VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

STRATEGICKÁ DESKOVÁ HRA S NEURČITOSTÍ

STRATEGIC GAME WITH UNCERTAINITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MARTIN GERŽA**
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 bakalářské práce

Student: **Gerža Martin**
Program: Informační technologie
Název: **Strategická desková hra s neurčitostí**
Strategic Game with Uncertainity
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s pravidly deskových her, kde aktuální stav hry bývá pro hráče po většinu času utajený. Mezi takové deskové hry patří například hra "Scotland Yard".
2. Určete metody, které by měly být důvodně vhodné pro realizaci systému, který bude hrát tuto hru autonomně. Zaměřte vedle klasických metod hraní her i metody pro počítačové učení, jako jsou například metody posilovaného učení a hlubokého učení.
3. Pro jednotlivé role figur ve hře, to znamená jak pro figuru, která je hledána, tak pro figury, které ji hledají, implementujte algoritmy řízení a ověřte jejich schopnost plnit zadané cíle.
4. Vyhodnoťte úspěšnost obou stran hry pro různé míry zapojení metod strojového učení a diskutujte zjištěné výsledky.

Literatura:

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

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání.

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

Vedoucí práce: **Zbořil František, doc. Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Tato práce je zaměřena na realizaci systému pro hraní deskové hry Scotland Yard autonomně a porovnání tohoto systému s jemu podobnými. Zaměřil jsem se na získání dostatečných informací o možnostech metod, které by měly být pro takový systém vhodné a rozhodl jsem se realizovat tento systém za pomocí metody Monte Carlo Tree Search. Výsledná realizace systému byla podrobena testování vůči podobným systémům, přičemž bylo dosaženo výborného výsledku proti jinému systému, který využíval totožnou metodu. Proti systému využívajícímu metody Alfa-Beta bylo dosaženo výsledků vyrovnaných. Hlavním výsledkem práce je funkční verze autonomního systému pro hraní hry Scotland Yard na zmenšeném poli. Zároveň je poskytnuta možnost využití dvou podobných systémů v rámci jednoho programu za účelem porovnávání jejich realizací.

Abstract

This thesis focuses on the implementation of a system for playing the board game Scotland Yard autonomously and also focuses on a comparison of this system with similar ones. I focused on obtaining enough information about the possible methods that should be suitable for such a system and decided to implement this system using the Monte Carlo Tree Search method. The result implementation of the system was tested against similar systems, achieving an excellent result against another system that used an equivalent method. There was achieved a balanced result gainst a system that used the Alpha-Beta method. The main result of this work is a working version of an autonomous system for playing the game Scotland Yard on a reduced field. It also provides the possibility of using two similar systems within a single program in order to compare their implementations.

Klíčová slova

deskové hry, strategické hry, hry s neurčitostí, Scotland Yard, Monte Carlo Tree Search, autonomní hraní, algoritmy řízení, strojové učení, metody hraní her

Keywords

board games, strategy games, games with uncertainty, Scotland Yard, Monte Carlo Tree Search, autonomous gaming, control algorithms, machine learning, gaming methods

Citace

GERŽA, Martin. *Strategická desková hra s neurčitostí*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. František Zbořil, Ph.D.

Strategická desková hra s neurčitostí

Prohlášení

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

.....
Martin Gerža
4. května 2022

Poděkování

Rád bych tímto poděkoval svému vedoucímu práce doc. Ing. Františkovi Zbořilovi, Ph.D., za odborné vedení, konzultace a podnětné návrhy k práci.

Obsah

Kapitola 1

Úvod

Deskové hry byly odjakživa považovány nejen za prostředek zábavy, ale také jako velice dobrý nástroj sloužící pro trénování logiky a lidského mozku jako takového. V dnešní době máme mnoho takových her k dispozici nejen jako fyzické verze vytvořené z kartonu a plastu, ale čím dál tím více se do tohoto odvětví začíná zapojovat svět digitální techniky. Některé takové hry by dokonce bez využití mobilního telefonu, tabletu či počítače nemusely jít ani hrát, jelikož jsou jejich pravidla přímo na této technologii postavena. Nutno podotknout, že takových titulů stále není mnoho.

S rychlým posunem technologií vpřed se také začaly deskové hry převádět do digitální formy, přičemž zde, mimo klasické hraní více hráčů (ať už na jednom či více zařízeních), přišla na scénu i varianta hraní člověka s počítačem – umělou inteligencí. Tato umělá inteligence soupeře má obvykle za úkol vyrovnat se člověku za cílem co nejlepšího herního zážitku. Přesto se velice často stává, že i ta nejvyšší nastavená obtížnost je v některých hrách nakonec velmi jednoduše překonatelná. Z tohoto důvodu jsou poté hráči, kteří se v daných hrách chtějí zdokonalovat za cílem vítězství v soutěžích nuceni přejít zpět k nejčastějšímu způsobu tréninku – hry s druhým reálným hráčem.

Mezi takovéto hry patří především klasičtější hry jako jsou například šachy, ale čím dál tím více se začínají objevovat i různé ať už zábavové či vážnější soutěže v různých deskových hrách. I za tímto účelem se začalo odvětví umělé inteligence více rozvíjet.

V rámci této práce je zapojen jeden z velice důležitých faktorů, které může desková hra mít – neurčitost. Desková hra Scotland Yard v této vlastnosti vyniká, jelikož jedna strana takřka většinu času netuší, kde se vyskytuje protivník, kterého se snaží na rozlehlé mapě chytit. V oblasti umělé inteligence pro tuto hru je proto již tento faktor mírnou zátěží, jelikož mnoho metod, které se standardně využívají, se dokáže rozhodovat čistě pouze dle aktuálního stavu na hrací ploše. V tomto specifickém úkolu již proběhlo několik řešení jak ve světě, tak přímo v rámci bakalářských prací na VUT FIT. Tato práce si klade za cíl především předešlá řešení zdokonalit za použití správných metod a obecně poté porovnat úspěšnost vlastní verze s předešlými.

V následující kapitole je nejprve popsán úvod do strategických deskových her s neurčitým stavem, přičemž jsou poté více do hloubky popsána pravidla hry Scotland Yard. Na tuto kapitolu navazuje pojednání o vhodných metodách realizace systému, který by zvládl Scotland Yard hrát autonomně. Následně jsou v další kapitole popsány aktuálně dosažené výsledky. Hlavně pak ze dvou bakalářských prací, které využily metod Alfa-Beta a Monte Carlo Tree Search. V poslední kapitole je poté popsána vlastní implementace systému založeného také na metodě Monte Carlo Tree Search a celkový popis experimentů se srovnáním výsledků při postavení všech jednotlivých systémů proti sobě.

Kapitola 2

Strategické deskové hry s neurčitým stavem

Deskové hry jsou v mnoha případech založeny na strategii a závislé na tazích. Vždy, pokud chce hráč dané hry vyhrát, musí přijít s určitou strategií, jak provede jednotlivé tahy a jak bude reagovat na tahy protihráče. Toto celé využití strategických pohybů je však o to těžší při přidání faktoru náhody, jako jsou například hody kostkou nebo třeba slepý výběr z karet. Dalším faktorem ovlivňujícím hru je provedení některých tahů neviditelně pro soupeře, což se stalo i důležitým faktorem v rámci zaměření této práce. V této kapitole jsou detailněji popsány deskové hry s neurčitostí. Nejprve je výskyt neurčitosti v pravidlech těchto her popsán obecně na několika příkladech. Poté následuje popis pravidel a celkový koncept hry Scotland Yard, která je hlavním předmětem této práce. Přičemž hned po tomto popisu následuje vysvětlení zavedení určitých zjednodušených pravidel této hry tak, jako tomu již bylo v předešlých verzích (bakalářských pracích Adriána Tulušáka [10] a Michala Sovy [8]).

