje složitost herního plánu tím, že vzhledem ke specifické povaze urbanizace této oblasti řada sekcí mapy tvoří pravidelnou mřížku.

Další hrou využívající podobné mechanismy je Fury of Dracula (u nás známé jako Běsnění Drákuly), která pronásledování zasazuje na pozadí známého románu Brama Stokera. Jako herní plán zde slouží mapa Evropy s jejími důležitými železničními spojnicemi v odpovídající historicé éře. Tato hra ale využívá jiné principy pohybu pro prchajícího a pronásledující, a navíc zavádí řadu dalších mechanismů nad rámec hledání pozice pronásledovaného. Například že prchající v případě dostižení není automaticky poražen a v závislosti na dalších faktorech hry musí být obvykle dostižen několikrát [9].

Pro potřeby této práce jsem zvážil případné zjednodušení pouze na část pronásledování, bohužel kombinace dílčích odlišnosti v mechanismech pohybu neumožňují udělat to s jistotou kvalitního výsledku.

Pravidla hry Scotland Yard

Hra probíhá na herním plánu o 199 polích, viz. obrázek 2.1. Jednotlivá pole jsou různě propojena až čtyřmi druhy hran reprezentujícími čtyři možné druhy dopravy mezi nimi – taxi, autobus, metro a loď. A 18 z těchto polí je také možným startovním místem pro jednotlivé hráčské figury.

Hry se účastní dvě strany. Tou první je osamocený prchající (zde nazýván Mister X, či Mr. X). Druhou stranou je skupina dvou až pěti pronásledujících (zde nazýváni Detektivové). Na začátku hry je každý hráč vybaven daným množstvím lístků na dopravu:

- Každý z Detektivů má 10 lístků na taxi, 8 lístků na autobus, a 4 lístky na metro
- Mr. X má 4 lístky na taxi, 3 lístky na autobus, 3 lístky na metro, 2 lístky dvojitého pohybu, a tolik černých lístků kolik je ve hře Detektivů

Každý lístek umožňuje jedno využití daného druhu dopravy, tedy přesun z pole, na kterém se figura nachází, na jedno z polí, které je s ním daným druhem dopravy přímo spojeno. Lístky, které použili Detektivové, dostává Mr. X. Lístky, které použil Mr. X, jsou vyřazovány ze hry. Černé lístky pak umožňují použití libovolného druhu dopravy, včetně lodě (což tedy může pouze Mr. X).

Po náhodném rozlosování startovních polí umístí každý z Detektivů svoji figuru na příslušné pole na mapě. Mr. X figuru neumisťuje a svoje pole pouze zaznamená do cestovního deníku. Hru začíná Mr. X a jeho tah spočívá v tom, že do cestovního deníku zapíše nové pole, na které se přesouvá, a překryje ho odpovídajícím lístkem. Detektivové tedy (s výjimkou uplatnění černého lístku) ví jaký způsob dopravy použil. Následně provede svůj tah každý z detektivů a uplatněním příslušných lístků se pohnou.

Detektivové se vždy hýbou ve stejném pořadí, a na žádném poli nesmí nikdy být více než jeden Detektiv. Platí také, že se žádný z hráčů nesmí svého tahu zříct a pokud je schopen se nějak pohnout, musí tak učinit. A to i pokud by to pro něj bylo nevýhodné. Pokud je hráči pohyb znemožněn (například Detektiv který se ocitl na poli odkud lze cestovat jen pomocí taxi pro které už vyčerpal všechny svoje lístky) zůstává v jeho tahu figura na místě a ostatní normálně pokračují ve hře.

Mr. X se po většinu hry pohybuje skrytě, ale na 3, 8, 13 a 18 kroku své cesty svou pozici musí zveřejnit. Na druhou stranu má Mr. X k dispozici dva lístky dvojitého pohybu, které může využít, aby po provedení obvyklého pohybu ve svém tahu provedl okamžitě pohyb další. Oba pohyby se řídí obvyklými pravidly a pro každý musí odevzdat správný lístek.

Konec hry nastává když:

- Jeden z Detektivů se nachází na stejném poli jako Mr. X (vítězství Detektivů).
- Žádný z Detekivů už se nemůže pohnout (vítězství Mr. X).
- Mr. X odehrál 22 tahů (24 pohybů) aniž by byl dopaden (vítězství Mr. X).

S tím, že poslední varianta je ve skutečnosti jen specifickým případem varianty předchozí. Vzhledem k omezenému množství lístků, které mají Detektivové k dispozici, je zaručeno, že po 22 tazích již možnost pohybu mít nebudou.

Pokud by došlo k tomu, že se Mr. X pohne na pole, kde už se nachází jeden z Detektivů, je rovněž dostižen. Nemůže toto pole "přeskočit" dvojitým pohybem. Na konci hry Mr. X zveřejní svůj cestovní deník k ověření že všechny pohyby byly platné [23].

Obrázek 2.1: Herní plán hry Scotland Yard [24] možné startovní pozice jsou:
13, 26, 29, 34, 50, 53, 91, 94, 103 ,112, 117, 132, 138, 141, 155, 174, 197, 198

2.3 Návrh hry na principu Scotland Yard

Pro tento projekt navrhuji vlastní variantu hry na principu hry Scotland Yard. Účelem je zachovat stejné mechanismy na kterých hra staví. Tedy po většinu hry skrytý pohyb prchajícího, početní převahu pronásledujících, více možných druhů dopravy, a na straně pronásledujících určitý limit jejich používání. Jako základ mapy poslouží jihovýchodní roh mapy Scotland Yard s 33 poli, tedy jedna šestina mapy původní.

Pro určení počtu pronásledujících vyzkouším variantu se dvěma i třemi. Protože řada polí má právě tři odchozí cesty, úplné obklíčení prchajícího ve dvou nebude možné, ve třech už ano. Což by na dané mapě mělo dělat zásadní rozdíl v jejich úspěšnosti.

Na mapě jsou možné dva druhy dopravy - taxi a autobus, rovněž kopírující linky tak, jak byly definovány na mapě původní, ovšem s přečíslováním stanic do intervalu 1 až 33, viz. obrázek 2.2. Na mapě bude 6 startovních pozic, zvolených tak, aby každé dvě od sebe dělily alespoň tři pohyby. Ekvivalentní redukce na jednu šestinu z původních 18 nepřichází v úvahu. Pro variantu dvou pronásledovatelů by mohli na základě své startovní pozice rovnou odvodit startovní pozici prchajícího a pro variantu tří by vůbec nešlo umístit všechny figury.

Ostatní pravidla pohybu a vítězství jednotlivých stran zůstavají zachována, tak, jak byla definována v předchozí sekci. Délka hry bude zredukována na 18 kol s tím, že prchající bude svou pozici zveřejňovat v kolech 2, 6, 10 a 14. Mechanismus odevzdávání lístků prchajícímu zůstane zachován. Zredukované délce hry bude samozřejmě odpovídat i úvodní distribuce lístků:

- Každý z Detektivů dostane 10 lístků na taxi a 8 lístků na autobus
- Mr. X dostane 4 lístky na taxi a 4 lístky na autobus

V otázce vynechání nebo zachování mechanismů černých lístků a dvojitého pohybu na straně prchajícího jsem přihlédl k existujícím pracím, které se zabývaly problematikou Scotland Yardu. Zatím co některé z těchto prací tyto mechanismy zachovávají, případně se snaží i zefektivnit jejich využívání zavedením dodatečných rozhodovacích pravidel na straně pronásledovaného [3, 20], v jiných pracích jsou oba typy lístků zcela vynechávány a i přesto hra zůstává dostatečně komplexní problematikou [8]. Rozhodl jsem se tedy také přistoupit k jejich vynechání.

K redukci herního plánu na čtvercovou, nebo jiným způsobem pravidelnou, mřížku ani ke zjednodušení pohybu na jeden typ dopravy přistoupuji. V takové verzi je sice hra stále vhodnou platformou k ověření možných přístupů strojového učení k jejímu řešení [26], ale pravidelnost herního plánu může vést k výskytu symetrických vzorů, které nalezení vhodných strategií pohybu dost výrazně zjednodušují.

Pokud se zamýšlená verze hry ukáže být jako příliš jednoduchá nebo hledání co nejlepších herních strategií povede na zcela jednostranné výsledky (jak by mohlo hrozit v případě hry tří pronásledujících) je další možností rozšíření herního plánu na polovinu původní desky Scotland Yardu, tedy 100 polí. Jako další rozšíření uvažuji i zavedení tramvajové dopravy, opět co nejblíže původnímu rozsahu na hrací ploše. V takovém případě už by mělo být v pořádku navýšení počtu pronásledujících na 4 a případně prodloužení herní doby na původních 22 tahů. Záměrem práce není porovnávat mezi sebou jednotlivé varianty hry, ale najít vhodnou variantu hry, na které bude možné efektivně porovnat jednotlivé přístupy hledání hráčských strategií.

Obrázek 2.2: Herní plán vlastní hry založený na výseku sekce mapy Scotland Yard [24]
možné startovní pozice: 1, 5, 15, 19, 22, 30

Kapitola 3

Metody umělé inteligence

Tato kapitola poskytuje obecný přehled některých metod umělé inteligence, a to od jednoduchých, které lze použít, když je možné deterministicky ohodnotit celý stavový prostor, až po složitější, schopné vypořádat se s neúplností informace, či s faktorem náhody. V první části je předpokladem hra dvou hráčů střídající se v tazích kde jsou jasně dány jak podmínky výhry a prohry, tak i ohodnotitelnost jednotlivých průchodů stavovým prostorem.

Dále jsou představeny metody využívající různé formy strojového učení které, nezbytně nevyžadují podrobnou definici všech herních mechanismů, ale na základě pravidel výhry a prohry se dokáží opakovaný hraním a získáváním zpětné vazby z předchozích průběhů zdokonalovat.

AND/OR grafy

Tento algoritmus je nejobecnější možnou abstrakcí sekvenční hry dvou hráčů a vyjadřuje její průběh z pohledu aktivního hráče jako stromový graf, kde se na jednotlivých úrovních zanoření střídají AND a OR uzly. Listy na nejnižší úrovni grafu představují konkrétní stavy, kde došlo k výhře nebo prohře aktivního hráče. Z pohledu aktivního hráče je rozhodnutí o jeho vlastním tahu (tedy včetně kořene stromu) uzlem typu OR. Pokud je hráč na tahu, stačí mu, aby z možných potomků OR uzlu vedl k výhře jediný, protože si mezi nimi může vybrat. Tahy protihráče jsou naopak uzlem typu AND. Protihráč se snaží aktivnímu hráči ve výhře zabránit a ze svého uzlu vybírá ten, který k výhře aktivního hráče nevede. Tah protihráče je tedy z pohledu aktivního hráče vítězný pouze tehdy, pokud k výhře aktivního hráče vedou všechny možnosti které protihráč má. Jinými slovy pro řešitelnost AND uzlu je potřeba, aby protihráč nemohl žádou ze svých možností ve výhře zabránit. Jednoduchost principu tohoto algoritmu je také příčinou jeho zásadní nevýhody, což je nutnost vyhodnocovat všechy podstromy grafu až do listů [25].

Mini-Max

Pokročilejší variantou předchozího přístupu je algoritmus Mini-Max, který místo ohodnocování možných výsledků pouze jako výhra či prohra zavádí číselné ohodnocení stavů. Ve stromovém grafu se pak na jednotlivých úrovních střídají uzly Minima a Maxima. Uzly aktivního hráče jsou typu Maximum, aktivní hráč se snaží maximalizovat získané ohodnocení a vybírá tedy ve svém tahu potomka s nejvyšším ohodnocením. Uzly protihráče jsou naopak typu Minimum, protihráč se snaží aktivnímu hráči dovolit pouze co nejnižší ohodnocení a vybírá tedy potomka, který aktivnímu hráči přinese co nejméně.

Číselné ohodnocení stavů je prováděno heuristickou funkcí. Ta pro stav výhry a prohry dává nejvyšší, respektive nejnižší, možné hodnoty z možného rozsahu, je ale schopna ohodnotit i všechny ostatní stavy hry. Toto hodnocení může mít různou časovou složitost od prostého součtu zbývajících kamenů na desce či vzdálenosti figur do cíle, po složitě vyhovované výpočty. Každopádně musí zůstat vyčíslitelné pro každý stav v rozumném výpočetním čase.

Zásadní výhodou oproti AND/OR grafům je, že není nutné vyhodnocovat celý Mini-Max strom až do listů. Po průzkumu do požadované hloubky (případně do časového limitu při prohledávání do šířky) lze na uzly aplikovat ohodnocovací heuristickou funkci a její výsledek propagovat zpátky k rodičovskému uzlu. I tento přístup ale může být dost limitující, protože předpokládá ve všech větvích stromu průzkum do stejné hloubky. V závislosti na průměrném faktoru větvení (určujícím kolik potomků má v průměru každý uzel ve stromě) může počet vyhodnocovaných uzlů narůstat tak rychle, že algoritmus může jít jen do malé hloubky [2].

Alfa-Beta prořezávání

Další zefektivnění přináší algoritmus Alfa-Beta prořezávání. Jeho základní myšlenkou je snaha redukovat množství prohledáváných uzlů vyřazením těch, u kterých lze s jistotou říct, že nikdy nebudou vybrány. To může vést k zásadní redukci faktoru větvení mezi uzly které jsou skutečně prohledávány oproti celkovému počtu uzlů a tedy umožnit ve stejném čase prohledání do větší hloubky.

V průběhu prohledávání stromu jsou průběžně aktualizovány pomocné hodnoty α a β. S tím, že α udržuje nejnižší možný výsledek který je aktivnímu hráči zaručený při snaze vybírat ve svých tazích co nejvyšší hodnoty, zatímco β udržuje nejvyšší možný výsledek který protiháč dovolí při snaze vybírat ve svých tazích co nejnižší hodnoty. Inicializovány jsou na $\alpha = -\infty$ a $\beta = \infty$, respektive minimum a maximum ohodnocovacího rozsahu.

Vyhodnocení nejlepší varianty v daném stromu probíhá prohledáváním do hloubky (respektive omezené houbky nastavené jako parametr prohledávání) a před zanořením na nižší úroveň je vždy ověřováno splnění podmínky $\alpha < \beta$. Pro tahy aktivního hráče je hodnota α aktualizována jako maximum mezi stávající hodnotou α a hodnotami vrácenými prohledáním dostupných podstromů. Po dokončení prohledání všech podstromů aktivního uzlu nebo při porušení podmínky $\alpha < \beta$ je aktuální hodnota α vrácena výše. Pro tahy protihráče je naopak pro aktualizaci hodnoty β vybíráno minimum mezi stávající hodnotou β a hodnotami z prohledávaných podstromů [27].

Porušení podmínky $\alpha < \beta$ v podstatě znamená, že někde v již prozkoumané části stromu existuje pro hráče lepší alternativa. V tom případě nemá cenu prohledávat další podstromy aktivního uzlu, a algoritmus se může vrátit o úroveň výše. Případně skončit pokud už je v kořenovém uzlu.

Efektivita Alfa-Beta prořezávání v redukci faktoru větvení se může dost výrazně lišit s ohledem na to, které podstromy jsou prohledány nejdříve. V ideálním případě může redukovat faktor větvení b až na $\sqrt{b}$ a tedy při vynaložení stejného výpočetního úsilí prodloužit hloubku prohledávání na dvojnásobek oproti použití Mini-Maxu. V praktických případech obvykle přináší redukci na $\sqrt[4]{b^3}$, tedy prodloužení hloubky o třetinu. Pokud ale existuje predikce pro to, které podstromy přinesou největší změnu výsledku ohodnocovací heuristiky, je vhodné prozkoumávat nejdříve ty, prořezávání se tak může výrazně přiblížit své ideální efektivitě [25].

3.1 Metody strojového učení

Základními myšlenkami strojového učení jsou schopnost aplikace řešit úlohy, bez toho, aby musel být celý postup explicitně naprogramován, a princip postupného zdokonalování v řešení konkrétní úlohy jejím opakováním, podobně jako dochází k učení u lidí. Zásadním předpokladem efektivního strojového učení je dostatečné množství vstupních dat. Ta mohou být daným vstupem algoritmu nebo průběžně generována.

Zároveň je zde ale riziko efektu přeučení, kdy algoritmus začne docházet k výsledkům, které sice velmi přesně pasují na trénovací data, ale ztrácejí schopnost zobecňovat. Tomu lze bránit rozdělením vstupních dat na trénovací a testovací sadu, obvykle v poměru 9:1 nebo 8:2, kdy je každých několik cyklů učení na trénovacích datech proloženo ověřováním na datech testovacích. Dokud stoupá přesnost na trénovacích i testovacích datech, je pravděpodobné, že učení probíhá správně. Pokud se přesnost na trénovacích datech zvyšuje, ale na testovacích klesá, zřejmě dochází k přeučení a je vhodné učení zastavit nebo ho nějakým způsobem korigovat [25].

V závislosti na tom, jakým způsobem učení probíhá, rozlišujeme tři kategorie učení.

Učení s učitelem

Prvním příkladem je učení na trénovací sadě, kterou tvoří množina dvojic možných vstupů s napárovanými očekávanými výstupy. Snahou algoritmu je najít souvislosti a pravidla, která umožní na základě natrénování zařadit nový vstup na nejlépe odpovídající výstup. Toto umožňují algoritmy vytváření rozhodovacích stromů, pro jejich efektivitu je snaha v každém kroku rozhodování vybírat takové atributy, které přinášejí co největší informační zisk.

Učení bez učitele

Pokud nemáme pro množinu vzorových vstupů definované očekávané výstupy, je zapotřebí jiný přístup. Strojové učení musí samo hledat pravidla podobnosti pro vhodné shlukování. Tuto úlohu si lze představit jako n-rozměrný prostor tvořený jednotlivými atributy trénovacích dat, do kterého jsou jednotlivé záznamy zanášeny jako body. Algoritmus se pak snaží o hledání co nejbližších shluků těchto bodů.

Posilované učení

Posilované učení využívá místo trénovací sady dat principu ohodnocování zpětnou vazbou. Algoritmus nejprve provádí rozhodnutí náhodně (na slepo), nebo s využitím jednoduché heuristiky. Pak u každého průchodu na základě toho, jestli vedl k úspěchu nebo neúspěchu dostává odměnu nebo penalizaci, která je přiřazena k provedeným rozhodnutím. Při dalších průchodech jsou pak výsledky předchozích průchodů zohledňovány, a volby, které vedly k odměně, získávají postupně preferenci.

Tento proces musí být pozvolný, aby při výskytu více možných cest k úspěchu zvítězila ta nejefektivnější, což by se nestalo, pokud algoritmus dá hned silnou preferenci první cestě na kterou při náhodných průchodech narazí [25].

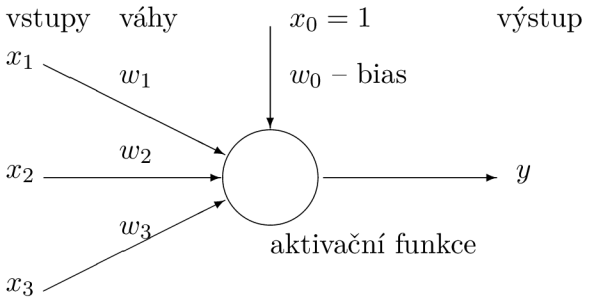
3.2 Neuronové sítě

Neuronové sítě jsou populární a v dnešní době široce uplatňovanou variantou strojového učení. Princip jejich fungování je založen na napodobení funkce biologických neuronů, stavebních prvků lidského mozku, a způsobu, kterým mezi sebou komunikují. Neuronové sítě jsou vhodné pro práci s neurčitostní a využití nacházejí například ve zpracování hlasu, rozpoznávání obrazů, a v neposlední řadě v realizaci umělých inteligencí počítačového soupeře pro různé hry.

Výpočetní model neuronu

Výpočetním modelem neuronu je perceptron. Perceptron může mít libovolné množství vstupů x_n, které přicházejí z vnějšku nebo z jiných neuronů. Každému ze vstupů je přidělena váha w_n obvykle v rozsahu $w_n \in \langle -1; 1 \rangle$, tyto váhy určují, jak velký vliv mají jednotlivé vstupy na výstup neuronu. Ze vstupů a vah je vypočten vnitřní potenciál neuronu ξ, konkrétní výpočet definuje bázová funkce neuronu, což je obvykle prostý váhovaný součet všech vstupů, tedy skalární součin vektoru vstupů a vektoru vah [19].

Jak je generován výstup pak určuje aktivační funkce neuronu, nejjednodušší variantou je skoková aktivační funkce, ta pracuje s dalším parametrem neuronu, což je práh, kterého musí vnitřní potenciál dosáhnout, aby neuron aktivoval svůj výstup. Místo porovnávání prahu s vnitřním potenciálem je obvykle práh zohledněn už při výpočtu vnitřního potenciálu jako takzvaný bias (předpětí), který je zápornou hodnotou prahu. Bias je realizován jako váha w_0 připojená na vstup x_0 s fixní hodnotou $x_0 = 1$. Vnitřní potenciál $\xi = \sum_{i=0}^{n} x_i w_i = \vec{x}\vec{w}$ pak stačí porovnávat s nulou a konfigurace neuronu je vyjádřena jeho vektorem vah $\vec{w}$.



Obrázek 3.1: Schéma perceptronu

Výchozí váhy neuronu mohou být nastaveny libovolně. Učení neuronu pak probíhá na sadě vstupních vektorů a odpovídajících očekávaných výstupů. Ty jsou cyklicky procházeny s tím, že pro ty, které byly správně zařazeny, jsou váhy neuronu zachovány. Pro vstupy které nedaly očekávaný výstup dochází ke korekci vah neuronu podle koeficientu učení μ. Korekce mohou probíhat po každém nesprávném výstupu nebo dávkově po zpracování celé sady vstupů z vyhodnocení všech jejich výstupů. Tento cyklus se opakuje dokud nedojde k úspěšnému průchodu celé sady bez nutnosti korekce [17].

Zásadním omezením neuronu s lineární bázovou funkcí je schopnost řešit pouze lineárně separovatelné úlohy, samostatný perceptron tedy nedokáže řešit například logickou funkci XOR a případná snaha o učení na ní nikdy neskončí. I tak se ale jedná o zásadní prvek strojového učení a stavební kámen rozsáhlejších řešení.

Neuronové sítě

Řešení složitějších úloh lze realizovat vzájemným propojováním neuronů do neuronových sítí, kde výstupy jednotlivých neuronů mohou být přivedeny na vstupy dalších. Způsoby propojení dělí neuronové sítě do dvou kategorií. Tou první jsou sítě acyklické, kde propojení neuronů tvoří posloupnost vrstev. Vstupem první vrstvy jsou vnější vstupy neuronové sítě jako takové, vstupem druhé vrstvy pak vstupy sítě a výstupy z první vrstvy, a tak dále. Výstupem sítě jsou pak výstupy vybraných neuronů z libovolných vrstev. Toto zapojení znamená, že výstup z vyšší vrstvy nikdy neovlivňuje vrstvu nižní a žádný neuron v tomto zapojení není ovlivňován vlastním výstupem, a to ani přeneseně přes jiné neurony [25].

Specifickou variantou sítě acyklické je pak síť dopředná, kde jsou mezi sebou propojeny pouze bezprostředně následující vrstvy. Vstup celé sítě považujeme za výstup pomyslné nulté vrstvy, která tvoří vstupy pro první vrstvu. Vstupy druhé vrstvy jsou pak tvořeny pouze výstupy první vrstvy, a tak dále. Výstupem sítě jsou pak výstupy vrstvy poslední. Dopředné sítě bývají obvykle realizovány jako plně propojené, tedy každý z výstupů předchozí vrstvy je propojen na všechny vstupy vrstvy následující. Stále zde platí, že všechny vstupy neuronů jsou váhovány a každý neuron má nastavený bias. Plně propojené dopředné sítě mohou být efektivně vyhodnocovány sekvencí skalárních součinů, případně maticovými operacemi.

Druhou kategorií neurovoných sítí jsou sítě rekurentní, kde výstupy jednotlivých neuronů mohou být přiváděny zpět na jejich vstupy, ať už přímo nebo smyčkou přes další neurony. To může být realizováno jako plné propojení všech neuronů se všemi, nebo to může být síť s vrstvami, kde jsou mezi sebou cyklicky propojeny jen neurony v rámci každé vrstvy, případně i jiné zpětnovazební principy. Například Hopfieldova síť která pracuje jako asociativní paměť.

V neuronových sítích nemusí být vhodné používat jednoduchou skokovou aktivační funknci. Místo toho je často vhodnější aktivační funkce spojitá, jako například sigmoidální $y = \frac{1}{1+e^{-\xi}}$ která pokrývá spojitě rozsah od $y \in (0; 1)$ a zároveň má největší sklon kolem středu svého rozsahu. To je výhodou při učení sítě protože "nerozhodné" neurony s výstupem kolem 0,5 mohou být snadněji ovlivněny, zatím co ty s výstupem blížícím se okrajům intervalu jsou stabilnější a je jižší pravděpodobnost, že budou ovlivněny malou změnou v předchozích vrstvách [19].

Učení dopředné neuronové sítě

Učení acyklické dopředné plně propojené sítě probíhá na podobném principu jako učení jednotlivého perceptronu. Je zapotřebí dostatečného množství trénovacích dat, sestávajících ze vstupů a jim odpovídajícím výstupů. Požadovaným výstupem může být aktivace právě jednoho konkrétního neuronu (tzv. hot-one encoded) při použití ke klasifikaci do tříd, případně obecně aktivace nějaké kombinace výstupních neuronů.

Pro každý ze vstupů je očekávaný výstup porovnáván se skutečným. Na základě jejich rozdílu v každém z výstupních neuronů může být spočítána chybová funkce jako suma druhých mocnin těchto rozdílů. Průměr chybových funkcí napříč všemi trénovacími daty pak udává, jak dobře nastavení sítě pasuje na trénovací data.

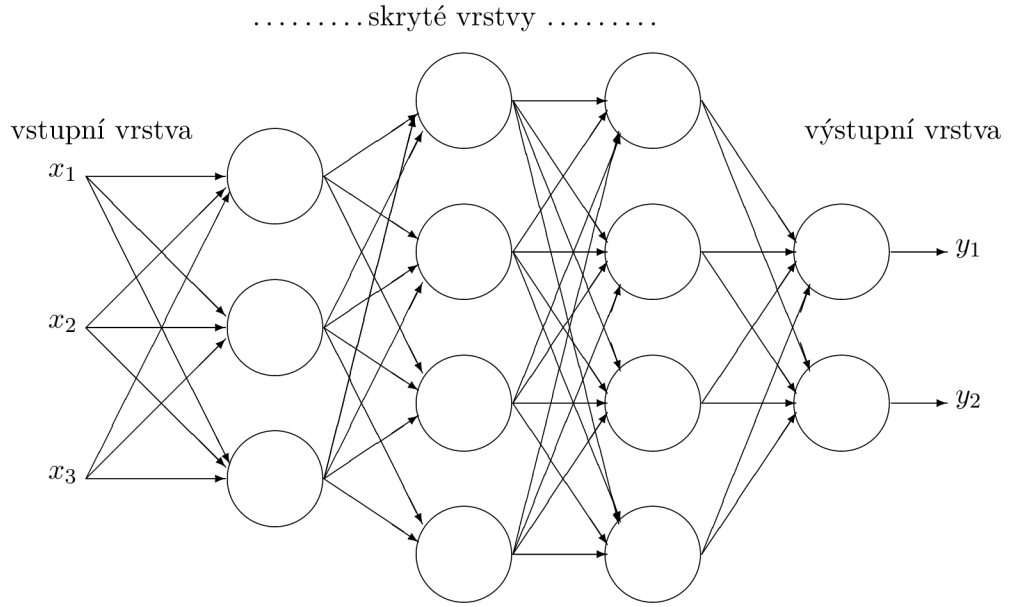
Odchylka od očekávaných výstupů také pro každý z výstupních neuronů určuje jeho požadovanou korekci. Provedení korekce vícevrstvých sítí už ale není tak přímočaré jako u učení jednoho perceptronu, kromě úpravy vah samotných výstupních neuronů je potřeba také úprava výstupů neuronů v předchozích vrstvách, a to úměrně k tomu, jak přispívají vrstvě následující.

K realizaci potřebých úprav konfigurace sítě se používá algoritmus backpropagation, který na základě potřebých korekcí napříč všemi trénovacími daty hledá s využitím prvních derivací takzvaný gradientní sestup, sloužící k nalezení těch vah napříč celou sítí, jejichž úprava přinese změnu ve výstupních neuronech s největšími odchylkami od požadovaného výstupu a zároveň bude mít jen malý vliv na neurony, jejichž výstup se blíží tomu požadovanému blíží [19].

Hluboké neuronové sítě

Na vícevrstvé neuronové sítě můžeme nahlížet také tak, že každá vrstva ve svých vstupech hledá dílčí vzory a extrahuje jejich příznaky. Informace o těchto vzorech pak předává vrstvě další, která v nich hledá vzory komplexnější. Zřejmě největší síla strojového učení spočívá právě v tom, že vhodně dimenzovaná síť může při učení pro danou úlohu nalézat vzory, které by člověk jen obtížně a zdlouhavě definoval, případně by je nebyl schopen nalézt vůbec [10].

Tohoto principu se využívá u hlubokých neuronových sítí, kde na vstupní vrstvu přivádíme zadání problému, na výstupu očkáváme odpověď a vnitřní logika řešení problému ve skrytých vrstvách je ponechána na učícím alogoritmu. Otázkou zůstává, co pro daný problém znamená vhodné dimenzování sítě. Vstupní vrstva by měla pokrývat všechny relevatní informace, ale přesný počet a dimenze skrytých vrstev může být obtížné odhadnout a často může být potřeba hledat kompromis mezi komplexitou sítě a jejím trénovacím časem experimentálně [25].



Obrázek 3.2: Hluboká neuronová síť se 3 skrytými vrstvami

Specifickou variantou hlubokých neuronových sítí jsou pak sítě konvoluční. Kromě neuronových vrstev obsahují také vrstvy konvoluční, jejichž vstup je interpretován jako dvourozměrná matice, ze které je aplikací konvolučního filtru (kernelu) extrahována menší matici příznaků. To může být pro některé aplikace efektivnější než další neuronová vrstva, uplatnění nacházejí například ve zpracování obrazu [12].

3.3 Monte Carlo Tree Search

Další ze zajímavých metod umělé inteligence je Monte Carlo Tree Search (MCTS). Mezi její přednosti patří dobrá efektivita pro celou řadu uplatnění bez potřeby rozsáhlých expertních doménově specifických znalostí konkrétní problematiky a flexibilní doba běhu algorimu. Díky tomu našla tato metoda využití mimo jiné pro strategické tahové hry [7] [20]. Je rovněž aplikovatelná pro hry s prvkem neurčitosti a skrytou informací, byť zde už je potřeba začlenění určitých doménově specifických znalostí které pomohou skrytou informaci zpracovat [5]. Metody na principu Monte Carlo mají úspěch dokonce i u strategických počítačových her v reálném čase, kde na určité úrovni abstrakce dokáží překonat běžně používané metody umělé inteligence jak kvalitou svého rozhodování tak schopností mít v něm prvek nepředvídatelnosti a zároveň se vypořádávat se skrytými informacemi ze strany soupeře [6].

MCTS je best-first search metoda která svým během zpracovává vyhledávací strom podobně jako Mini-Max nebo Alfa-Beta, k ohodnocování uzlů ovšem nepoužívá heuristickou funkci, ale provádí na nich Monte Carlo simulaci. Při použití pro tahové hry je kořenem stromu výchozí stav hry, respektive stav ze kterého hráč dělá rozhodnutí, jeho větvení odpovídá jednotlivým možným tahům z tohoto stavu, a uzly potomků jsou stavy hry po provedení daných tahů. Protože celý strom může být velmi rozsáhlý, nesnaží se ho algoritmus hned od začátku vygenerovat až do listů. Místo toho začíná jen s kořenem a každým cyklem do stromu přidává další uzel který ještě nebyl prozkoumán. Každý cyklus tedy přináší nové informace a zlepšuje kvalitu prohledávání, zároveň díky tomu může být algoritmus ukončován po určitém počtu cyklů nebo v libovolném cyklu uplynutím stanoveného času.

Ve volbě prozkoumávaných tahů figuruje prvek náhody, a proto lze na MCTS pohlížet také jako na metodu stochastické optimalizace které hledá řešení maximalizací šance výběru nejlepšího tahu nebo maximalizací hodnoty vybraného tahu. Přesněji řečeno se toho snaží dosáhnout v kořeni vyhledávacího stromu. V uzlech potomků je cílem co nejpřesněji ohodnotit skutečnou kvalitu tahu který do nich vede a při výběru mezi dvěma následujícími tahy které měly srovnatelný počet průchodů by měl být ten kvalitnější vždy lépe ohodnocený. Správné ohodnocení uzlu ale není triviální. Průměrování ohodnocení všech prozkoumaných potomků bude při velkém faktoru větvení podceňovat nejlepšího potomka. Převzetí ohodnocení z nejlepšího potomka naopak může přinést značné zkreslení když je prozkoumána jen malá část z potomků [7].