2.1 Výskyt neurčitosti v pravidlech různých her

Neurčitost jako taková se dá jednoduše popsat jako skrytí jednoho či více tahů (akcí) před protihráčem. Dá se ale také popsat například jako provedení umístění nějaké suroviny nebo figurky na danou pozici, přičemž se po několika kolech hry odhalí, kam tato surovina či figurka byla položena. Vše toto spadá na nějakou dobu do neurčitého stavu, kdy hráč takové hry nemůže přesně určit, jak hra aktuálně probíhá. V takové situaci již záleží velice na strategii, která se obvykle odvíjí dle toho, o jakou hru se přesně jedná. V některých hrách je využito dopočítávání, například při hrách s využitím karet. V těchto hrách se dá odhadovat zbylý počet karet s nějakou danou hodnotou či funkcí tak, aby hráč dosáhl co nejlepšího pořadí zahrání karet.

Stratego

Jakožto velice dobrý příklad takovéto hry je určitě nutno zmínit hru *Stratego* (česká pravidla hry viz [4]). Hraje se na čtvercové hrací ploše tvořené ze 100 hracích polí, na která je umístěno 40 červených a 40 modrých figur s různými vlastnostmi a hodnostmi. Již od začátku hry je rozmístění figur naprosto utajeno protihráči. V takové situaci je tudíž nutno zapojit logiku odebírání možných figur, které již byly ze hry pomocí útoku odstraněny. Stav hry se tudíž mění na základě provedených tahů a v případě, kdy dojde ke střetu dvou figur

na jednom poli. Celá hra tedy disponuje vysokou neurčitostí, jelikož až do chvíle střetu dvou figur může hráč jen odhadovat, jakého typu je daná protihráčova figura.

6 bere!

Druhou hrou, na které se dá hezky zobrazit příklad neurčitosti je hra *6 bere!* (česká pravidla hry viz [2]). Jedná se o karetní hru, při které se na stůl vykládají karty hráčů seřazeny dle jejich hodnot. Na stole jsou vyznačeny 4 řady pod sebou, ve kterých jsou tyto karty seřazovány. Jakmile hráč překoná v jedné z řad svou kartou počet 6 karet, bere všechny karty z dané řady. Jelikož je karet přesný počet 104 a každá je ohodnocena hodnotou od 1 do 104, tak vzniká velice jednoduchá strategie – vybrat kartu vždy tak, aby nebyla dosazena jakožto 6. v některé z řad. V tomto případě se dá odhadovat, jak budou vybírat ostatní hráči a zároveň odstraňovat možnosti dle již zahraných karet. Neznámý stav hry tak nastává vždy, když v daném kole musí všichni hráči vybrat jednu kartu ze své ruky lícem dolů na stůl a poté všichni najednou odhalí svou hodnotu a od nejmenší karty po největší provedou umístění. V takové situaci může hrát určitou roli i náhoda, ale pokud jsou hráči již se hrou zkušení, nastává veliká snaha co nejlépe dopočítat, která karta je v daném stavu hry nejvýhodnější.

Zaměření na strukturu tahů

Vzhledem k předešlým dvěma příkladům bylo poukázáno na fakt, jak utajenost stavu hry může výrazně ovlivňovat hru jako takovou. Oba příklady jsou však součástí kategorie her, které jsou hrány tahově v kolech. Tento typ hraní je považován za základní vzhledem k jeho jednoduché struktuře. Jednou za kolo provede každý hráč jeden svůj tah danou figurkou, táhne jednu kartu či provede jednu danou akci. Mnoho her však rozšiřuje tyto možnosti například na více akcí za jeden tah hráče. Tyto akce se mohou vykonávat buď v určitém pořadí, nebo jsou provedeny dle rozhodnutí hráče.

Jednou z takových her je například hra *Charterstone* (pravidla viz [9]), ve které je hráči přisuzováno během tahu udělat různý počet umístění figurek, které spouštějí různé dodatečné akce budov, na které byly umístěny. V takovém případě může nastat v tazích dokonce chaos, jelikož některé budovy mohou hráči tah o mnoho akcí prodloužit. Ani protihráč však nepřijde zkrátka, protože má možnost do takových možných prodloužení vstoupit a využití některých budov zabránit i mimo jeho vlastní tah.

Neurčitost a umělá inteligence

Neurčitost se v pravidlech různých her vyskytuje v mnoha podobách. Někdy ovlivňuje hru jen mírně, ale v jiné hře zase může být základním stavebním kamenem celých pravidel. Do mnoha her je takto dodána míra ovlivnění stavu hry tak, aby hráči nemohli přesně určit následujících několik tahů protihráče.

Ohledně umělé inteligence, která se čím dál tím více zapojuje do různých variant hraní lze říci, že se velice dobře začíná projevovat ohledně konkurence lidským hráčům. Existuje mnoho různých systémů pro hraní různých tahových her. Tyto systémy jsou využívány jakožto prostředky pro trénink a zdokonalování jejich hráčů či jako zapojení určité zpětné odezvy pro hráče na profesionální úrovni.

Co se týče zapojení umělé inteligence do provedení nějaké hry s pravidly obsahujícími neurčitý stav je těžké odhadnout, jak moc bude systém složitý jak na implementaci dostatečné logiky, tak na možnosti výpočetních prostředků. Některé metody vyhodnocování

nejlepších možných tahů ve složitějších hrách mohou totiž dospět k velice vysokým nárokům na paměť výpočetní techniky. V takových případech je poté dle situace nutno buď pravidla převádět na jednodušší varianty, kdy se dá stále říci, že jsou daná pravidla dostatečná dle požadavků, nebo pokud opravdu chceme udržet celá pravidla, je nutno výpočetní výkon posílit. V případě zjednodušených pravidel poté již záleží na stanovených požadavcích na systém. Pokud by tedy měl systém sloužit pro trénování hry reálným hráčem, je nutno docílit nejlépe stejných pravidel. Oproti tomu v případě porovnávání úspěšnosti různých metod řešení daného systému nemusíme docílit naprosté identičnosti vůči prvotním pravidlům hry.

2.2 Scotland Yard — podrobný popis pravidel a výskytu neurčitosti

Desková hra Scotland Yard vyšla roku 1983, kdy také získala ocenění *Spiel des Jahres*. Jedná se o tahovou strategickou hru, kdy se snaží jeden z hráčů jakožto Mistr X uniknout ostatním protihráčům představujícím agenty policejního sboru. Celá hra se odehrává na mapě obsahující 200 různých zastávek, po kterých se hráči pohybují třemi druhy dopravních prostředků. Celá hra by takto představovala pouze taktické pohyby policejního sboru blíže k Mistru X. V tomto případě je zde však důležitým faktorem schopnost Mistra X pohybovat se mezi místy skrytě a ukázat se vždy jen jednou za určitý počet kol. Hře tento prvek okamžitě dodává určitou míru obtížnosti při odhadování tahů Mistra X.