Vzhledem k tomu, jak MCTS prozkoumává, je zásadní správná volba uzlů k expanzi. Jednou z lákavých možností je stanovení prahu ořezávání s tím, že uzly, jejichž ohodnocení po určitém základním množství průchodů tohoto prahu nedosáhne, už nebudou dál prohledávány. To může přinést velké zrychlení, ale zároveň nese riziko, že bude odříznut podstrom který by vedl na dobré výsledky. Náhodné průchody totiž nemají zaručenou rovnoměrnou distribuci a může tedy dojít k předčasnému odříznutí tahu které je sám o sobě slabý, ale lze na něj navázat tahem velmi silným aniž by tento následující tah byl prozkoumán. Určitá preference lépe ohodnocených tahů je stále žádoucí, protože u řady průběhů lze předpokládat že si hráč převahu získanou silným tahem udrží a zaměření tímto směrem je tedy efektivnějším využitím prohledávacího času než průzkum zcela náhodný. Vhodnější je tedy přístup který přiděluje silnějším tahům vyšší pravděpodobnost dalšího prozkoumání ale ty slabší nevylučuje úplně. To staví MCTS do podobné problematiky jakou je N-armed bandit problem, ovšem s tím rozdílem, že N-armed bandit problem se snaží minimalizovat čas strávený průzkumem sub-optomálních možností. Naopak u MCTS průběžný průzkum nevadí pokud stále vede k tomu, že je nakonec vybrána nejlepší možnost [15].

Algoritmus Monte Carlo Tree Search

Algoritmus MCTS začíná vygenerováním kořene stomu a následně opakuje cyklus kterým do stromu přidává další uzly. V každém uzlu udržuje stav hry kterému uzel odpovídá, čítač průchodů uzlem a čítač souhrnného skóre těchto průchodů. Cyklus MCTS tvoří čtyři kroky.

1. Výběr (Selection) prochází strom od kořene směrem k listům dokud nenarazí na uzel u kterého ještě nebyly expandováni všichni potomci. V uzlech kde potomci expandováni jsou při sestupu upřednostňuje potomky s lepším ohodnocením. Ohodnocení potomků je spočítáno jako UCT (Upper Confidence Tree).

$$v_i = \bar{x_i} + C \times \sqrt{\frac{ln(n_p)}{n_i}}$$

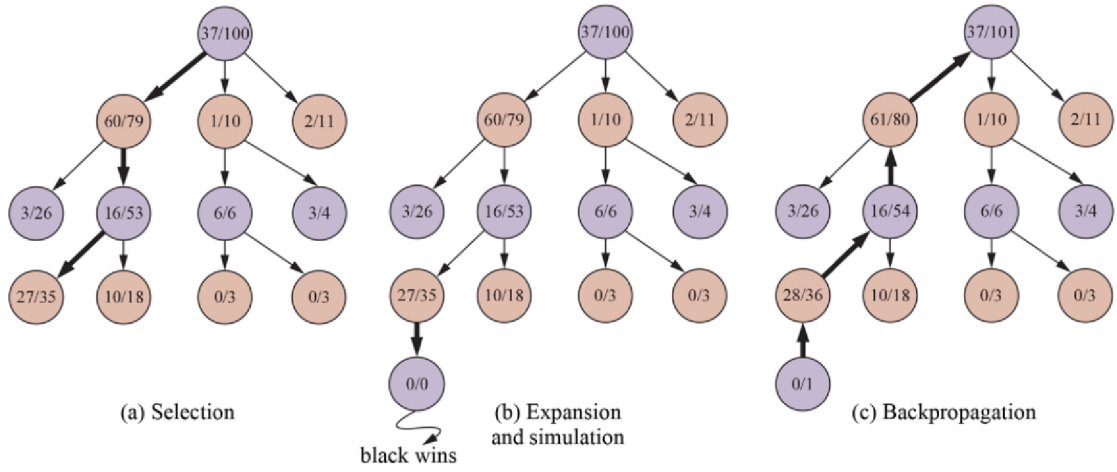
kde v_i je ohodnocení potomka i, $\bar{x_i}$ je jeho průměrné skóre, n_p je počet průchodů rodičovským uzlem a n_i je počet průchodů uzlem potomka. Konstanta C je parametrem vyvažujícím preferenci průzkumu (exploration) a využívání (exploitation) při průchodu stromu. To znamená, že určuje jak silně bude preferována expanze po cestě lépe hodnocených potomků oproti cestě méně prozkoumaným potomkům.

2. Expanze (Expansion) do stromu přidává dosud neprozkoumaného potomka aktivního uzlu. Pokud takových potomků existuje více, zvolí jednoho z nich náhodně. Potomek odpovídá provedení nějakého tahu a tedy posunu do nového stavu hry.

3. Simulace (Playout) spouští simulaci hry ze stavu který odpovídá nově přidanému uzlu až do konce hry. Tato simulace může volit tahy náhodně nebo využívat pomocné rozhodovací funkce využívající znalostí konkrétní hry. [4]

4. Zpětná propagace (Backpropagation) po dokončení simulace je ohodnocen stav hry do kterého došla. Toto ohodnocení je přiřazeno nově přidanému uzlu a také přičteno všem uzlů které se nacházel po cestě do nově přidaného uzlu.

Tyto čtyři kroky se opakují po stanovený počet cyklů nebo dokud nevyprší čas, a při dostatečném množství průchodů budou čítače uzlů v generovaném stromu konvergovat k tomu, aby nejlepší tahy vedly do potomků s nejvyšším počtem průchodů [20] [25].



Obrázek 3.3: Příklad jednoho cyklu MCTS [25]

Kapitola 4

Inteligence hry typu Scotland Yard

Mým záměrem je vytvořit několik verzí umělé inteligence jak pro prchajícího, tak pro pronásledující, aby bylo možné porovnat efektivitu různých přístupů vůči sobě navzájem. Zároveň chci v aplikaci mít flexibilitu pro práci s mapou a možnost vizualizace průběhu hry. Jádro aplikace bude schopné načíst mapu z externího souboru (konkrétní formát vyjádření herního plánu a dalších náležitostí ještě upřesním), na jejím základě inicializovat hru včetně rozlosování startovních pozic a udržovat přehled o průběhu hry, případně ho logovat a/nebo zobrazovat.

Zároveň bude žádoucí umožnit hru za každou ze stran živým hráčem. To sice nebude využito ve fázi učení protože pro naučení systému bude třeba odehrát řádově tisíce partií a bude tedy ponecháno na hrách stoje proti stroji. Lidský hráč se ale bude hodit pro testování úrovně již naučených inteligencí. Vizuální stránka zůstane spíše minimalistická, není totiž hlavním cílem práce.

Jádro hry bude komunikovat s moduly inteligence obou hráčů (jak bylo vysvětleno v sekci 2 pronásledující mají společný cíl a proto je zde považujeme za jednoho hráče), podávat jim informace o stavu hry, vyzývat je k rozhodnutí o tahu, verifikovat je-li požadovaný tah legální a případně ho provést. Tato modularizace zajistí lepší kontrolu toku informací a umožní organizaci sekvence více partií pro trénování.

Pro ani jednu ze stran neplánuji dělat modul který by volil tahy zcela náhodně, protože i jednoduchá informovaná metoda bude přinášet lepší výsledky než náhodné tahy [26] [18].

Modul reprezentující prchajícího má k dispozici stejné informace jako jádro hry.

Pro modul pronásledujících je z principu hry po většinu času skryta informace o pozici prchajícího. Může si ale na základě informací které k dispozici má některé další odvodit. Na začátku hry hráč pronásledujících ví, jaké jsou na hracím plánu startovní lokace a zároveň na které z nich byly umístěny jeho vlastní figury. Odvodí tedy, že prchající musí být na některé ze zbývajících startovních lokací. Po každém odehraném tahu prchajícího může na základě použitého lístku tuto množinu aktualizovat a místo lokací na kterých prchající mohl být vyčíslit lokace na které se mohl daným lístkem pohnout. Zároveň hráč pronásledujících z této množiny může vyřadit lokace na které vstoupil některou ze svých figur. Po objevení prchajícího v jednom ze stanovených tahů se množina začně konstruovat znovu od posledního pole kde byl viděn, opět podle použitých lístků. Tato množina pak jde využít heuristikou informované metody nebo pomocnou rozhodovací funkcí v MCTS simulaci [20].

Této dedukce by byl schopný i lidský hráč, byť by ji zřejmě neprováděl už před prvním objevením prchajícho protože v té fázi je to výpočetně náročné. Místo toho by pravděpodobně své první tahy hrál náhodně nebo se snažil své figury rozmístit tak, aby měly v

okamžik objevení prchajícího co největší počet otevřených cest a mohl ho začít obkličovat ať se objeví kdekoliv [14].

4.1 Alfa-Beta

Prohledávání celého stavového prostoru hry je v rozumném výpočetním čase neúnosné. Lze ale použít prohledávání do omezené hloubky, dva tahy dopředu stále únosné jsou. Faktor větvení tahu detektivů je maximálně 25 nebo 125 (z každého pole na mapě vede až 5 cest, a hýbou se 2 nebo 3 detektivové) pohyb prchajícího má faktor větvení 5 protože hýbe jen jednou figurou, detektivové ale musí uvažovat ne jeho skutečnou pozici která je jim skrytá ale celou množinu možných pozic.

Heuristické funkce

Cílem prchajícího je aby nebyl pronásledovateli dostižen. Svou pozici tedy bude hodnotit podle toho jak je od pronásledovatelů vzdálený, respektive jak vzdálený je od nejbližšího pronásledovatele, protože právě ten je pro něj bezprostřední hrozbou. Průměrná vzdálenost všech pronásledovatelů není směrodatná a může být dokonce zavádějící.

Druhou prioritou prchajícího je mít možnost úniku když se pronásledovatelé začnou přibližovat, takže jeho další rozhodovací kritérium je počet a množství druhů cest které z daného pole vedou. Jeho heuristická funkce může parametrizovat jakou váhu kterému z kritérií přikládá. Vyzkouším varianty, byť čekám že nejvhodnější bude prioritizovat vždy vzálenost a počet cest až jako pomocné kritérium v případě stejných vzdáleností.

Pronásledovatelé se naopak snaží minimalizovat svou vzdálenost od prchajícího, respektive od jeho možných pozic. Lze uvažovat dva přístupy jak to pojmout. Prvním je minimalizace součtu, nebo váhovaného součtu, vzdáleností od všech polí kde se prchající může nacházet. Druhým je minimalizace vzdálenosti od nejvzdálenějšího pole kde by prchající mohl být. Intuitivně by druhý přístup mohl být výhodnější protože jde přímo proti strategii o kterou se snaží prchající. Zároveň se nabízí to ověřit experimentálně.

Pro pohyb pronásledovatelů bude pravděpodobně vhodné zavést ještě další pomocné kritérim a to rozestup který mezi sebou mají. Jednak proto, že detektivové nesmí vstoupit na stejné pole a pokud stojí na sousedních polích redukují si možnost pohybu v následujícím tahu. Za druhé proto, že pracují jako tým a snaží se společně pokrýt prostor tak, aby prchajícího obklíčili, tedy detektivové by měli být mírně penalizováni pokud jsou k sobě blíže než na 2 pole, tato penalizace ale nesmí překročit odměnu za zkrácení vzálenosti k prchajícímu. Opět se nabízí k experimentálnímu ověření.

Vzhledem k tomu, že pronásledovatelé znají cíl prchajícího, můžou předpokládat že se bude snažit zvyšovat svoji vzdálenost od nich. Nabízí se tedy otázka jestli by pro prchajího nemohlo být za určitých okolností výhodné hrát jinak a zvolit tah kterým se k pronásledovatelům přibližuje protože předpokládá, že oni se budou rozhodovat jako by se vzdaloval. Možnost "zmatení protivníka" ale zřejmě skutečně nevyvažuje riziko hraní sub-optimálního tahu [3] [20].

Při uvažování vzáleností na herním plánu je potřeba počítat s tím, že existuje více druhů dopravy, takže naivně může být vzdálenost počítána jako nejkratší z nich. Při uvažování pohybu prchajícího to většinou bude odpovídat skutečnosti. Při pohybu pronásledovatelů je ale třeba zvážit i to, že mají k dispozici jen omezené množství lístků. Může se tedy stát, především ke konci hry, že některý z druhů dopravy už nebudou schopni využít. Když tedy prchající uvažuje jak daleko jsou od něj pronásledovatelé může zohlednit i toto omezení.

4.2 Neuronová síť

Dalším modulem bude implementace hráčských inteligencí hlubokou neuronovou sítí. Existuje dobrý předpoklad, že tento přístup je pro problematiku hry na principu Scotland Yard vhodný [8]. Otázkou zůstává jaké vlastnosti konkrétního řešení zvolit.

Velikost vstupní vrstvy sítě musí ze stavu hry dostávat veškerá relevantní data pro hráčské rozhodnutí. V první řadě pozice jednotlivých figur, ty musí být reprezentovány jako 1 z N (hot-one encoded) protože indexování polí je libovolně permutovatelné.

Z pohledu prchajícího tedy potřebuje znát svoji pozici a pozice jednotlivých pronásledujících. Pohyby pronásledujících jsou omezeny tím jaké mají k dispozici lístky takže pokud mají být trasovány pohybové možnosti nejsou pronásledující zaměnitelní a bude potřeba pozici každého reprezentovat zvlášť. Druhou možností je kompaktnější M z N, kde M je počet pronásledujících a k počtu lístků se buďto nepřihlíží vůbec nebo jen souhrnně. Chtěl bych tedy vyzkoušet jestli toto zjednodušení bude mít citelný vliv na degradaci kvality rozhodování nebo vede ke kompaktnějšímu srovnatelně kvalitnímu řešení. Rizikem je, že bude prchající zbytečně opatrný a bude uvažovat ohrožení i po cestách které pronásledující nemohou reálně vykonat.

Z pohledu pronásledujících je reprezentace vlastní pozice obdobná, reprezentace pozice prchajícího je ale o něco složitější. Může být vyjádřena buďto jako množina možných pozic jejíž odvození bylo vysvětleno na začátku kapitoly 4 nebo jako kombinace poslední známé pozice a počtu tahů které od ní uplynuly. Reprezentace konkrétní sekvence lístků které byly od poslední známé pozice použity zřejmě není vhodná minimálně proto, že délka sekvence se bude měnit každé kolo a počet vystupních neuronů je fixní.

Ve variantách kde bude každý pronásledující identifikován samostatně bude potřeba zahrnout do vstupu sítě i počet lístků jednotlivách typů které každému z nich zbývají.

Velikost skrytých vrstev sítě bude vzhledem ke komplexnosti rozhodování potřebovat alespoň tři vrstvy. Pro velikost skrytých vrstev se intuitivně nabízí zvolit je stejné jako velikost vstupní vrstvy nebo jako polovinu vstupní vrstvy. Další možností je počet neuronů v každé další vrstvě snižovat. Zde bude potřeba jednotlivé varianty ověřit experimentálně.

Velikost výstupní vrstvy sítě lze dimenzovat jednoduše. Od sítě se očekává rozhodnutí kam táhnout, takže výstupem bude preference pozice na kterou táhnout. Velikost bude pro prchajícího odpovídat počtu možných pozic. Pro detektivy pak počtu pozic krát počet detektivů, protože detektivové se rozhodují společně. Z výstupních tahů pak bude odehrán ten s nejlepším ohodnocením který je zároveň proveditelný.

Zpětná vazba pro trénování sítě bude vyhodnocena až na konci odehrání celé partie. Základním kritériem bude samozřejmě odměna výhry (+1) a penalizace prohry (-1). Dále lze využít toho, že prchající zapisuje svoje tahy do cestovního deníku a ten dává na konci partie k nahlédnutí. To mohou pronásledující porovnat se svými tahy a v každém tahu zjistit kterými rozhodnutími se k prchajícímu přibližovali, ty by mělo učení také odměnit (+0.1). Prchající naopak odměňuje tahy kterými se vzdaluje (+0.1). Dále by učení mělo penalizovat navrhování nemožných tahů, otázka je jak silně (uvažuji rozsah -0.01 až -0.1).

Očekávám, že pro natrénování každé varianty sítě budou potřeba řádově desetitisíce odehraných her, takže časová náročnost jejich vzájemného porovnávání nebude zrovna zanedbatelná. Přesto bych rád několik variant vyzkoušel. Porovnat složitějších a jednodušších varianty sítě, v reprezentaci pozic i velikosti a počtu mezivrstev. Síť která z porovnání s ostatními sítěmi vyjde nejlépe pak bude porovnávána s ostatními typy modulů inteligence.

4.3 Monte Carlo Tree Search

Moduly hráčské inteligence využívající MCTS budou vycházet ze základní verze algoritmu jak byla popsána výše 3.3. V jednotlivých uzlech stromu bude kromě čítačů nezbytných pro výpočet UCT reprezentován také aktuální stav hry. Na straně prchajícího to bude možné splnit bez obtíží. Na straně pronásledujících už to nelze udělat přímo a bude využito poslední známé lokace, respektive množiny možných míst odvozených z použitých lístků, odvozené v sekci 4.

Prvním z parametrů řešení bude nastavení konstanty C které určuje poměr prozkoumávání a využívání. Zde jsou pro danou aplikaci ověřené hodnoty 0.2 a 2.0 [20]. Chtěl bych vyzkoušet dvojnásobky na rozsah 0.2 až 3.2

Dalším rozhodnutím bude jak realizovat výběr tahů v kroku simulace. Zde se nabízí hned několik možností. Jednak použití prosté maximalizace (respektive minimalizace) zdáleností, případně ϵ-greedy výběr s použitím maximalizující (minimalizující) heuristiky.