Podrobný popis pravidel hry

Celková pravidla hry jsou dostupná viz [5]. Hrací deska se skládá konkrétně z 200 míst, na kterých lze postavit figurku. Část této desky je pro lepší představu vyobrazena na obrázku 2.1.

Hrají proti sobě jeden Mistr X a čtyři až pět agentů, kteří jsou při menším počtu hráčů mezi ně rozděleni. Po hrací desce se tedy lze pohybovat pomocí tří druhů dopravních prostředků, přičemž za každý prostředek musí hráči hrající za agenty policie odhodit příslušný lístek. Druhy dopravy a počet lístků, které na počátku hry agenti obdrží je následovný:

- Taxi – 10 lístků, krátké vzdálenosti, propojeny skoro všechny stanice
- Autobus – 8 lístků, delší vzdálenosti mezi dvěma až pěti stanicemi
- Metro – 4 lístky, dlouhé vzdálenosti, jen pár stanic na mapě

Těmito počty dopravních lístků jsou tedy agenti omezeni. Mistr X může cestovat bez omezení počtu využití některého z prostředků. Na počátku hry si tedy každý hráč vylosuje počáteční pozici a na tu se postaví s výjimkou Mistra X, který začíná rovnou skrytě. Poté v každém kole vždy provede tah Mistr X a po něm mohou každý z agentů provést 1 svůj tah. Takto se vše opakuje maximálně 22 tahů. Pokud ani při posledním tahu nevstoupí agenti na místo, kde stojí Mistr X, nebo jej v průběhu hry neobklíčí, je Mistr X prohlášen za vítěze.

Odhalení pozice Mistra X je nejlepším vodítkem agentů k jeho dopadení. Mistr X se takto odhalí čtyřikrát za celou hru, a to v kolech 3, 8, 13 a 18. Po každém svém tahu vezme lístek příslušného prostředku, který použil a dosadí jej do desky jeho pohybů. Agenti takto alespoň vědí, jaký prostředek byl použit. Pro Mistra X jsou přístupné ještě dva druhy zvýhodnění (lístků). Těmi jsou dvě možnosti dvojitého tahu a pět možností tahu se

Obrázek 2.1: **Ukázka části hrací plochy hry *Scotland Yard*** Zdroj: Vlastní snímek

skrytým použitým dopravním prostředkem. Při využití tahu se skrytým prostředkem může Mistr X také díky tomuto lístku využít trajektu, který se vyskytuje v dolní třetině hrací plochy. Agenti tímto trajektem cestovat nemohou. I proto pro ně tedy není nijak jednoduché Mistra X dopadnout.

Výskyt neurčitosti

Z hlediska neurčitosti jsou pravidla této hry dobře obsažena. V rámci každého skrytého tahu se neustále zvyšuje počet možností, kde by mohl Mistr X stát. Nutno podotknout, že v rámci kol 1 a 2 jsou pohyby agentů čistě taktickou přípravou na kolo 3, kdy se poprvé Mistr X odhalí. V těchto prvních dvou kolech je tedy rozhodně výhodné, aby se agenti snažili dostat do stanic metra či alespoň autobusů tak, aby ve třetím kole měli co nejvíce možných stanic pokrytých, a zároveň se mohli rychleji přesunout na kratší vzdálenost ti agenti, kteří budou od Mistra X nejdále. Od momentu prvního odhalení Mistra X lze již postupovat systémem odhadování následujících tahů. Nejen, že na mapě lze vidět všechna možná místa kam mohl Mistr X jet, ale zároveň pokud nevyužil skrytých tahů, lze odvozovat jeho pozici i dle použitých dopravních prostředků. Mistr X tak musí velice dobře volit, zda bude chtít dle dané situace utéci delší vzdálenost, ale riskovat tak mnohem menší počet možností tahů kde by jej mohli agenti čekat, nebo raději použít taxík, který má místa nasednutí a vysednutí takřka všude na mapě.

Pro názornou ukázku počtu možností se můžeme podívat na následující obrázek 2.2. Aktuálně nastalo čtvrté kolo, kdy má začít opět Mistrem X (bílá figurka), který se z odha-

lené pozice má pohnout na nové místo (odstraní svou figurku z dané pozice a poté si zapíše pozici následující a založí ji kartičkou příslušného použitého prostředku). Na pozicích 48 a 79 se vyskytují dva agenti, kteří sem došli v kole třetím. Mistr X tak má možnost přesunout se za pomocí taxíku (žluté cesty) na pozice 64 a 80 nebo za pomocí autobusu (zelené cesty) na pozice 34 a 100. V této situaci se tedy jeví pozice 34 jako nejméně výhodná, jelikož na ni může hned v daném kole červený agent vstoupit. Mistr X by totiž v případě nepoužití tahu se skrytým dopravním prostředkem a pouze zobrazení, že použil autobus, mohl tím pádem stát jen na pozici 100. Dal by tak totiž jasné vodítko, na jakou pozici se dále mají agenti zaměřovat.

Obrázek 2.2: **Ukázka možností tahu Mistra X (bílá figurka)** Zdroj: Vlastní snímek

Neurčitost ohledně skrytých tahů je tedy vysoká, jelikož mezi koly 3 a 8 může nastat opravdu veliké množství tahů, u kterých neustále narůstají možnosti dalších tahů. Agentům tedy nezbývá nic jiného než se snažit v takových chvílích odpočítávat místa, kde nemůže Mistr X určitě být a snažit se dostat na ta místa, kde by se mohl vyskytovat. Mistr X tak může tedy dosáhnout při použití všech výhod velice rychle výhodné pozice, ale i jeho výhody jsou dostatečně omezeny, aby hra nebyla pro jednu ze stran obtížnější.

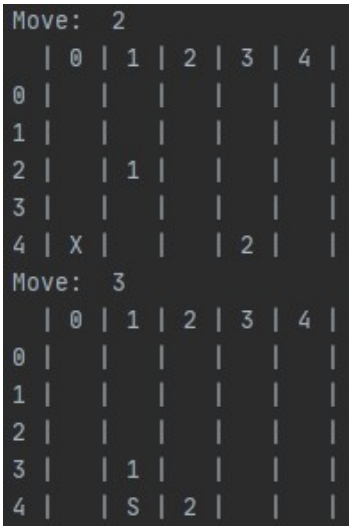
2.3 Scotland Yard – zjednodušená pravidla

Dřívější dvě bakalářské práce založené na stejném tématu byly řešeny za použití zjednodušených pravidel. Tato pravidla proto byla použita i v rámci řešení této práce, jelikož jedním z jejích cílů bylo pokusit se o zdokonalení předešlých prací a porovnat je mezi se-

bou. Z tohoto důvodu je tato část věnována popisu pravidel, která byla zavedena tak, aby bylo potřeba méně výpočetních prostředků, ale i tak se hra nestala triviálním úkolem.