Na straně pronásledujících je pak otázka jak uvažovat skrytou pozici prchajícího. Zde je hned několik možností:

- náhodný výběr ze všech možných lokací prchajícího
- snižování sumy vzdálenosti ke všem možným lokacím prchajícího
- výběr nejvzdálenější možné lokace prchajícího
- pravděpodobnostní výběr lokace prchajícího váhovaný vzdáleností (ruleta)

U kterých bych chtěl experimentálně ověřit jejich efektivitu, s tím že za nejslibnější považuji poslední dvě.

V každém případě se budu snažit udržet výpočetní složitost v takovém rozsahu aby bylo pro každý běh algoritmu MCTS provedeno alespoň 5000 cyklů a zároveň tempo hry nebylo pomalejší něž 5 vteřin na tah.

Poslední uvažovanou možností je použít pro rozhodování v rámci MCTS simulace již naučenou neuronovou síť. To může být buďto společně jak na straně prchajícího tak u pronásledujících, nebo jednostranně oproti jednodušší rozhodovací logice. Cílem je vyzkoušet jestli lze s MCTS podporovaným neuronovou sítí dosáhnout lepších výsledků než při rozhodování přímo neurovonou sítí samotnou, nebo MCTS s jinou logikou v simulaci.

Kapitola 5

Implementace

Tato kapitola popisuje realizaci hry navržené v sekci 2.3. Jako implementační jazyk byl zvolen Python (v3.10.4). V nadcházejících sekcích budou rozebrány podrobnosti jednotlivých komponent aplikace, vysvětlena vnitřní logika jejich fungování, představeny datové stuktury používané v aplikaci a popsáno obecné použití aplikace s parametrizací spuštění i grafické uživatelské rozhraní pro testování hry lidským hráčem.

Na základě výše uvedného návrhu implementuje tato aplikace verzi hry s (až) dvěma druhy dopravy, zde pracovně nazývanými `Taxi` a `Tram` a libovolným propojením stanic.

Konkrétní definice herního plánu, a také parametrizace běhu hry, jsou popsány dále 5.1. Zmíněny jsou i klíčové datové struktury, což by mělo usnadnit pochopení rozhodování které na jejich základě aplikace provádí.

Součástí implementace jsou různé varianty umělé inteligence za obě strany hry. Tedy zaprvé fantoma, pracujícího s úplnou informací o stavu hry. Zadruhé pak detektivy, pro které je po většinu hry fantomova pozice skryta, a jejichž rozhodování musí spoléhat na dedukci, respektive vhodnou heuristiku.

Nejjednodušším rozhodovacím mechanismem je náhodná volba tahů 5.2.1, zavedená pouze pro základní ověření funkčnosti a referenci, s předpokladem, že v porovnání s ostatními variantami bude ve výrazné většině případů prohrávat.

Sofistikovanější metodou je heuristické rozhodování 5.2.2, které se snaží na základě dostupných informací každým tahem zlepšovat svoji pozici. V případě fantoma to znamená zvyšovat nebo alespoň nezmenšovat svoji vzdálenost od detektivů, v případě detektivů pak nejprve dedukovat možné pozice fantoma a následně se k nim snažit přiblížit.

Následující přístupy rozšiřují rozhodovací logiku o využití strojového učení, jednak je to Monte Carlo Tree Search (MCTS) 5.2.3, generující z bodu rozhodování strom možných průběhů hry s volitelnou preferencí využívání strategií, které se zdají být silné na první pohled, vůči zdánlivě slabším, které mohou vést na lepší výsledek ve vzdálenějším horizontu. Relativě vyšší rozhodovací čas by měl být vyvážen kvalitou získaného rozhodnutí.

Další variantou je pak použití neuronové sítě 5.2.4, která po patřičném natrénování převádí vektor stavu hry na rozhodnutí o nejvhodnějším tahu. Zásadní je zde za prvé struktura, respektive dimenze sítě, a za druhé zajištění vhodných trénovacích dat, ať už odvozováním z odehraných partií nebo vhodným automatizovaným generováním. Tento dataset by měl mít objem v řádech desetitisíců až statisíců, takže manuální definice je nemyslitelná.

Výše uvedené metody strojového rozhodování jsou pak doplněny možností ovládat jednu ze stran hry přímo uživatelem 5.3. Tomu je poskytnuta grafická vizualizace stavu hry se všemi relevantními informacemi a volba mezi možnostmi tahů je ponechána na něm.

Hry, kde se obě strany rozhodují strojově, graficky vizualizovány nejsou, v případě potřeby ale mohou být jejich výsledky vypisovány na příkazovou řádku, případně logovány do souborů, a to jak souhrnně, tak na úrovni jednotlivých partií.

5.1 Reprezentace dat

Tato sekce popisuje vstupní data nezbytná pro funkci aplikace, její výstupy a některé klíčové vnitřní struktury.

Konfigurace spuštění aplikace

Aplikace je spouštěna souborem `main.py`, její nedílnou součástí je ale také konfigurační soubor `config.ini`, který definuje, co a jak přesně bude konkrétní běh zpracovávat. Konfigurační soubor obsahuje 3 sekce.

`[GAME]` je první z nich, obsahuje povinně parametr `boardfile`, určující soubor, ze kterého bude načtena definice herního plánu. Ten může být následovaný volitelným `gameset`, určujícím číselně počet her, které mají být po spuštění odehrány (pokud není uvedeno proběhne jen jedna hra). Parametr také může být místo čísla nastaven na hodnotu `variants`, v tom případě proběhne tolik her, aby se vyzkoušely všechny varianty startovního rozestavení figur. V případě herních plánů použitých v sekci testování 6 to bude variace 4 ze 6, tedy 360. Dále jsou zde parametry `cmd` a `log`, určující intenzitu výstupů na příkazovou řádku a do logů, oba ve třech stupních: `no` - žádný výstup, `set` - souhrnný výstup za celý set her, a `game` - výstup za každou jednotlivou hru. Případné logy jsou pak ukládány ve složce `logs`.

Následuje sekce `[PHANTOM]`, jejím hlavním parametem je `phantom`, určující typ fantomovy rozhodovací logiky, varianty `random heuristic mcts mctshe nn player`, odpovídají variantám rozhodovací logiky, popsaným v předchozí sekci 5, s tím, že `mcts` používá při odehrávání her sestavujících svůj strom náhodné rozhodování, zatímco `mctshe` používá heuristické. Obě varianty MCTS jsou ovlivněny parametry `thinktime` a `constant`, určujícími dobu jeho běhu ve vteřinách a konstantu C, vyvažující preferenci průzkumu (exploration) a využívání (exploitation). Ostatní logiky tyto parametry ignorují, dimenzování neuronové sítě je počítáno automaticky, podle principů popsaných dále 5.2.4.

Sekce `[DETECTIVES]` pak používá hlavní parametr `detectives`, případně svoje hodnoty `thinktime` a `constant`, jejichž očekávané hodnoty a význam jsou obdobné jako u fantoma.

Vstupní soubor herní desky

Herní deska je popsána souborem ve formátu `.json`. Ten obsahuje pole (array) `plan` se záznamy pro jednotlivá herní pole (zastávky) hrací plochy. Každá ze zastávek pak obsahuje povinně atributy `stop`, udávající číslo zastávky (indexováno od 1, jako u fyzické verze hry Scotland Yard), atribut `init` udávající, jestli je zastávka startovní, tedy jestli na ni může být umístěna figura při startovním rozestavení, a dvojici polí `taxi` a `tram`, obsahující zastávky, na které se dá dojet daným druhem dopravy. Pokud jeden z druhů dopravy z dané zastávky neodjíždí, příslušné pole bude v záznamu zastávky prázdné.

U záznamu zastávky mohou být také volitelné atributy `posX` a `posY`, udávající, kde má být zastávka vykreslena při vizualizaci hry. Jednotky souřadnic jsou abstraktní a před vykreslením budou automaticky přepočítány podle rozměru okna.

Například tedy, pátá zastávka může být startovní, vedou z ní `taxi` spoje na zastávky 4 a 12, žádný `tram` spoj, a je specifikováno, kam ji vykreslit.

```
{"stop":5,"init":true,"posX":1,"posY":8,"taxi":[{"stop":4},{"stop":12}],"tram":[]},
```

Příklad je zvolen pro svou stručnost, u většiny zastávek je počet spojů vyšší. Reciprocitně se očekává, že jeden z `taxi` spojů ze zastávek 4 a 12 povede vždy na zastávku 5. Tento formát zápisu je sice redundantní, protože všechny spoje musí být obousměrné, explicitní uvádějí obou směrů se ale ukázalo jako vhodný nástroj automatizované kontroly konzistence a ochrana proti překlepům při přepisu z fyzického plánu hry.

Kromě pole zastávek pak soubor plánu hry obsahuje ještě objekt `phantom` s položkami `taxi` a `tram` určujícími startovní množství příslušných lístků, objekt `detectives`, s obdobnými specifikacemi lístků a navíc položkou `count` určující počet detektivů ve hře.

Poslední součástí specifikace je objekt `game` s položkou `turns` určující celkové trvání hry a pole `reveals`, určující tahy, ve kterých se fantom ukazuje.

Výpisy a logy

Výstupem aplikace může být log průběhu hry, případně setu her se stejným nastavením. Kromě hlavičky parametrů, s jakými byla hra spuštěna, je průběh jedné hry reprezentován polem o délce odpovídající počtu tahů hry plus 3 a šířce dvojnásobku počtu hráčů, s tím, že první dva sloupce odpovídají tahům fantoma, a každé další dva tahům jednoho z detektivů. První řádek logu reprezentuje zbývající tikety v pořadí `taxi`, `tram`, další řádky pak pozice v daném tahu a způsob dopravy `1 = taxi`, `2 = tram`, a `0 = pass`. Na příkladu níže 5.2, tedy fantom začíná na zastávce 19 a z ní se prvním tahem přesunul `taxi` na zastávku 28. Log je vždy z pohledu fantoma, tady všechny tahy viditelné. Soubor logu je pojmenován podle času vytvoření a pořadovým číslem hry v sekvenci, log setu pak jen časem. Log setu obsahuje hlavičku parametrů a souhrnnou statistiku úspěchu jednotlivých stran, jak lze vidět na příkladu 5.1.

Výpis průběhu hry na příkazové řádce hlavičku neobsahuje, a ukazuje jen tabulku průběhu, obdobně výpis setu udává jen souhrnný výsledek.

```
Board: plan1det3.json
Phantom: heuristic
Detectives: heuristic

Total games: 10
Phantom wins: 5 (50.00%)
Detectives win: 5 (50.00%)
```

Obrázek 5.1: Příklad souhrnného logu setu her

```
Board: plan1det3.json
Phantom: heuristic
Detectives: heuristic
[[17  6  0  4  0  0  0  0]
 [19  1  5  1 15  1  1  1]
 [28  1 12  1 16  1  2  2]
 [21  1 11  1 18  1 16  1]
 [22  1 12  1 11  1 17  2]
 [21  1 13  1 12  1 33  2]
 [28  1 22  1 11  2 20  1]
 [27  1 21  1 17  2 19  1]
 [26  2 28  1 33  1 17  1]
 [30  2 27  0 32  1 25  1]
 [14  2 27  0 26  2 24  1]
 [ 6  2 27  0 30  2 15  1]
 [ 1  1 27  0 14  1 16  2]
 [ 8  1 27  0 15  0  2  0]
 [16  0  0  0  0  0  0  0]]
Phantom wins
```

Obrázek 5.2: Příklad logu jedné konkrétní hry

Vnitřní reprezentace dat

Zpracování vstupního souboru herní desky a evidenci průběhů hry řeší třída `GameBoard`, definovaná v `gameBoard.py`, která po inicializaci zpracuje informace potřebné perzistentně napříč všemi hrami v setu a připraví vzor matice průběhu hry se startovními distribucemi lístků, bez rozlosování pozic. K tomuto vzoru ještě vytvoří odpovídající masku zakrývající informace, které nevidí detektivové.

Třída reprezentace hry si dále udržuje dvě verze dopravní sítě. Tou první je prosté převedení zápisu ze vstupního souboru tak, aby šlo podle čísla zastávky, resp. indexu pole, vyhledat přímé spoje, které z ní vedou. Před potvrzením této reprezentace je také otestováno, že všechny uvedené spoje jsou symetrické.

Druhou reprezentací možných přesunů mezi zastávkami je pak plně propojená dopravní síť, jejíž struktura je ilustrována na obrázku 5.3. Je to čtvercová matice o dimenzích odpovídajících počtu zastávek, která pro libovolnou kombinaci výchozí a cílové zastávky obsahuje seznam nejlepších možných cest. Cesty jsou oceněny podle spotřeby lístů (a tedy i tahů) na jejich vykonání a za nejlepší možné cesty jsou považovány ty, pro které neexistuje alternativa, která by byla v jednom druhu lístků levnější a v dalších alespoň stejně drahá. Rozlišení druhů lístků je nutné, nelze pohlížet pouze na celkový součet spotřebovaných lístků, protože především v pozdější části hry může dojít k situacím, kdy hráči dochází jeden druh lístků, takže cesta s nejmenší sumou už je pro něj nedosažitelná, ale cesta s vyšší sumou v jiné kombinaci lístků stále ano.

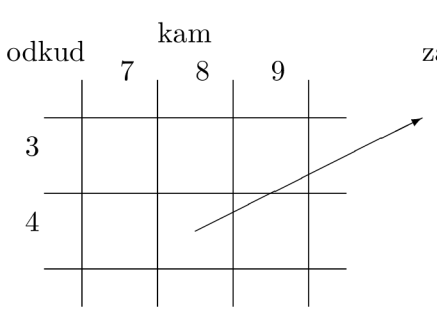
Sestavení této matice probíhá přepsáním přímých spojů do maticového formátu a následnou iterací po celém jejím rozsahu, prodlužováním existujících cest vždy o jeden krok, s eliminací těch, pro které existuje striktně lepší alternativa, dokud nenastane iterace při které už nebylo nic přidáno. Toto sestavení je tedy výpočetně poměrně náročné $O(n^3)$ probíhá ale pouze jednou při načtení spojů a výsledná matice pak může být znovupoužívána napříč celým setem her.

Tato plně propojená dopravní síť je pak využívána pro rozhodování jednotlivých hráčských algoritmů. Vzhledem k tomu, že její matice je sestavena deterministicky na základě veřejně známých informací dostupných všem hráčům, může být její vytvoření součástí herní desky a hráči si ji odtud v případě potřeby převezmou.

Samotné záznamy cest mezi dvěma zastávkami pak neobsahují celou cestu, ale pouze její první krok a celkovou cenu v jednotlivých druzích lístků. Vzhledem k tomu, že matice byla sestavena iterativně, se lze po každém kroku znovu dotázat a alternativy cest do cíle budou kratší právě o předchozí vykonaný krok. Na druhou stranu, u provedeného kroku je nutné explicitně rozlišovat druh použitého dopravního prostředku, a to vzhledem k tomu, že mezi některými zastávkami existují oba druhy spojů. Lístek použitý v prvním kroku by mohl zkreslit celkovou cenu. Tato možnost se pro většinu herních plánů bude týkat pouze jednotek procent cest, alternativou by ale bylo ověřovat případy, kdy jsou na stejný krok použitelné oba druhy spojů, explicitně při každém tahu, takže je vhodnější to mít uvedeno rovnou jako součást záznamu. Pro další zefektivnění je při sestavování matice a řadě dalších operace využíváno knihovny NumPy [21].

Nová hra pak může být zahájena kopií předpřipravené matice průběhu hry a obsazením startovních pozic jednotlivých figur, to může být provedeno náhodným rozlosováním nebo na základě číselného seedu (který umožňuje predikovatelné rozestavení). Zároveň jsou vynulovány čítače tahů a aktivního hráče.

Probíhající hra může být dotazována na stav herního plánu (s rozlišením pohledu fantoma a detektivů), existující spoje, případně další fakta. Pro provedení tahu je poskytnuta funkce která zároveň ověřuje, jestli je požadovaný tah platný.



	cesta		cena	
zastávka	spoj	taxi	tram	
2	2	2	1	
2	2	1	2	
3	1	4	0	

Obrázek 5.3: Vizualizace plně propojené dopravní sítě

5.2 Implementace hráčů

Jádro hry tvoří `main.py`, který po zpracování konfigurace a delegaci načtení herní desky vytváří instanci požadovaného druhu fantoma a detektivů. Následně volá vytvoření instance hry a v cyklu předává hráčům informace o stavu hry a jejich rozhodnutí na základě těchto informací aplikuje na hru, odehrává tahy a zaznamenává výsledky.

Odpovídající druhy rozhodovací logiky fantoma a detektivů jsou vždy principileně velmi podobné, s jediným zásadní rozdílem, jak se detektivové vypořádávají se skrytou informací o pozici fantoma. Obě varianty tedy budou popisovány společně se zmíněním příslušných rozdílů. Detektivové odehrávají v pevně dané posloupnosti, pro rozhodování používají všichni stejný modul, výpočet rozhodnutí je ale prováděn pro každého zvlášť.

Varianta, která by se snažila zkoordinovat rozhodnutí všech detektivů naráz a po nalezení shody je odehrát jako celou sekvenci, by byla podstatně náročnější a preference je zde spíše na vyzkoušení více možných variant hráčské logiky než zdokonalování jedné konkrétní.

5.2.1 Náhodné tahy

Nejjednodušší rozhodovací logika, implementovaná v rámci souborů `phantomRandom.py` a `detectiveRandom.py`, je náhodná volba tahů. Jak fantom, tak i detektivové, odehrávají tak, že nejprve porovnají svou pozici s tabulkou přímých spojů a vyčíslí z ní platné tahy, které mohou vykonat ze stávající zastávky. Následně náhodně vyberou jednu z těchto možností. Detektivové si navíc musí ověřit, že se nepohybují na pozici jiného detektiva, takový tah by byl neplatný, o pozici fantoma se ale nezajímají.

5.2.2 Heuristické rozhodování

Heuristické rozhodování, realizované `phantomHeuristic.py` a `detectiveHeuristic.py`, před volbou tahu analyzuje svoje možnosti a snaží se nalézt nejvhodnější rozhodnutí, kterým zlepší svoji šanci na výhru. K tomu využívá především plně propojenou dopravní síť, popsanou výše 5.1. Ta umožňuje ověřit jaké vzdálenosti, repektive možné cesty, od sebe dělí libovolná dvě pole na herním plánu. Tyto cesty mohou mít několik variant v různých kombinacích lístků. Vzhledem k tomu, že informace o zbývajících lístcích každého z hráčů je veřejně dostupná, lze rovnou ověřit, které cesty pro hráče jsou nebo nejsou vykonatelné.

Z pohledu fantoma to znamená, že po tom, co z tabulky přímých spojů vyčíslí zastávky, na které se může ve svém tahu pohnout, může každou svoji možnost ohodnotit. Porovnáním pozicí detektivů s maticí plně propojené dopravní sítě a zbývajícími lístky jednotlivých detektivů zjistí, v kolika tazích je pro každého z nich zastávka dosažitelá, respektive jestli vůbec.

Ohodnocení pro jednotlivé detektivy pak může fantom použít na výpočet celkového ohodnocení daného pole, intuitivně se nabízí převzetí vzdálenosti nejbližšího detektiva, suma vzdáleností všech detektivů nebo váhovaný součet vzáleností detektivů. Aby šlo váhovaný součet v aplikaci použít stejnou logikou, jako ostatní varianty, musí pro něj platit, že vyšší ohodnocení je lepší (fantom se snaží detektivům co nejvíce vzdálit), a zároveň, že čím je konkrétní detektiv blíže, tím větší má vliv na výsledné ohodnocení. Ve zvolené implementaci tedy hodnocení začíná na konstantě násobené počtem detektivů. Od ní je za každého detektiva odečtěna druhá mocnina vzdálenosti, o kterou je blíže než 6 polí. Tedy detektiv ve vzdálenosti 6 a více nemá vliv, detektiv ve vzdálenosti 5 odečítá 1 (1^2), detektiv ve vzdálenosti 4 odečítá 4 (2^2) a detektiv ve vzdálenosti 0, tedy stojící přímo na poli, kam by se fantom pohnul, odečítá 36.

Z nabízených možností pak fantom vybere tu s nejvyšším ohodnocením, snaží se detektivům co nejvíce vzdálit. V případě shody hodnocení používá pomocné kritérium, kterým je počet odchozích cest z dané zastávky (s přihlédnutím k dostupným lístkům), zajišťuje si tak co nejpestřejší výběr možností v následujícím tahu.

Z pohledu detektivů je rozhodování složitější, snaží se fantomovi co nejvíce přiblížit, o jeho pozici ale mají po většinu hry jen omezené informace. Prvním krokem jejich heuristiky je tedy, na základě dostupných informací, zúžení možností kde se fantom může skutečně nacházet. Tato množina je sestavována iterativně, buďto z množiny možných startovních pozic, redukované o startovní pozice detektivů, nebo z poslední známé pozice, kde se fantom ukázal. Podle použitého lístku jsou z tabulky přímých spojů vybrány všechny zastávky, na které se lze ze stávající množiny zastávek dostat. Následně jsou vyřazeny ty, které by

kolidovaly s pozicí některého z detektivů, a výsledná množina se stává výchozí množinou pro další iteraci.

Z množiny možných pozic fantoma pak detektivní heuristika rozhodne kterou se bude snažit pronásledovat. Je samozřejmě nutné zohlednit pozice ostatních detektivů, aby provedený tah nekolidoval, a vzdálenosti jsou přebírány z plně propojené dopravní sítě, obdobně jako u rozhodování fantoma. S přihlédnutím k tomu má heuristika několik možností preference. Jednak zaměření na nejbližší z možných fantomových pozic, kde je sice nepravděpodobné, že se tam fantom bude skutečně nacházet, pozice je ale nejlépe dosažitelná a jeji průchod zredukuje další možnosti fantomova skrývání. Další možností je směřování k nejvzdálenější možné pozici fantoma, kde by se na základě fantomovy logiky mohl skutečně nacházet. U varianty sledování nejvzdálenější pozice ale zároveň hrozí, že v pozdější fázi hry nebude dosažitelná kvůli omezenému množství zbývajících lístků. To může řešit podmíněná preference nejvzdálenější pozice, pokud je dosažitelná, následovaná preferencí nejbližší pozice.

Pokud není detektiv v některé z výše uvedených variant schopen celé cesty ani k nejbližší fantomově pozici, zahájí alespoň pohyb tímto směrem, pokud na něj má první potřebný lístek.

Náhodný výběr ze všech možností, případně náhodný výběr s váhováním podle vzdálenosti, zde využity nejsou. U tohoto heuristického přístupu je snaha o maximální konzistenci, posílenou tím, že i v případě shodně oceněných možností není rozhodnutí náhodné, ale je provedeno jako modulo počtu možností z hashe pozic jednotlivých figur ve hře.

Důvodem je to, že prvek náhodného rozhodování je nedílnou součástí všech ostatních metod, a striktní determinističnost heuristiky by měla poskytnout lepší referenci pro porovnání heuristického a stochastického přístupu.

Jak u fantoma, tak u detektivů, platí, že pokud mohou udělat nějaký tah, tak i musí, byť by tím svoji situaci zhoršili, možnost vynechání tahu `pass` nastává pouze vynucením. Posledním krokem heuristiky je tedy volba proveditelného tahu v případě, že nebyl nalezen žádný tah s preferencí. Tento princip se pak opakuje i u dalších rozhodovacích mechanismů.

5.2.3 Monte Carlo Tree Search

Implementace metody Monte Carlo Tree Search (MCTS), prováděná v `phantomMcts.py` a `detectvieMcts.py`, využívá principů metody popsaných dříve v rozboru 3.3, oproti původnímu návrhu 4.3, ale doznala některých odlišností.

Zpracovávaný strom MCTS je reprezentován jako seznam a neukládá v uzlech přímo samotné stavy hry. Vzhledem k tomu, že průběh hry je reprezentován celou třídou, by to bylo nepraktické. Místo toho položky stromu tvoří záznamy sestávající z indexu rodičovského uzlu, čítače ohodnocení, čítače počtu průchodů a tabulky možných tahů z daného stavu. Každý z tahů, který už byl expandován, obsahuje index příslušného listu, který slouží jako odkaz na něj. Neexpandovaný záznam obsahuje pouze tah jako takový - tedy aktivního hráče, druh dopravy a cílovou zastávku. Kořen stromu (nultá položka seznamu) odpovídá stavu ze kterého je MCTS spouštěn, což slouží jako bod zastavení backpropagation a zároveň zajišťuje, že nulový index nikdy nebude potomkem.

Průchod stromem probíhá kopírováním výchozího stavu hry a následným odehráváním tahů v sekvenci procházených listů. Výběr uzlů při sestupu preferuje nepozkoumané uzly a následně se řídí vzorcem Upper Confidence Tree popsaným v teoretickém rozboru 3.3 s tím, že konstanta C je volitelná jako jeden z parametrů konfiguračního souboru 5.1.

Při expanzi uzlu jsou v tabulce přímých spojů vyhledány možné tahy z něj a podle těch vytvořeny další záznamy do seznamu, jejich referencí na rodiče je expandovaný uzel, respektive jeho index.

Pro fázi odehrání (playout) je možnost výběru mezi náhodným a heuristickým rozhodováním hráčů, to je rovněž specifikováno v rámci konfiguračního souboru. MCTS tedy integruje oba výše popsané přístupy. Volba je automaticky pro oba hráče stejná, aby odehrávání bylo pokud možno vyvážené a nezreslovalo výsledek MCTS jako takového.

Když odehrání dosáhne výsledku je provedena zpětná propagace, ta využívá výše zmíněné indexy rodičovských uzlů, po kterých se vrací až ke kořeni a aktualizuje čítače procházených uzlů, ohodnocení je přímočaré - vítězství 1 bod, prohra 0, remízy z principu neexistují.

Rozvoj stromu je ukončen časovým limitem, který je definován v rámci konfigurace a ověřován vůči systémovému času. Podle principu MCTS je pak vybrán potomek kořene s nejvyšším počtem průchodů. Konkrétní nastavení parametru C a časového limitu bude nutné ověřit v rámci experimentů 6, a je vhodné zohlednit, že strom je sestavován znovu pro každý z hráčových tahů, v případě detektivů pak pro každou z figur.

Tahy vedoucí přímo do koncového stavu už nejsou expandovány a když by mělo nastat jejich odehrání, tak neproběhne, a rovnou vrací výsledek. Další zefektivnění, zejména v koncové fázi hry, je kontrola větvení výchozího stavu před zahájením MCTS a v případě jednozačné volby, např. vynucený `pass`, algoritmus vůbec nespustí.

Z pohledu detektivů je třeba opět řešit neúplnou informaci o pohybu fantoma. Prvním krokem je zúžení možností podle principu heuristického přístupu 5.2.2, a to i v případě, že playout algoritmu jako takový bude náhodný. Z možností je pak vybrán jeden předpoklad, a to zcela náhodně nebo s preferencí vzdálenosti. Ten považujeme za pozici fantoma pro konkrétní běh MCTS. Pro realizaci toho předpokladu je v rámci třídy stavu hry, respektive její kopii, možnost vnucení nové pozice fantoma. Konstrukce stromu, a následující playouty, pak vychází z ní.

Nový odhad pro každý průchod není pro tuto realizaci vhodný, protože by, především v úvodní části hry, před prvním odhalením fantoma, strom neúměrně větvil a zkreslil jeho vypovídací hodnotu.

5.2.4 Neuronová síť

Realizace hráčské inteligence neuronovou sítí je řešena v rámci skriptů `phantomNn.py` a `detectiveNn.py`. Implementace využívá knihovnu TensorFlow [11] a jako základní kostra pro její použití vychází z cvičení v rámci kurzu Biologií inspirované počítače [13].

Nasazení neuronové sítě na problematiku má podobu plně propojené dopředné sítě bez využití konvolučních vrstev s aktivační funkcí ReLU ve skrytých vrstvách a aktivační funkci softmax ve výstupní vrstvě. Konkrétní počet a velikost skrytých vrstev sítě bude předmětem experimentů ve fázi testování 6.4. Velikost výstupní vrstvy je přímo daná rozměrem herního plánu, respektive počtem zastávek na něm - očekávaným výstupem sítě je určení preference možných tahů.

Vstupní vrstva pak musí odpovídat formátu vstupních dat. Pro zajištění těchto vstupních dat se původní myšlenka zpětné analýzy odehraných her a dílčího hodnocení jednotlivých kroků, jak byla zamýšlená v rámci návrhu 4.2, ukázala jako nepraktická. Místo toho jsou trénovací data generována samotným modulem před spuštěním trénování sítě.

Dataset pro učení neuronové sítě je vytvořen vyčíslením všech možných pozice figur na herním plnánu a jejich napárování s nejvhodnějšími tahy pro minimalizaci, případně

maximalizaci, vzdálenosti fantoma od detektivů. V této fázi vycházíme z předpokladu, že všechny pozice jsou viditelné a všichni hráči mají dostatek lístků pro libovolnou cestu.

Zohledňování zbývajících lístků přímo v datasetu explicitním vyčíslením by ho ještě zásadně zvětšovalo a to násobkem vyšším než součin startovních počtů lístů jednotlivých typů za každého z hráčů, v rámci provedených testů 6 by to znamenalo $(8{\cdot}4)^4$, respektive 2^{20}, a je tedy vyjmuto z trénování a ověřováno až v kroku rozhodnutí.

Existuje pak několik možností, jak pozice figur na herním plánu reprezentovat. Jednak na ně lze pohlížet striktně jako na variaci - tedy pozice fantoma i každého z detektivů je unikátní a bude reprezetována jako hot one encoded vektor o velikosti počtu zastávek a celý záznam pak bude mít délku počet zastávek krát počet figur ve hře.

Další možností je úvaha, že detektivové jsou vzájemně zaměnitelní, pozice fantoma zůstává hot one encoded, ale pozice všech detektivů jsou zapsány do společného vektoru nastavením příslušných bitů jejich pozic. Záznam pak má délku dvojnásobku počtu zastávek, kde polovina reprezentuje pozici fantoma a polovina souhrnně pozice všech detektivů.

Poslední možností, kterou zde uvažujeme, je reprezentace z pozice detektiva, který rozlišuje hot one encoded pozici fantoma, hot one encoded pozici svoji, a společně kódovanou pozici všech ostatních detektivů. Záznam pak má délku trojnásobku počtu zastávek.

V každém případě je z vygenerovaného datasetu odebráno náhodným výběrem 20% záznamů, které poslouží jako testovací data pro učení sítě, zbývajících 80% pak zůstává jako data trénovací.

Výstupem natrénované sítě je pak namapování herní pozice ve stejném formátu, v jakém byla trénovací data na preference následujícího tahu. Tyto preference jsou pak porovnány s možnostmi následujícího tahu na základě dostupných lístků a z nich je zvolen nejlépe ohodnocený proveditelný tah.

Pro rozhodování detektivů pak, na rozdíl od předchozích metod, nedochází k odhadu konkrétní pozice fantoma, na místo toho je vektor rozhodnutí sestaven z celé množiny možných pozic fantoma.

5.3 Uživatelské rozhraní

Grafické uživatelské rozhraní je implementováno přímo v rámci `main.py` s využitím kolekce grafických knihoven PyGame [22]. Pokud je v konfiguraci aplikace nastaveno, že za jednu ze stran bude hrát uživatel, inicializuje se před spuštěním hry samotné ještě vykreslovací okno aplikace. Při tahu uživatele jsou pak do okna zobrazeny informace o aktuálním stavu hry a hráči nabídnuty možnosti jeho tahu.

Ve středu vykreslovacího okna jsou zobrazeny zastávky s očíslováním a jednotlivé spoje mezi nimi v barevném kódování `taxi` - žlutá, `tram` - modrá. Pozice zastávek mohou být specifikovány prostřednictvím `posX` a `posY` v rámci souboru definujícího herní plán. Jestliže tento volitelný parametr nemají, je jejich vykreslení automaticky uspořádáno do nejmenšího možného čtverce. Pokud nebylo propojení linek zamýšleno pro čtvercovou mřížku, nezaručuje automatické vykreslení přehlednost a je lepší pozice specifikovat explicitně, rozdíl je demonstrován na příkladu vykreslení 5.4 reprezentujícím herní plán z návrhu 2.2. Pro čtvercovou mřížku naopak usnadňuje definici bez nutnosti specifikovat pozice explicitně, jak ukazuje následující příklad 5.5.

Další zobrazovanou informací o stavu hry jsou zbývající lístky fantoma a jednotlivých detektivů, ty jsou zobrazeny v horní části okna s kódováním odpovídajícím barvám spojů a indikací aktivního hráče.