Popis pravidel

Základní plocha je čtvercová s délkou strany 5 políček. Na začátku hry se náhodně rozmístí jak Mistr X, tak dva agenti každý na jiné políčko. Hra se hraje na 15 kol, přičemž Mistr X se odhaluje každé třetí kolo. Pohyb je na této mapě omezen čistě na jedno sousední políčko, tudíž se vytrácí možnost zavedení logiky použití dopravních prostředků a zároveň také Mistr X nemůže použít dvojitý tah. Jinak jsou ostatní části pravidel totožné. Na následujícím obrázku 2.3 je ukázáno, jak toto pole vypadá v rámci dvou tahů po sobě, kde vidíme agenty označeny čísly **1** a **2** a Mistra X označeného písmenem **X** když je skryt, potažmo **S**, když je viděn. Jeden tah (na obrázku označen jako *Move*) je tedy vždy vykreslen již po provedení tahů všech figurek najednou (nejdříve Mistr X a po něm agent 1 a 2).



Obrázek 2.3: **Plocha hry *Scotland Yard* při zjednodušených pravidlech** Zdroj: Vlastní snímek

Výskyt neurčitosti

Neurčitost se v rámci tohoto jednoduššího provedení nijak neliší od výskytu v původních pravidlech, jelikož jsme opět od třetího kola odkázáni na pozici Mistra X tam, kde se odhalil. Poté opět můžou agenti odhadovat místa, na která se mohl pohnout, a tak se snažit jej neustále obklíčovat a určitým způsobem mu nadbíhat tak, aby měl čím dál tím menší počet možných pohybů. U této zjednodušené verze, kde využíváme čtvercová pole, je vždy každé figuře zpřístupněn počet 2 až 4 možných tahů podle toho, kde se zrovna hráč nachází. V takovém momentu se tedy jeví jako nejvýhodnější pro Mistra X utíkat od agentů co nejdále, ale nejlépe se neocitnout v rohu, jelikož při hře dvou agentů dle zjednodušených pravidel by zde mohl být velmi brzy obklíčen.

Taktika agentů je zde tedy následující. Nejdříve se během prvních dvou kol rozmístit do prostoru mapy. Jakmile se Mistr X ve třetím kole zobrazí, dostat se nejlépe do pozic tak, aby jej mohli co nejvíce omezit. V případě že jej nechytili, tak v dalším kole zkoušet pozice sousední jeho minulé pozici, popřípadě se pohybovat dále na možné obsazené pozice, ale

už opět zároveň tak, aby k němu měli v kole následujícím co nejlepší přístup, jelikož se má znova odhalit. Takto neustále postupuje hra, dokud se buď taktika vydaří nebo Mistr X zvládne po celou dobu unikat.

Z důvodu odstranění dopravních prostředků z pravidel byla v této variantě neurčitost mírně navýšena, jelikož agenti nemají tušení, jak se Mistr X pohyboval. Ale jelikož se vzhledem k tomu i změnila mapa na jednodušší ztvárnění s méně zastávkami a méně propojeními mezi nimi, dostatečně se pro obě strany pravidla srovnala. Vzhledem k tomu i úprava odhalování Mistra X častěji vedla k vytvoření oboustranně vyrovnaného soupeře.

Výhody a nevýhody

Výhodami tohoto provedení hry je nižší náročnost na výpočetní výkon při realizaci systému, který by měl hru hrát autonomně. Zároveň se na jednodušší verzi lépe demonstruje pohyb jednotlivých hráčů v závislosti na jejich přemýšlení a taktice. V takovém případě při trénování hráče v základních principech hry by tato zjednodušená verze mohla být naprosto dostatečným prvkem, než by přešel na její celou verzi se všemi dalšími možnostmi. Logika tahů totiž zůstává naprosto nezměněna, právě naopak dokonce mírně zjednodušena tak, aby si hráč dokázal opravdu představit všechny možné stavy, které by mohly nastat v daném kole či po některém z vybraných tahů. V takovém případě poté můžou být jednodušší pravidla přehlednější a sloužit jako dostatečný nástroj pro prvotní vytvoření taktiky, jak nejlépe hru hrát a dosáhnout tak kladných výsledků.

Nevýhodou se poté bere pravý opak, kdy zjednodušená hra neprovede hráče, který ji hraje všemi koncepty, které plná hra poskytuje. V rámci hraní plnohodnotné hry se musí počítat i s omezením pohybů různými prostředky, které výrazně ovlivňuje dynamiku tahů. Hráč tak ve zjednodušené hře nemusí vůbec přemýšlet nad strategií, který lístek je ochotný pro danou situaci odhodit, což je důležitým faktorem. Tento faktor by se však postupným zaváděním vylepšení zjednodušené verze dal doplnit.

Možná vylepšení a posunutí zjednodušené verze

Zjednodušená verze by se dala poupravit za pomoci různých vylepšení tak, aby co nejlépe doplňovala své nedostatky co nejblíže verzi plné. Díky tomu by se poté dalo dosáhnout například verze, kde by byla přidána ještě jedna mapa, která by v některých místech vedla na místa mapy první, a tak by se dal zavést systém lístků, jako tomu je v plných pravidlech. Toto by se dalo samozřejmě rozvíjet i v rámci velikosti map, které by šlo rozšířit, či některá místa úplně odstranit tak, aby se z mapy nestal pouze jen čtverec, ale měla by vícestrannou formu. Tímto způsobem by se poté dalo celou hru co nejvíce přibližovat plné verzi tak, aby se dalo prostřednictvím několika verzí trénovat jen určité aspekty dané hry.

Trénink na několika verzích, které se každá zaměřují na jiné stránky hry by byl určitě velice výhodným nástrojem. Nemuselo by to znamenat mnoho pro jednu takovou konkrétní hru jako je Scotland Yard. Ale při použití této metody na některé další deskové hry by se mohlo uplatnění této techniky hodit. Vylepšení jsou při zvládnutí zjednodušené verze výhodná, pokud jsou zavedena tak, aby poté pomohla při hraní s pravidly plné hry.

Jednodušší verze hry je tedy velice výhodnou alternativou jak v oblasti vytvoření určitého systému, který by ji měl hrát autonomně, jelikož bude výhodnější při práci s volnými výpočetními prostředky, ale také na ní lze lépe vidět rozdíl dopadů jednotlivých tahů na stav hry. Přičemž se stále dá postupně vylepšovat v závislosti na úspěchu daného systému, jeho prostředcích, ale i požadavcích, které může stanovit i hráč, který by ji chtěl využít k vlastnímu tréninku taktiky, či jen tak obyčejné hře.

Kapitola 3

Metody strojového hraní her

Tato kapitola se bude věnovat podrobnému popisu různých metod, které již byly použity na implementaci systémů pro hraní her. Zároveň se zaměří na vyhovující metody, které by se daly použít pro realizaci takového systému, sloužícího k hraní her s neurčitým stavem. Jelikož již byly dvě metody na realizaci systému pro hraní zjednodušené verze Scotland Yard použity, bude popis jejich úspěšnosti popsán v kapitole 4. Zároveň tam budou také podrobněji rozebrána úskalí těchto metod v tomto konkrétním případě. I proto se tato kapitola zaměřuje spíše obecněji na celkovou škálu použitelných metod u her s neurčitostí i bez ní a popisuje doposud využité metody na řešení různých podobných případů. Následující části této kapitoly jsou podpořeny studijní oporou předmětu *Základy umělé inteligence* (IZU) viz [11] a publikací S. Russella a P. Norviga [7].

3.1 Základní metody

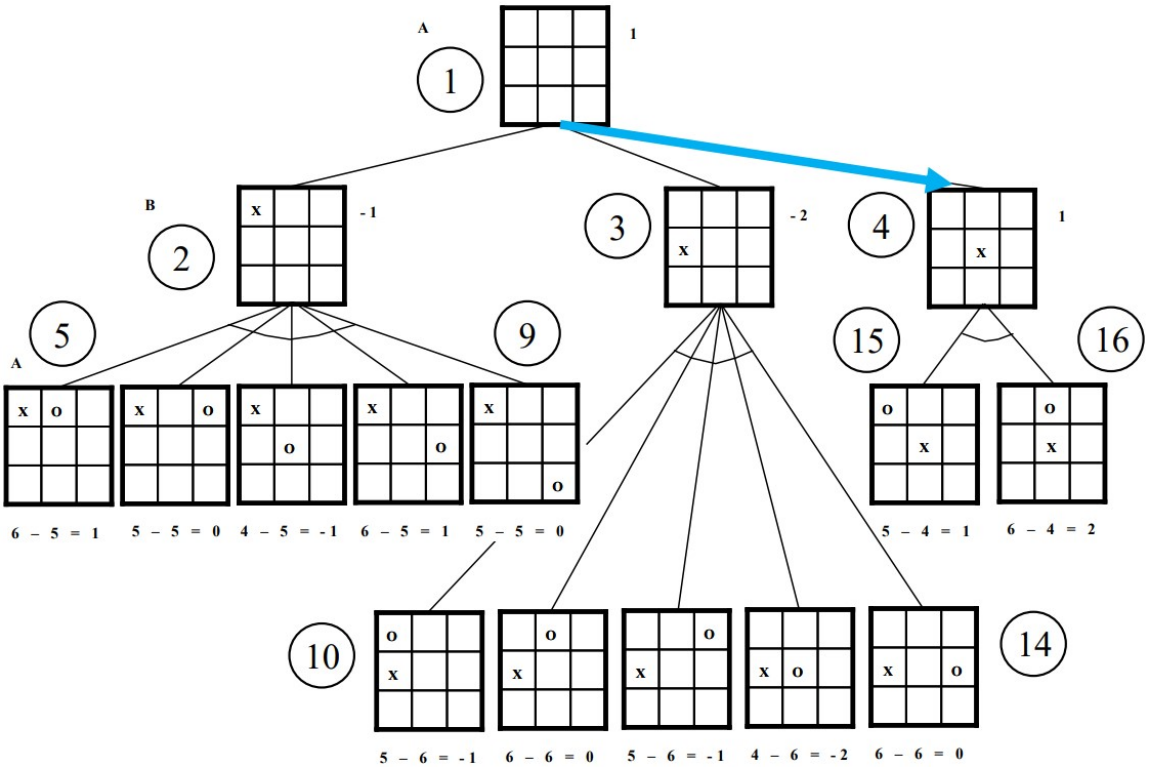
Jednodušší hry, kde je možno obsáhnout do stavového stromu všechny možné tahy obou hráčů až do konce hry lze velice dobře vyhodnocovat základními metodami. Stavovým stromem je v tomto případě myšlen AND/OR graf, kdy na něm můžeme využít neinformované či informované metody (AO či AO* algoritmus). U těchto dvou základních algoritmů mohou být využita klasická procházení stavových stromů, kdy jednoduše projdeme celým stromem například pomocí slepého prohledávání do šířky (BFS), nebo slepého prohledávání do hloubky (DFS). Takové využití metod přichází u neurčitosti jen na naprosto jednoduché hry, které obsahují opravdu jen pár tahů. Velice dobrým příkladem takové hry, kde by se daly tyto metody využít jsou *Piškvorky* s velikostí hracího pole 3x3 jinak taky pojmenovávány jako *Tic Tac Toe*. V takovém případě by bylo možné tuto hru zahrát, avšak při zvětšení hracího pole by se stavový strom velice rozšířil a nastala by velká náročnost na výpočetní prostředky, tudíž by se staly metody velice kontraproduktivními. Tím pádem jsou základní metody velice nedostatečnými při hraní nějaké hry s neurčitostí.

V takovém případě je tedy nutno se zaměřit na metody, u kterých vždy nemusí být potřebné dojít až k celkovému závěru hry, ale může jim stačit pouze několik následujících tahů obou hráčů k určení vhodného tahu pro zahrání. Takové metody nemusí být sice v některých případech naprosto spolehlivé, ale pokud jsou dobře nastaveny vzhledem k promýšlení všech tahů a berou v potaz všechny aspekty hry, dá se z nich i tak vytvořit velice silný soupeř vůči hráčům lidským.

MiniMax

Jedna ze základních metod pro strojové hraní her se nazývá MiniMax, která se používá při hraní her dvou hráčů. Jejím základem je stejnojmenná procedura, která je založena na procházení stavového stromu. Dle předem nastavené hloubky se prochází postupně všemi možnými uzly tak, aby poté procedura vrátila ohodnocení aktuálního uzlu a tah, který k danému ohodnocení vedl. Poté podle toho, které ohodnocení uzlu bylo nejvyšší, je tah s tímto ohodnocením vybrán jakožto nejlepší na použití. Názornou ukázkou takového stromu je následující obrázek 3.1 zobrazující využití metody MiniMax při hře *Piškvorky*.

Na obrázku lze vidět postupné ohodnocení listových uzlů a jejich zpětnou propagaci zpět do svých rodičovských uzlů za účelem získání nejvýhodnějšího tahu v rámci piškvorkového pole o velikosti 3x3. Kořenový uzel je v úplně počátečním stavu, tudíž je hrací pole zatím prázdné, a tak se provede rozhodnutí, kam bude umístěn první křížek. Jelikož je zvolena hloubka zanoření o hodnotě 2, jsou tak v jednotlivých potomkovských uzlech demonstrovány vždy dva tahy. Je tak vždy proveden jeden tah hráče s křížkem a následně druhý tah hráče s kolečkem (jeho reakce na aktuální stav hry). Touto metodou je tedy dosaženo zahrání tahu, který se jeví z hlediska dvou tahů budoucích jako nejvýhodnější – na obrázku 3.1 vyznačen modrou šipkou.



Obrázek 3.1: **Minimax stavový strom pro piškvorky** Zdroj: Studijní opora IZU viz [11]

MiniMax – nevýhoda

Hlavní nevýhodou využití této metody je prohledávání všech uzlů stavového stromu. I když se díky zaměření se jen na několik následujících tahů strom zmenšil, dochází v mnoha různých hrách na přílišné větvení. Toto větvení uzlů není vnímáno jako chybné, jelikož

je nutno prozkoumat všechny možné tahy, které lze provést. V některých případech však může nastat při průchodu jednoho rodičovského uzlu, že hned při prvním potomkovi bude získáno nejlepší ohodnocení, které lze v daném rodiči dosáhnout. V takové chvíli však tato metoda neobsahuje žádnou vymoženost, díky které by byla schopna procházení již zastavit a propagovat zpět tuto nejlepší hodnotu. Kvůli tomu nastane i průchod dalších potomků a průchod stromem je zbytečně prodloužen čímž se celkové určení nejvýhodnějšího tahu stává náročnějším na čas.