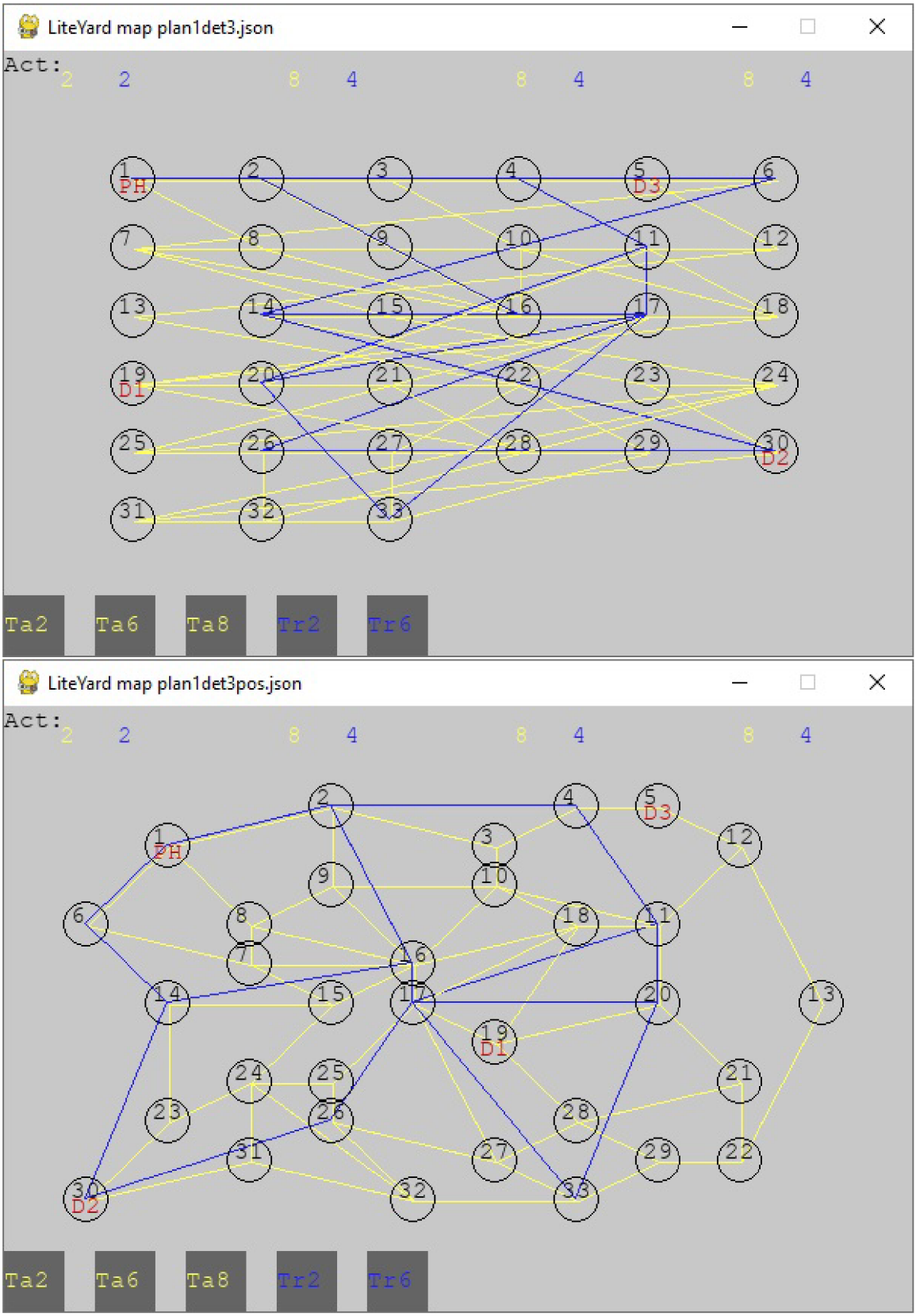
Pozice samotných hráčů na herním plánu jsou vypsány na příslušných pozicích, pokud jsou známy, a zdůrazněny červeně. Identifikace figur používá označení `PH` a `D1` až `DN`.

Při hře detektivů jsou v levé části uživatelského rozhraní vypsány lístky použité fantomem, jeho skryté pozice jsou značeny otazníkem. V tazích, kdy se fantom objevuje, je jeho pozice vyznačena na herním plánu i vypsána do levého seznamu.

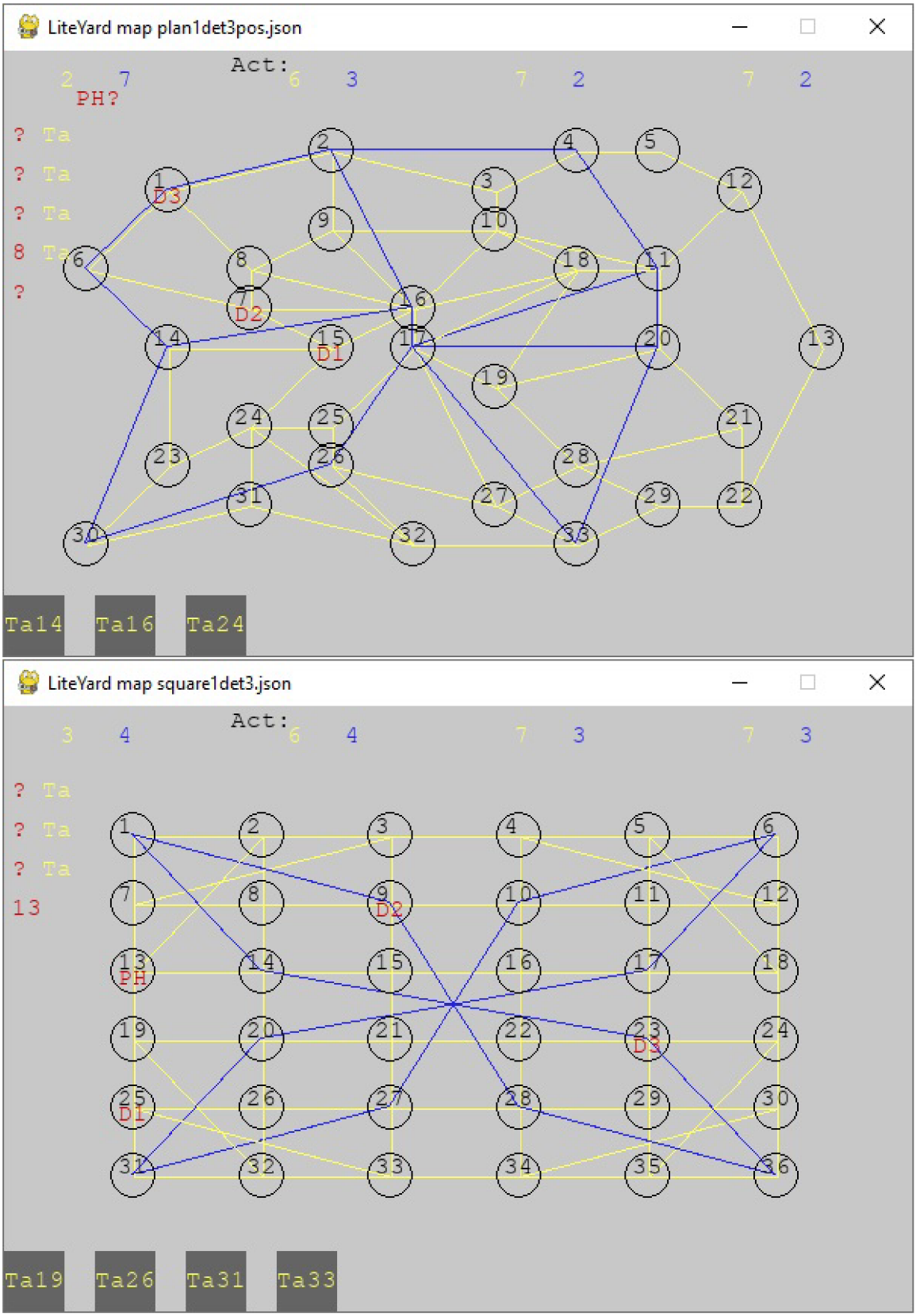
Možné tahy hráče jsou identifikovány příslušnými skripty `detectiveListMoves.py` a `phantomListMoves.py` a zobrazeny ve formě tlačítek na spodní straně uživatelského rozhraní. Zadávání tahů přímo kliknutím na následující zastávku by bylo nepraktické, protože neumožní rozlišení způsobu dopravy v případech, kdy jsou pro daný cíl možné oba.

Při zavření okna v průběhu probíhající partie je partie ukončena předáním neplatného tahu (`pass` na pozici 0). Při jakémkoliv konci hry z vícenásobného setu je hned zahájena hra další.

Příklady zobrazení hry z pohledu fantoma a detektivů jsou ukázány na následujících stránkách 5.4, 5.5.



Obrázek 5.4: Nová hra z pohledu fantoma - automatické a specifikované vykreslení



Obrázek 5.5: Probíhající hra z pohledu detektivů - plán a čtvercová mřížka

Kapitola 6

Testování

Tato kapitola se věnuje testování funkcionality aplikace a jednotlivých modulů hráčské inteligence v různých konfiguracích a mezi sebou navzájem. Pokud není uvedeno jinak každý test je realizován jako set 100 běhů v dané konfiguraci, pokud jsou uváděny výsledky pouze jedné strany je výsledkem druhé strany doplněk do 100.
Jako referenční stroje byly použity:

- Lenovo X1 Carbon gen4 (8GB RAM, Dual-Core 2.4GHz, SSD M.2) - pro testy MCTS
- Lenovo X1 Carbon gen3 (8GB RAM, Dual-Core 2.3GHz, SSD M.2) - pro ostatní testy

Oba s operačním systémem Win10 64bit. Důvodem této separace je vyšší časová náročnost běhů Monte Carlo Tree Search.

Jako herní plány pro účely těchto testů slouží `plan1det3.json` představující výřez mapy fyzické hry Scotland Yard o 33 polích jak byl představen v návrhu 2.2 s 6 startovními pozicemi (1,5,15,19,22,30) a `square1det3.json`, což je čtvercový středově symetrický plán o 36 polích, rovněž se 6 startovními pozicemi (2,6,15,23,26,34). Vizualizace obou je vidět na příkladu uživatelského rozhraní 5.5. Startovní pozice byly voleny tak, aby mezi každými dvěma byly pokud možno pravidelné rozestupy alespoň tří tahů a měly srovatelné množství výstupních cest, což ovšem u herního plánu převzatého z fyzické mapy nešlo dodržet striktně.

Délka hry je v obou případech 12 kol, s odhalením fantoma v 3, 6 a 9 kole. Startovní lístky detektivů jsou v poměru 2:1 rámcově reflektujícím hustotu propojení jednotlivých druhů dopravy a v součtu odpovídají počtu tahů hry, jako je tomu u originální verze Scotland Yardu [23]. Konkrétně tedy detektivové dostanou 8 `Taxi` lístků a 4 `Tram` lístky každý. Fantom začíná s dvěma lístky od každého druhu a podle principu hry dostává lístky použité detektivy.

Pro vizualizaci některých výsledků je využito knihovny Matplotlib [16].

6.1 Náhodné tahy

Testy s náhodnými tahy obou hráčů jednak ověřují integritu jádra hry, protože výskyt neplatného tahu nebo problém při zpracování tahu by hru ukončil bez výsledku, vedoucí dále na nesprávný součet setu. Za druhé test s náhodnými tahy pomáhá stanovit vhodý počet detektivů pro další testy. Cílem rozhodně není kalibrace 50:50, při náhodných tazích fantom ztrácí, respektive nevyužívá, svou výhodu a jeho úspěšnost by mohla být kolem 25%.

Jak ukazují výsledky v tabulce 6.1 součty her souhlasí a požadované úspěšnosti fantoma odpovídají pro obě testovací mapy právě tři detektivové.

Herní plán	plan1				square1			
Počet detektivů	1	2	3	4	1	2	3	4
Výhry fantoma	68	36	21	6	66	41	24	12
Výhry detektivů	32	64	79	94	34	59	76	88

Tabulka 6.1: Test úspěšnosti stran podle počtu figur detektivů

Dalším testem je pak porovnání náhodného rozhodování s ostatními hráčskými inteligencemi, kde by mělo platit, jak bylo zmíněno v rozboru implementace 5, že náhodný hráč se všemi ostatními variantami rozhodování v převážné většině případů prohrává. Výsledky v tabulkách 6.2 a 6.3 tento předpoklad podporují.

Herní plán	plan1det3	square1det3
Heuristický	2	0
MCTS	7	14
MCTS(He)	3	23
Neuronová síť	5	2

Tabulka 6.2: Úspěšnost náhodně hrajícího fantoma proti jiným hráčským inteligencím

Herní plán	plan1det3	square1det3
Heuristický	97	96
MCTS	65	86
MCTS(He)	77	95
Neuronová síť	98	96

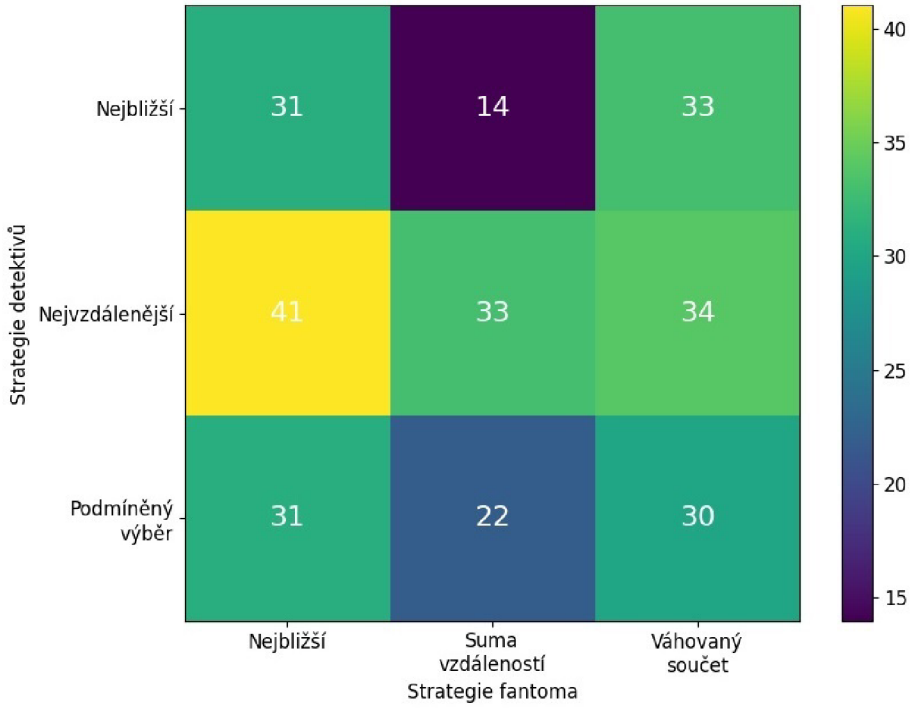
Tabulka 6.3: Úspěšnost různých verzí fantoma proti náhodně hrajícím detektivům.

6.2 Heuristické rozhodování

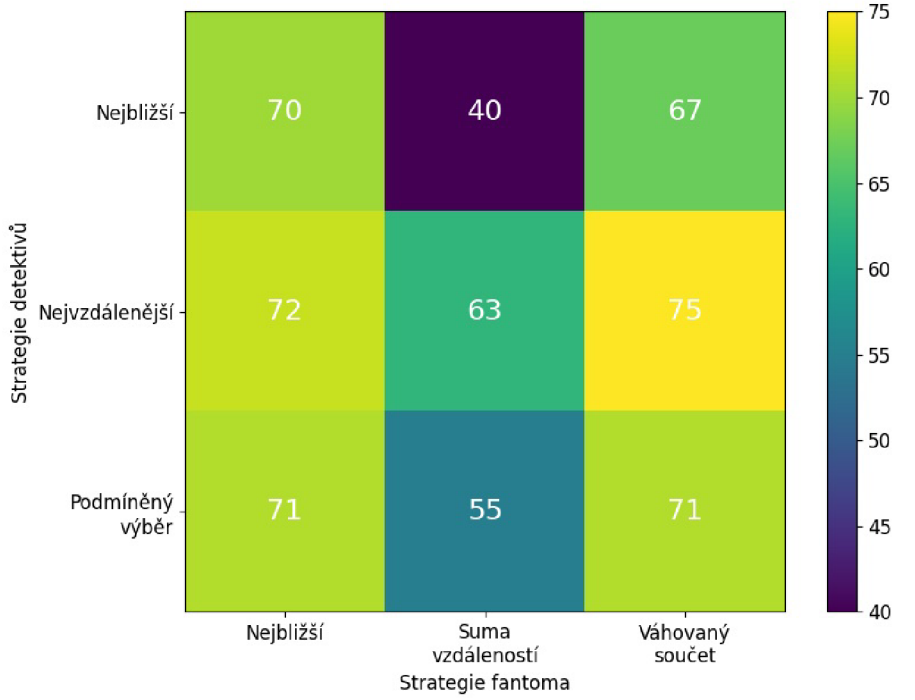
Testy heuristického rozhodování proti sobě porovnávají různé strategie přepočtu vzdálenosti na ohodnocení polí v heuristikách fantoma a detektivů. Jak bylo vysvětleno v popisu heuristik 5.2.2, pro každou stranu je na výběr ze tří variant výpočtu.

Viditelně se zde projevuje rozdíl mezi úspěšností fantoma na `plan1det3` 6.1 kde jsou úspěšnější detektivové, zřejmě proto, že v nestejnoměrné síti cest je snažší fantoma obklíčit a symetrickým `square1det3` 6.2 kde se fantomovi daří unikat lépe.