V takovém případě je tedy nutno se podívat na nějakou z pokročilejších metod, která by mohla tuto nevýhodu odstranit a nejlépe ještě více pomoci při procházení stavového stromu. I tak se dá o metodě MiniMax říci, že je dostatečně sofistikovanou při využití u jednodušších her, jak již bylo zmíněno. Při rozhodování se, jakou metodu je vhodné na daný problém použít, je tedy tento výběr velice závislý i na obsáhlosti úkolu, který je nutno řešit.

3.2 Metoda Alfa-Beta prořezávání

Tato metoda je založena na metodě MiniMax, přičemž je v několika věcech upravena tak, aby bylo dosaženo kratšího průchodu stavovým stromem. Jakmile se dojde opět na vyhodnocování stavů jako tomu bylo v metodě MiniMax, je využito dvou dalších parametrů α a β k přerušení vyhodnocování dalších potomkovských uzlů. Toto přerušení probíhá právě ve chvíli, kdy je dosaženo již vyššího (potažmo nižšího) možného ohodnocení, než by bylo dosaženo v dalších potomkovských uzlech. Důkladný popis celé metody je uveden v následující sekci, kde je popsán z pohledu obou hráčů vzhledem k zanoření ve struktuře stavového stromu. Díky tomuto rozhodování o zamezení průchodu některými tahy tedy nastává časová úspora, jelikož se jim v dané situaci jednoduše vyhneme.

Popis algoritmu Alfa-Beta

V potaz se bere hra dvou hráčů, A a B. Vstupními parametry algoritmu jsou stav herního pole hry *X*, maximální požadovaná hloubka zanoření *Max_Hl* a proměnná, určující, zda je tah hráče A či B jménem *Tah_hráče*. Posledními dvěma parametry jsou hodnoty $\alpha = -\infty$ a $\beta = \infty$. Algoritmus je rekurzivní, volá se při tahu hráče A a vrací ohodnocení aktuálního stavu a tah, který k tomuto ohodnocení vedl.

Procedura Alfa-Beta

1. Pokud je X listem nebo se aktuální hloubka zanoření rovná *Max_Hl*, vyhodnoť ohodnocení aktuálního stavu a vrať jej.

2. Pokud se *Tah_hráče* rovná hodnotě "A", proveď následující příkazy, pokud se rovná hodnotě "B", přeskoč na krok číslo 3.

   (a) Dokud je hodnota α menší než hodnota β ($\alpha < \beta$), volej proceduru Alfa-Beta pro další možný tah vycházející z tohoto stavu X, přičemž jsou této následující proceduře předány parametry s aktuálními hodnotami α a β. Při každém získání navrácené hodnoty z volání procedury vyber maximální hodnotu z navrácené a aktuální hodnoty stavu X a tuto maximální hodnotu nastav do α.

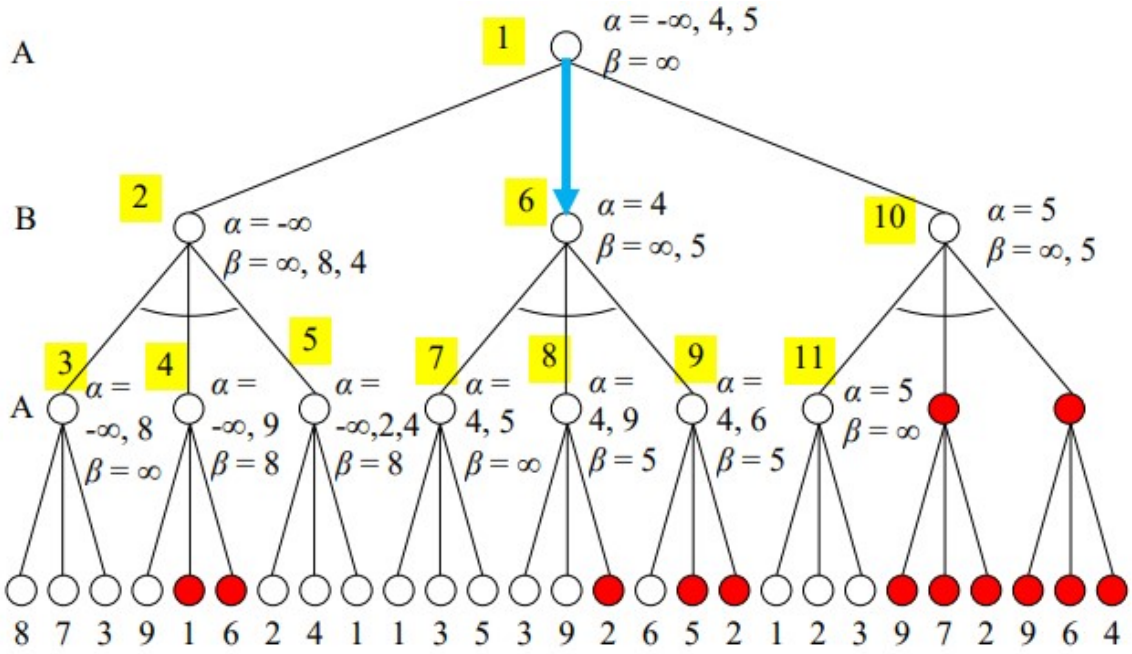
(b) V případě, že je α větší nebo rovna β ($\alpha \geq \beta$) či neexistuje-li pro uzel X další potomkovský uzel, vrať aktuální hodnotu α a jí přidělený tah, který vedl z kořenového uzlu.

3. Pokud se *Tah_hráče* rovná hodnotě "A", přeskoč následující příkazy

   (a) Dokud je hodnota α menší než hodnota β ($\alpha < \beta$), volej proceduru Alfa-Beta pro další možný tah vycházející z tohoto stavu X, přičemž jsou této následující proceduře předány parametry s aktuálními hodnotami α a β. Při každém získání navrácené hodnoty z volání procedury vyber minimální hodnotu z navrácené a aktuální hodnoty stavu X a tuto minimální hodnotu nastav do β.

   (b) V případě, že je α větší nebo rovna β ($\alpha \geq \beta$) či neexistuje-li pro uzel X další potomkovský uzel, vrať aktuální hodnotu β.

Využití této metody tedy není velice složité a dá se skvěle adaptovat na určitý styl hry dle jejích konkrétních potřeb. Jakožto ukázka takto vygenerovaného a vyhodnoceného stromu je v obrázku 3.2 naznačen průchod pomocí čísel se žlutým zvýrazněním. Je nutno si hlavně povšimnout červeně označených stavů, do kterých se tak díky algoritmu Alfa-Beta nemuselo vůbec zacházet, jelikož byly vyhodnoceny jakožto nepřínosné pro průchod kvůli jejich hodnotám v porovnání s hodnotami jejich rodičovských uzlů. Modrá šipka poté opět označuje zvolený tah, který byl názorně vyhodnocen jako nejlepší pro použití v aktuální situaci hry.



Obrázek 3.2: **Alfa-Beta ukázka stavového stromu s průchodem** Zdroj: Studijní opora IZU viz [11]

3.3 Pokročilé metody – strojové hraní her s neurčitostí

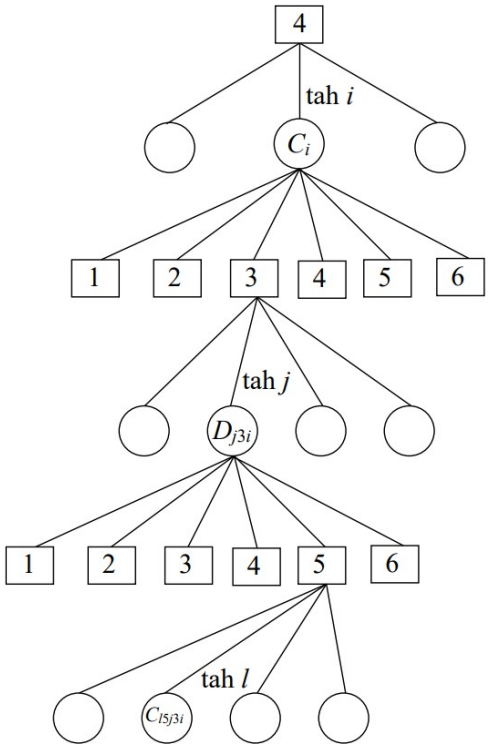
Metody sloužící pro využití na hry, ve kterých je zapojena nějaká míra neurčitosti se již řadí mezi ty pokročilejší. V některých případech vycházejí z metod základních, které byly

popsány v předešlých dvou částech této kapitoly. Je v nich však mnohem více nutno zapojit míru náhody a počítat tak s mnohem více možnostmi například v případě hodu kostkou, tahem náhodných karet nebo třeba skrytím některého z hráčů. Ve stavovém stromě může být tedy tímto způsobem vygenerováno více uzlů, které je potřeba vyhodnotit a zjistit, jak by na tom daný hráč po provedení takového tahu byl. Metody, které jsou tedy používány na řešení takových problémů mohou být náročnější na výpočetní výkon a časovou náročnost vyhodnocení. Jedním příkladem takových metod je například metoda Expectiminimax, která rozšiřuje metodu MiniMax a je jí také věnována následující podsekce.

Expectiminimax

Tato metoda velice dobře bere v potaz hru se zapojením neurčitosti například při použití hrací kostky. Při jejím využití je nutno brát v potaz pravděpodobnost (např. každého hodu kostky), která je ve stavovém stromě dosazena. Ačkoliv je tedy výběr hráčů ovlivněn určitými mírami náhody, stále se dá vybírat a ohodnocovat uzly dle toho, že se snaží hráči hrát co nejlépe tak, aby jim to co nejdříve přineslo vítězství.

Metoda Expectiminimax využívá podobného principu jako metoda MiniMax. Pro každého hráče v daném stavu bere postupně minimální potažmo maximální ohodnocení z potomkovských uzlů, které propaguje poté přes svůj uzel do rodičovského. V těchto uzlech však jak již bylo řečeno není použito pouze porovnání hodnot a vrácení nejmenší, resp. největší z nich, ale je zde zapojena i pravděpodobnost navyšující počet možných pohybů (tahů). Na následujícím obrázku 3.3 lze vidět, jak by mohla část stromu takové hry vypadat v případě hraní hry s klasickou šestihrannou kostkou. V rovnicích (3.1) a (3.2) lze vidět, jak se hodnoty jednotlivých uzlů tohoto stromu v případě takové hry vypočítají.



Obrázek 3.3: **Ukázka části stavového stromu při hře s šestihrannou kostkou** Zdroj: Studijní opora IZU viz [11]

Rovnice pro výpočet minimálního očekávaného ohodnocení pro uzel C_i má následující tvar, přičemž $P(hk)$ je pravděpodobnost výsledku hk, tedy hodu kostkou a min je funkce získávající minimální hodnotu z následujícího potomkovského uzlu.

$$expectimin\left(C_i\right) = \sum_k P\left(h_k\right) \cdot \min_j \left(D_{jk\ i}\right) \tag{3.1}$$

Podobně se postupuje při výpočtu $expectimax$ pro uzel D_j, akorát s maximální hodnotou.

$$expectimax\left(D_j\right) = \sum_k P\left(h_k\right) \cdot \max_l \left(C_{lk\ j}\right) \tag{3.2}$$

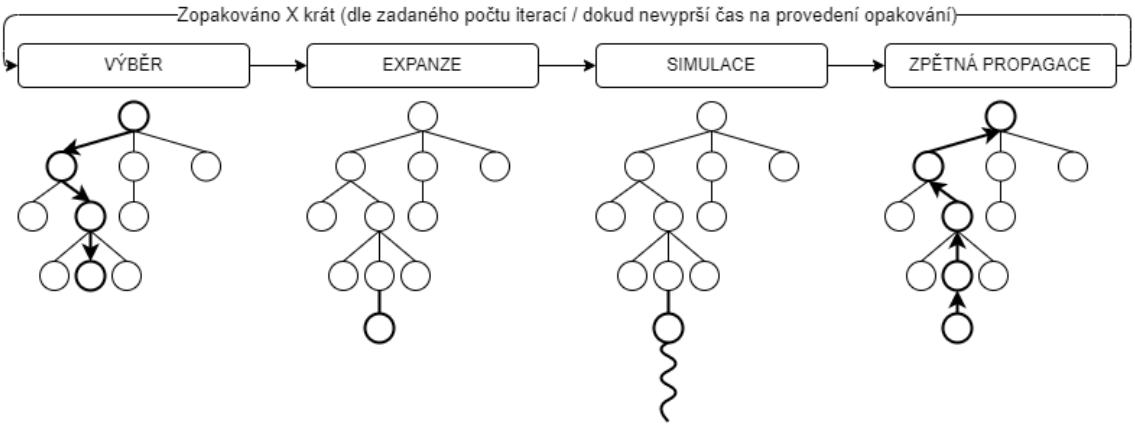
Podrobnější vysvětlení této metody a konkrétní příklad využití při použití v rámci deskové hry *Člověče nezlob se* jsou popsána ve Studijní opoře předmětu IZU [11].

3.4 Metoda Monte Carlo Tree Search

Tato metoda je zařazena již mezi ty více složitější. Metoda využívá jako předešlé metody také určitým způsobem stavový strom, ve kterém se generují jednotlivé tahy hráčů postupně za sebou i s tahy, které poté na ně reagují. V tomto případě je však využito možnosti simulování jednotlivých her začínajících z daných uzlů a vyhodnocování jejich ohodnocení na základě úspěchu či neúspěchu této simulace. Při využití této metody je tedy zahráno dle nastavení hned několik stovek až tisíců her, které vždy pro daný uzel, od kterého probíhala simulace vrací ohodnocení, které je propagováno také do rodičovských uzlů postupně až po kořen takového stavového stromu. Po provedení předem určeného počtu iterací je vybrán potomkovský uzel od toho kořenového, který získal nejvyšší ohodnocení a jeho tah je proveden. Tato metoda tak může při vhodném nastavení počátečních parametrů velmi dobře vybírat ty uzly, které byly užitečné (získaly dobré ohodnocení), ale zároveň i objevovat uzly nové. Více do hloubky je metoda popsána v následujících sekcích vycházejících z publikace [6], vědeckého článku popisujícího průzkum této metody viz [1] a článku [3].