V obou případech ale platí, že fantomovi strategie Nejbližší a Váhovaný součet jsou úspěštěnší než Suma vzdáleností a naopak pro detektivy jsou strategie Nejbližší a Podmíněný výběr úspěšnější než strategie Nejvzdálenější. Bohužel ani v jendom případě nelze jednoznačeně říct, že by jedna ze strategií dominovala napříč oběma herními plány.



Obrázek 6.1: Úspěšnost fantoma podle zvolené strategie na plan1det3



Obrázek 6.2: Úspěšnost fantoma podle zvolené strategie na square1det3

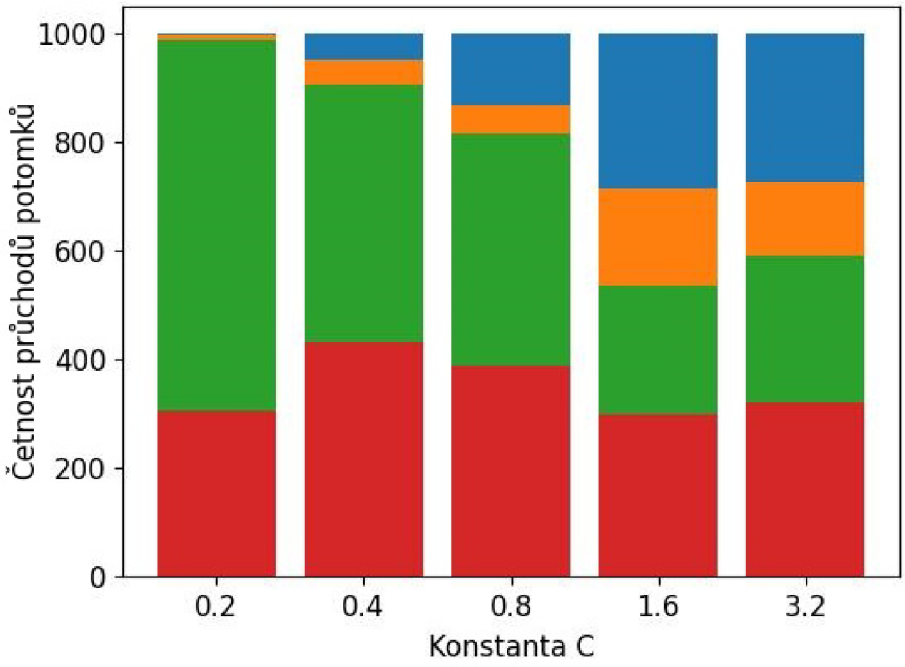
6.3 Monte Carlo Tree Search

U metody MCTS je v testech nejprve ověřován vliv konstanty C na vyvažování průzkumu (exploration) a využívání (exploitation). Pro účely tohoto testu byl algorismus upraven a místo omezení počtu cyklů dobou běhu byl stanoven čítačem na 1000 průchodů. Zároveň byl zafixován seed startovního rozestavení figur tak, aby kořen MCTS stromu měl právě 4 potomky. Na `plan1det3.json` je to seed 121 pro startovní pozici fanoma 15 a pozice detektivů 1, 23, 34. Na `plan1det3.json` pak seed 125 pro pozici fantoma 15 pozice detektivů 1, 5, 22. Použití obou herních plánů a obou stran hry by mělo zaručit objektivnější výsledek.

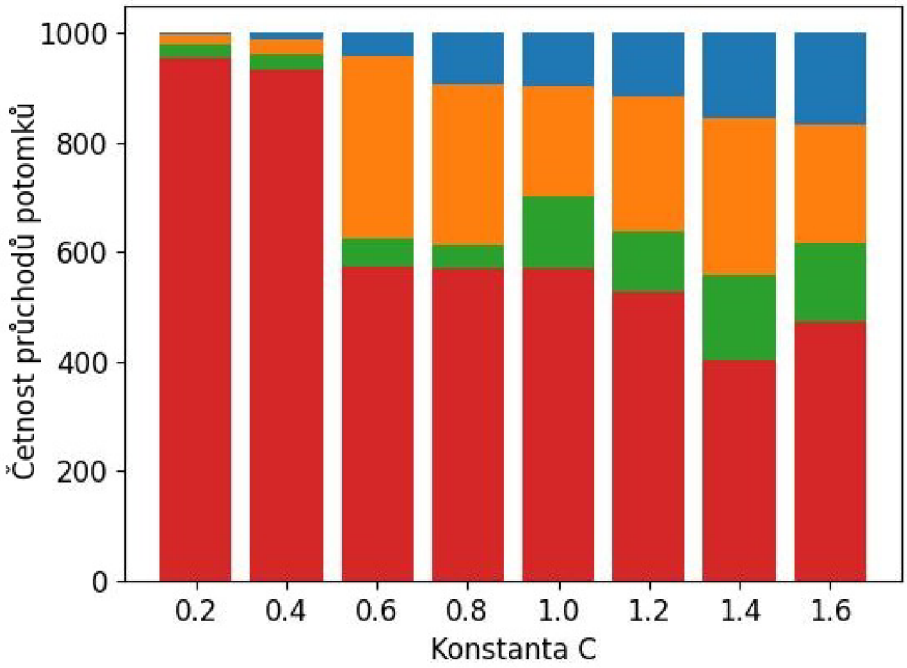
Testy byly provedeny nejprve v rozsahu 0.2 až 3.2 s krokem dvojnásobku jak bylo navrženo v teoretickém rozboru 4.3. Na základě jejich výsledku, vizualizovaném na 6.3 a ukazujícím, že ke změnám větvení dochází na rozsahu 0.4 až 1.6, pak byla druhá sada testů provedena na rozsahu 0.2 až 1.6 s jemnějším krokem 0.2.

Zde z výsledků, vizualizovaných na 6.4 vidíme, že pro C nižší než 0.6 se algoritmus silně zaměřuje na exploitaci prvního objeveného průchodu vedoucího na vítězný playout a jeho výstup zřejmě bude silně ovlivněn průběhem několika málo počátečních průchodů. V rozsahu 0.6 až 1.0 už začíná být větvení vyrovnanější a jako nevhodnější vypadá rozsah 1.0 až 1.6. To koresponduje s literaturou, která zmiňuje, že vhodné nastavení C může být $\sqrt{2}$ nebo 1.5 [25].

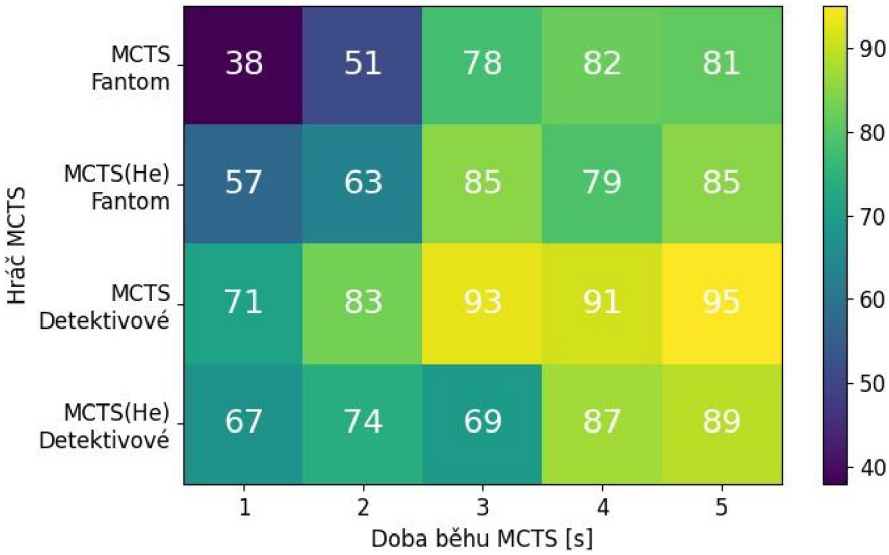
Při vhodné volbě C je úspěšnost MCTS ovlivňována dobou běhu která musí umožňovat dostatečný průzkum stromu aby se algoritmus dopracoval ke kvalitnímu rozhodnutí. Testy provedené na `plan1det3.json` naznačují, že pro zvolený rozsah herního plánu je to 4 až 5 vteřin, jak je vidět na zobrazení výsledků 6.5.



Obrázek 6.3: Vliv konstanty C na větvení z kořene stromu MCST pro hru na plan1det3, seed startovního rozestavení 121, tah z pohledu fantoma



Obrázek 6.4: Vliv konstanty C na větvení z kořene stromu MCST pro hru na square1det3, seed startovního rozestavení 125, tah z pohledu prvního detektiva

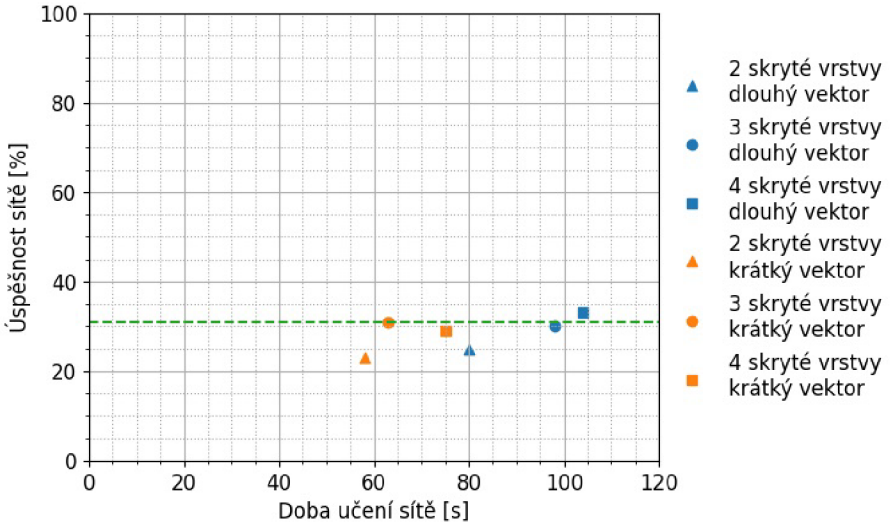


Obrázek 6.5: Vliv doby běhu algoritmu MCTS na úspěčnost MCTS hráče, testováno na plan1det3 proti náhodnému soupeři

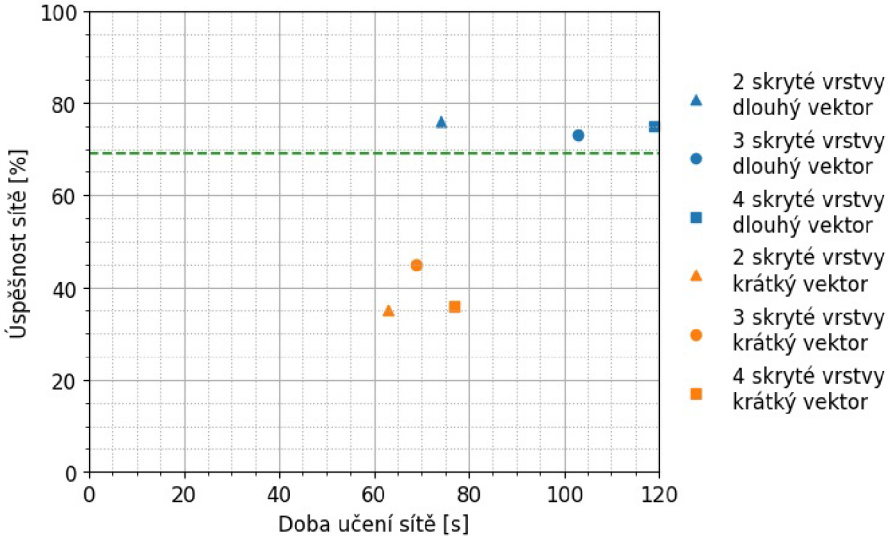
6.4 Neuronová síť

Pro implementace jednotlivých hráčů neuronovou sítí vychází řešení z datasetu, jehož velikost je přímo odvozena od velikost herní plochy, respektive počtu zastávek na ní. Zdůvodnění tohoto rozhodnutí bylo podrobněji vysvětleno v sekci 5.2.4.

Předmětem testování je porovnání různých zápisů datasetu do trénovacích vektorů, a to buďto s rozlišením jednotlivých detektivů, zde uvedeno jako `dlouhý vektor`, nebo se společným zápisem pozic všech detektivů, zde uvedeno jako `krátký vektor`. Druhým parametrem je počet skrytých vrstev sítě, vrstvy jsou plně propojené a o stejné velikosti jako vstupní vektory. Sledovanými parametry jsou doba trénování sítě a úspěšnost proti deterministickému heuristickému soupeři.



Obrázek 6.6: Výkonnost různých konfigurací neuronových sítí fantoma na plan1det3.json

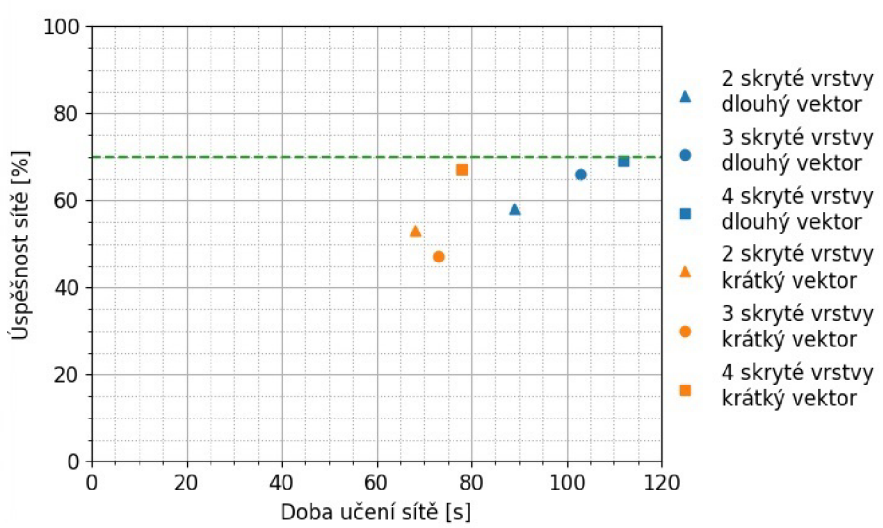


Obrázek 6.7: Výkonnost různých konfigurací neuronových sítí detektivů na plan1det3.json

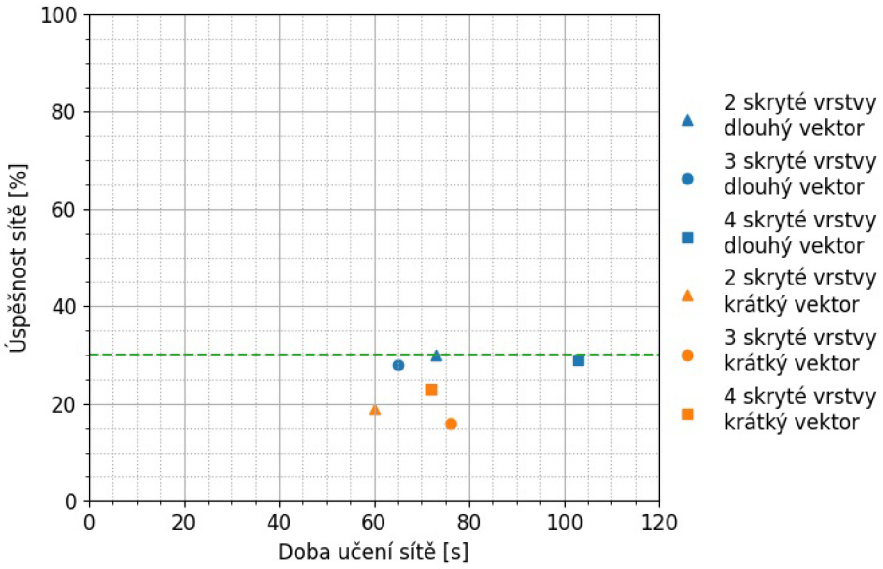
Úspěšnost je vztahována oproti heuristickému řešení, jehož výsledek je v grafech naznačen zelenou linkou. Z výsledků neuronových sítí detektivů 6.7 a 6.9 je vidět, že krátký vektor podává slabší výkon než referenční heuristika, dlouhý si při 3 a 4 skrytých vrstvách vede srovnatelně s referencí, nebo lépe.

U neuronové sítě fantoma je naopak vidět, že při použití čtyř vrstev sítě dosahuje úspěšnosti srovnatelné s referencí i pro krátký vektor, výsledky ilustrovány na grafech 6.6 a 6.8.

Tento výsledek intuitivně dává smysl, protože pro fantoma je zásadní vzdálenost mezi jeho polem a poli obsazenými detektivy, ale konkrétní rozmístění detektivů na obsazených polích je pro něj libovolně zaměnitelné. Rozhodující se detektiv naopak potřebuje jasně odlišit která figura je jeho a která ostatních detektivů.



Obrázek 6.8: Výkonnost různých konfigurací neuronových sítí fantoma na square1det3.json

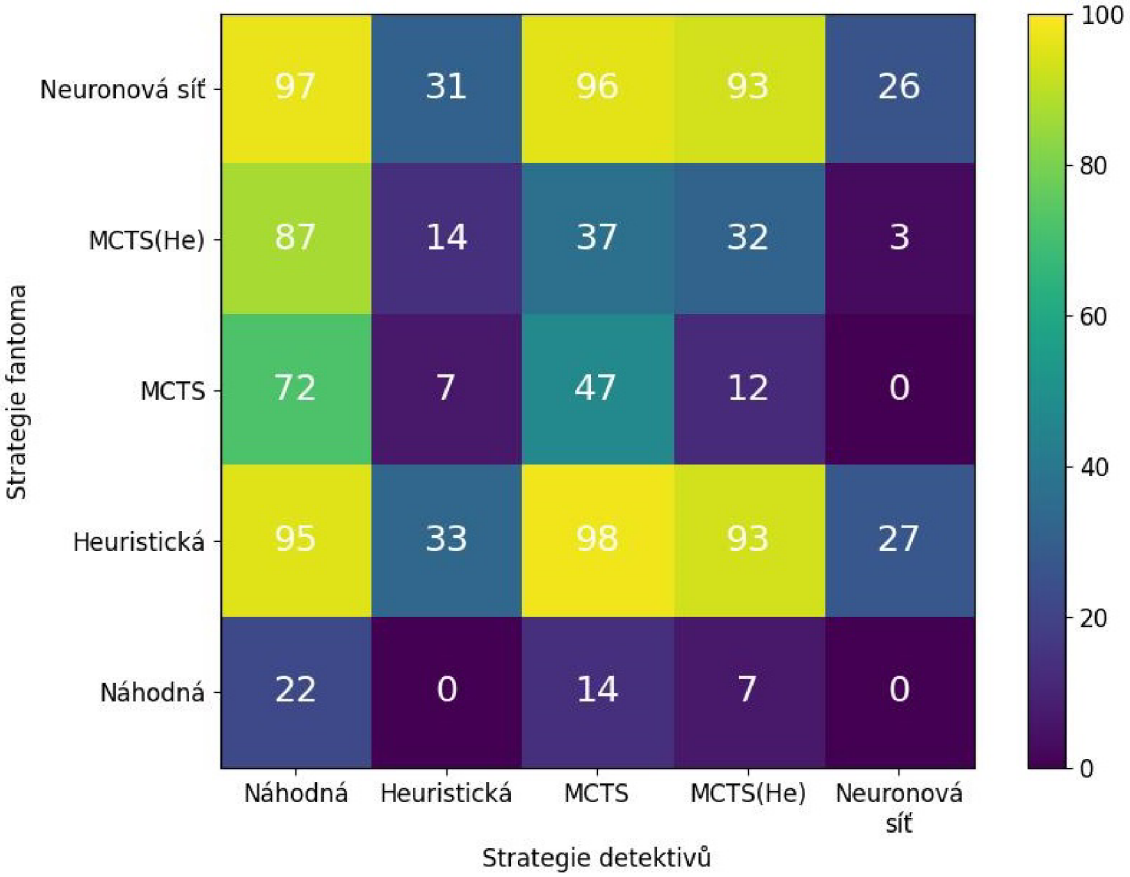


Obrázek 6.9: Výkonnost různých konfigurací neuronových sítí detektivů na plan1det3.json

6.5 Živý hráč a kombinované testy

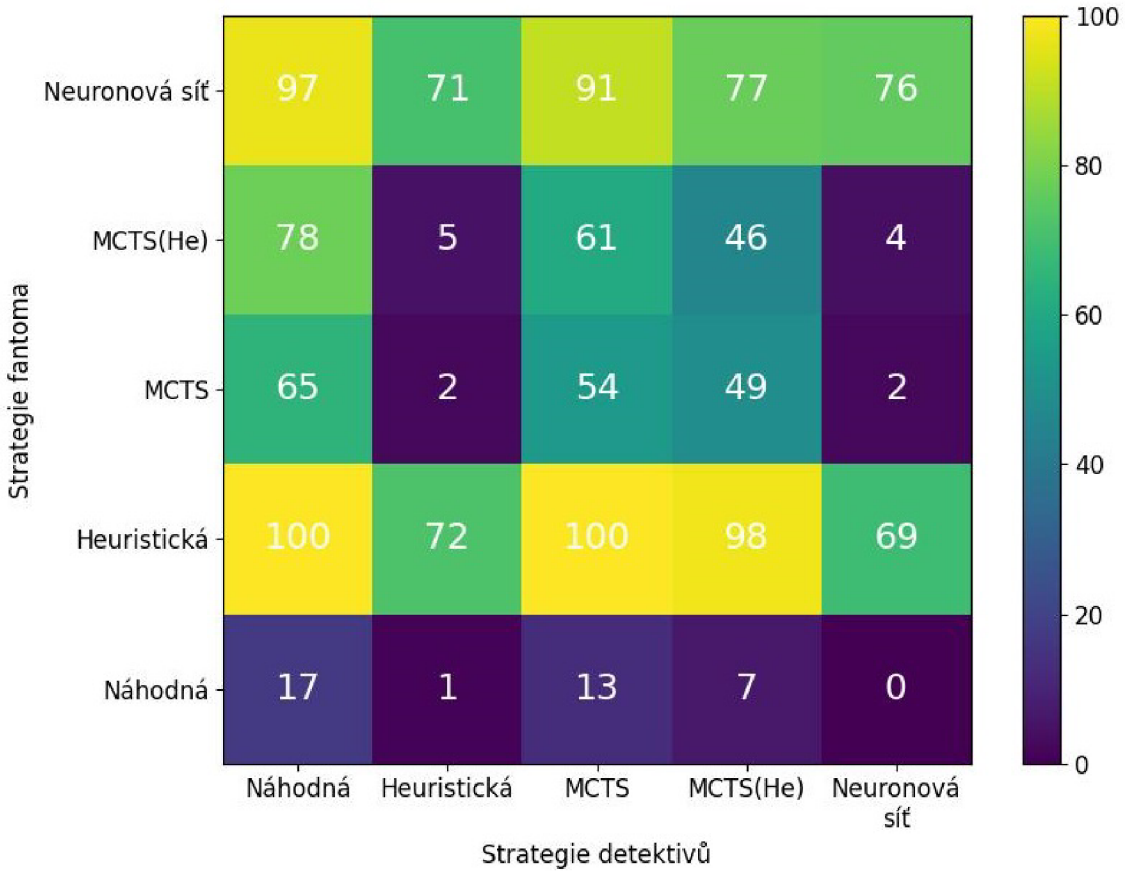
Poslední kategorií testů je přehledové srovnání všech implementovaných inteligencí za obě strany hry na obou herních plánech. Použity jsou vždy konfigurace které se dobře osvědčily v rámci testování každého typu zvlášť, tedy konkrétně:

- pro deterministickou heuristiku fantoma ocenění pole podle nejbližšího detektiva
- pro deterministickou heuristiku detektivů preference nejbližšho možného pole fantoma
- pro obě varianty MCTS konstanta $C = 1.4$ a doba běhu 5 vteřin na tah
- pro neuronovou síť na straně fantoma krátký vektor a 4 skryté vrstvy sítě
- pro neuronovou síť na straně detektivů dlouhý vektor a 4 skryté vrstvy sítě



Obrázek 6.10: Úspěšnost všech vyvinutých implementací fantoma proti všem impmenentacím detektivů na herním plánu plan1det3

Z výsledků testů na obou plánech, vizualizovaných 6.10 a 6.11, je vidět, že nejlépe si ve strovnání vedou heuristická implementace a neuronová síť. Nejslabší strategií je dle očekávání náhodná volba tahů.



Obrázek 6.11: Úspěšnost všech vyvinutých implementací fantoma proti všem impmenentacím detektivů na herním plánu square1det3

Závěrečným ověřením je hra živého hráče proti jednotlivým implementacím, z praktických důvodů omezeno na 10 partií na typ, hráč odehrává za fantoma na plan1det3 a za detektivy na square1det3. Hráč se snaží odehrávat do 5 vteřin.

Soupeř	Náhodný	Heuristický	MCTS	MCTS(He)	Neuronová síť
Výhry uživatele za fantoma	10	5	8	9	6
Výhry uživatele za detektivy	10	6	9	9	7

Tabulka 6.4: Test úspěšnosti uživatele proti jednotlivým implementacím

Z tabulky výsledků 6.4 je vidět, že živý hráč má stále převahu, částečně zřejmě díky možnosti holistického rozhodování které implementační logika nepostihuje, zároveň ale tato převaha není zcela jednoznačná a především u heuristické logiky a neuronové sítě se blíží vyrovnaným výsledkům.

Kapitola 7

Závěr

Cílem tohoto projektu bylo jednak navrhnout a implementovat hru na principu stolní hry Scotland Yard, ve které je po většinu hry pro jednu ze stran částečně skryt stav herní plochy, a to konkrétně pozice protihráče, druhak pak k této hře vytvořit několik druhů inteligence pro každou ze stran a porovnat jejich úspěšnost vůči sobě.

Ve fázi návrhu byly rozebrány principy sekvenčních asymetrických her se systémem prchající-pronásledující, vliv skryté informace v nich, a možnosti jak se s ním vypořádávat. Vlastní hra je pak navržena tak, aby tyto principy obsahovala. Součástí implementace je také logovací systém a grafické uživatelké rozhraní umožňující hru uživatele proti libovolné z inteligencí.

Pro danou hru bylo implementováno několik variant rozhodovací logiky pro obě strany hry. První z nich je deterministická heuristika využívající jednorázového předpočítání matice cest mezi libovolnými dvěma pozicemi na mapě se kterou je pak možné na základě pozorovaných nebo spekulovaných pozic figur ohodnotit jednotlivé možnosti tahů. Ta je kombinována s algoritmem pro zúžení možných pozic protihráče na základě dostupných neúplných informací.

Dalším implementovaným přístupem je využití metody Monte Carlo Tree Search (MCTS) která buduje rozhodovací strom z bodu, kde je činěno rozhodnutí, a spekulativním odehráváním možných průběhů hry s vyvažováním expanze nových cest a využívání již prozkoumaných hledá nejsilnější tah.

Posledním použitým řešením je neuronová síť, která na základě vygenerovaného datasetu natrénuje volbu výhodných tahů při známých stavech hry a jejich zobecněním pak hledá výhodné tahy při částečně skrytém stavu hry.

Pro každou z implementací bylo vyzkoušeno několik variant jejich nastavení na dvou různých herních plánech. Parametrizován byl výpočet ohodnocení herních polí, vyvažovací konstanta MCTS, délka běhu rozhodnutí, reprezentace dat pro neuronovou síť a strutura sítě jako takové. Varianty s nejslibnějšími výsledky z každého typu pak byly porovnány vůči sobě navzájem - deterministická heuristika a neuronová síť dosahovaly na obou stranách hry kvalitních výsledků. MCTS vykazuje zlepšování laděním vyvažovací konstanty a prodlužováním délky běhu, na navržené hře ale nedosáhl výsledků neuronové sítě.

Možností navázání na získané výsledky by mohlo být rozšíření přístupu použitém u neuronové sítě na hru Scotland Yard jako takovou, případně pro další hry na stejném principu. Pro MCTS by mohlo být vhodné ho doplnit podpůrnou heurisikou nebo kombinací s další metodou umělé inteligence.

Literatura

[1] Armstrong, D. *Probability of Poker Hands*. 2006. Dostupné z: https://www-users.cse.umn.edu/~reiner/Classes/Poker.pdf.

[2] Arts, A. *Competitive play in Stratego*. Maastricht, NL, 2010. Diplomová práce. Maastricht University, Department of Knowledge Engineering. Vedoucí práce Dr. Mark Winands and Maarten Schadd, MSc.. Dostupné z: https://project.dke.maastrichtuniversity.nl/games/files/msc/Arts_thesis.pdf.

[3] Borák, D. *Heuristiky pro hru Scotland Yard*. Praha, CZ, 2021. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická. Vedoucí práce RNDr. Vojtěch Kovařík, Ph.D.. Dostupné z: https://hdl.handle.net/10467/94486.

[4] Bouzy, B. Associating Domain-Dependent Knowledge and Monte Carlo Approaches within a Go Program. *Information Sciences*. 4. vyd. 2005, č. 175, s. 247–257. DOI: 10.1.1.6.1059. ISSN 0020-0255. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6.1059&rep=rep1&type=pdf.

[5] Broeck, G. Van den, Driessens, K. a Ramon, J. Monte-Carlo Tree Search in Poker using Expected Reward Distributions. *Advances in Machine Learning*. 1. vyd. 2009, č. 5828, s. 73–83. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-642-05224-8_28.

[6] Chung, M., Buro, M. a Schaeffer, J. *Monte Carlo Planning in RTS Games*. IEEE, 2005. Dostupné z: http://webdocs.cs.ualberta.ca/~jonathan/publications/ai_publications/mcsearch.pdf.

[7] Coulom, R. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Computers and Games*. 1. vyd. 2006, č. 4630, s. 72–83. DOI: 10.1007/978-3-540-75538-8_7. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-540-75538-8_7.

[8] Dash, T., Dambekodi, S., Reddy, P. a Abraham, A. Adversarial neural networks for playing hide-and-search board game Scotland Yard. *Neural Computing and Applications*. 1. vyd. 2020, č. 32, s. 3149 – 3164. DOI: 10.1007/s00521-018-3701-0. ISSN 0941-0643. Dostupné z: https://link.springer.com/article/10.1007%2Fs00521-018-3701-0.

[9] Fantasy Flight Games. *Manuál ke hře Fury of Dracula*. 3. vyd. 2015. Dostupné z: https://www.fgbradleys.com/rules/rules4/Fury%20of%20Dracula%20-%20rules.pdf.

[10] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016 [cit. 2021-12-30]. http://www.deeplearningbook.org.

[11] Google Brain Team. *Knihovna TensorFlow*. 2022 [cit. 2022-05-01]. Dostupné z: https://www.tensorflow.org.

[12] Hamerník, P. *Využití hlubokého učení pro rozpoznání textu v obrazu grafického uživatelského rozhraní*. Brno, CZ, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Lysek. Dostupné z: https://www.fit.vut.cz/study/thesis/22520/.

[13] Ing. Vojtěch Mrázek Ph.D.. *Laboratorní cvičení BIN4 : Neuronové sítě*. 2022 [cit. 2022-05-01]. Dostupné z: https://github.com/mrazekv/bin-lab-nn.

[14] JAVOZ. *Manuál ke hře Frantom staré Prahy*. 1987. Dostupné z: https://www.zatrolene-hry.cz/soubory/583/7218.pdf.

[15] Kocsis, L. a Szepesvári, C. Bandit Based Monte-Carlo Planning. *Machine Learning: ECML 2006*. 1. vyd. 2006, č. 4212, s. 282–293. Dostupné z: https://link.springer.com/chapter/10.1007/11871842_29.

[16] Matplotlib development team. *Knihovna Matplotlib*. 2022 [cit. 2022-05-16]. Dostupné z: https://matplotlib.org.

[17] Naumetc, D. *Artificial Neural Networks in plane control*. Valkeakoski, FI, 2016. Bakalářská práce. Häme University of Applied Sciences. Dostupné z: https://www.theseus.fi/bitstream/handle/10024/123282/Naumetc_Daniil.pdf.

[18] Neřád, V. *Umělá inteligence pro hraní her*. Brno, CZ, 2012. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kouřil. Dostupné z: https://www.fit.vut.cz/study/thesis/12165/.

[19] Nielsen, M. *Neural Networks and Deep Learning* [online]. 2018 [cit. 2021-12-30]. Dostupné z: https://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf.

[20] Nijssen, J. a Winands, M. Monte-Carlo Tree Search for the Game of Scotland Yard. In: IEEE. *Computational Intelligence and Games (CIG'11)*. 2011, s. 158 – 165. DOI: 10.1109/CIG.2011.6032002. ISBN 978-1-4577-0009-5. Dostupné z: https://www.researchgate.net/publication/224259872.

[21] NumPy Community. *Knihovna NumPy*. 2022 [cit. 2022-05-01]. Dostupné z: https://www.numpy.org.

[22] Pygame Community. *Kolekce grafických knihoven Pygame*. 2022 [cit. 2022-05-01]. Dostupné z: https://www.pygame.org/.

[23] Ravensburger. *Manuál ke hře Scotland Yard*. 2. vyd. 2000. Dostupné z: https://desktopgames.com.ua/games/199/scotlandyard_rules_en.pdf.

[24] Ravensburger. *Herní plán ke hře Scotland Yard*. 4. vyd. 2020.

[25] Russel, S. a Norvig, P. *Artificial Intelligence, A Modern Approach*. 4. vyd. Pearson, 2020. ISBN 978-0-13-461099-3.

[26] Sova, M. *Strategická desková hra s neurčitostí.* Brno, CZ, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, Ph.D. Dostupné z: https://www.fit.vut.cz/study/thesis/23706/.

[27] Tulušák, A. *Strategická desková hra s neurčitostí.* Brno, CZ, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, Ph.D. Dostupné z: https://www.fit.vut.cz/study/thesis/22304/.