Základní popis metody

Metoda se dělí na čtyři základní části, kterými jsou výběr, expanze, simulace a zpětná propagace. Tyto čtyři části jsou prováděny několikrát dle vhodně zvolené hodnoty. Obvykle alespoň 1000 iterací. V těchto iteracích jsou tyto části provedeny vždy ve stejném pořadí, jako byly vyjmenovány a generují tak postupně stavový strom i s upravováním ohodnocení jeho uzlů dle okolností. V rámci výběru proběhne určení uzlu, který se má dále vyhodnocovat, poté je provedena expanze tohoto vybraného uzlu (náhodně je získán jeden z možných tahů a je z něj vytvořen expandovaný uzel), z expandovaného uzlu je provedena náhodná simulace jedné hry až do konečného stavu, který určí ohodnocení, jak hra skončila (pro prvotní jednoduchost zda vyhrál hráč či oponent) a následně nastane zpětná propagace, která se zaslouží o ohodnocení uzlu, ze kterého byla hra simulována plus i dosazení tohoto ohodnocení do všech jeho předchůdců až ke kořenu stromu. Po provedení předem daného počtu iterací je ze všech takto vygenerovaných přímých potomkovských uzlů kořenu vybrán ten, který získal nejvyšší ohodnocení (v tomto příkladě nejvyšší počet výher) a daný tah, který tento uzel představoval, je vrácen jako nejvýhodnější k zahrání. Celý tento proces je znázorněn na obrázku 3.4.



Obrázek 3.4: **Monte Carlo Tree Search – diagram** Zdroj: Vlastní reinterpretace z [3]

Výběr

Výběr se zaměřuje na vhodné určení uzlu, který se má v danou chvíli v následujícím kroku expandovat. Probíhá od kořene stromu, kdy jsou postupně procházeny uzly, které mají nejvyšší ohodnocení dle funkce poskytující toto ohodnocení. Pokud je však při průchodu naraženo na uzel, jehož alespoň jeden potomkovský uzel nebyl doposud vybrán, je náhodně jeden takový potomkovský uzel vybrán a pokračuje se jeho expanzí.

Pomocí již zmíněné funkce pro ohodnocení je tedy zajištěna dostatečná míra objevování nových uzlů zároveň s použitím uzlů již objevených. Je tak při správné konfiguraci funkce zabráněno tzv. Exploration vs. Exploitation[1] problému. Při tomto problému nastává větší využití objevování nových uzlů na úkor využívání již objevených (a kladně ohodnocených) či naopak. V případě takového problému by poté z hlediska provedení mnoha iterací nemusel být vybrán vhodný následující tah.

Výběr zaručuje, že v případě nalezení velice výhodného tahu, který během simulací mnohokrát vedl k výhře, bude jeho uzel vybírán mnohem častěji, aby byl následně po provedení iterací tah zahrán. Díky zapojení objevování je zde však stále aplikován i výběr tahů nových, které jsou během iterací provedeny tak, aby byl i případný lepší tah objeven. Díky funkci pro ohodnocení jsou tedy tahy, které častěji při simulacích vedly k výhře lépe ohodnoceny. Proto jsou tyto tahy následně i vybrány, ale stále je zde zapojen i faktor výběru dalších tahů, které by mohly být přínosnými a lepšími.

Funkce pro ohodnocení využívá výpočet ohodnocení všech potomkovských uzlů, který je nazýván jako *Upper confidence bound*[2] *(UCB1)*, přičemž algoritmus kombinující tuto funkci pro ohodnocení *UCB1* a metodu Monte Carlo Tree Search je nazýván jako *Upper Confidence Bound applied to trees*[3] *(UCT)*. Pokud jsou tedy při výběru již všechny potomkovské uzly ve stavu, že byly již alespoň jednou navštíveny, je pro každý takový uzel tato funkce použita a uzel je tak ohodnocen. Následně je vybrán uzel s nejlepším ohodnocením a pokračuje se expanzí tohoto uzlu. Výpočet hodnoty pro UCB1 je vyjádřen následující rovnicí (3.3):

$$UCB1 = \frac{w_i}{s_i} + c\sqrt{\frac{\ln s_p}{s_i}} \tag{3.3}$$

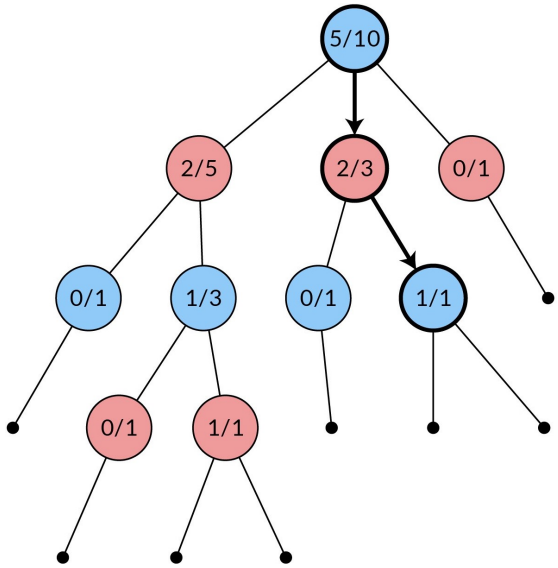
[1] Objevování (nových uzlů) vs. využívání (uzlů již objevených)
[2] V doslovném překladu jako *Horní hranice důvěry*
[3] V doslovném překladu jako *Horní hranice důvěry aplikována na stromy*

Jednotlivé proměnné a jejich popis:

- w_i – počet simulací proběhlých na tomto uzlu a vyhodnocených jakožto výhra pro hráče, kterého zastupuje tento uzel
- s_i – počet simulací proběhlých na tomto uzlu
- s_p – počet simulací proběhlých na rodičovském uzlu
- c – parametr pro míru ovlivnění objevování

Parametr c je konstanta, kterou se určuje míra zapojení objevování nových stavů vůči použití stavů již objevených. Díky ní je tedy zaručena dostatečná balance objevování a vyžití již objevených uzlů. Tato konstanta nemá předem stanovenou hodnotu, jelikož se odvíjí od ostatních okolností dané hry. Ve článku [3] je jako obvyklá hodnota této konstanty uvedena hodnota $\sqrt{2}$, ale oproti tomu v publikaci [6] je uvedeno nastavení na hodnotu $0,2$.

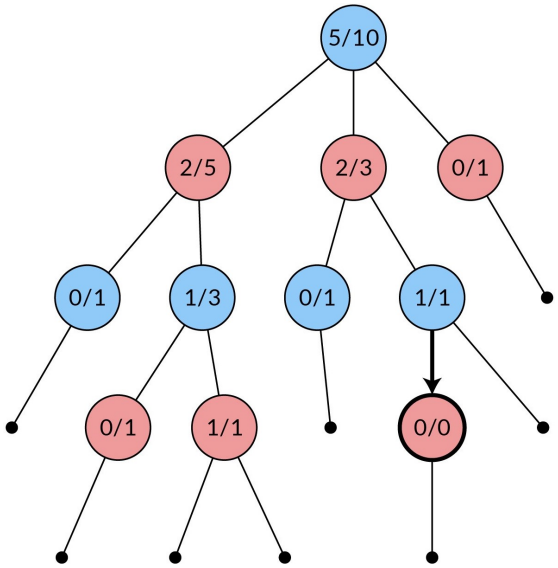
Na následujícím obrázku 3.5 je znázorněno provedení výběru na stromu, který má již několik uzlů vygenerovaných. Barevně je tedy vyznačen průchod uzly až k poslednímu vybranému, ze kterého bude v následujícím kroku probíhat expanze.



Obrázek 3.5: **Monte Carlo Tree Search – Výběr** Zdroj: [3]

Expanze

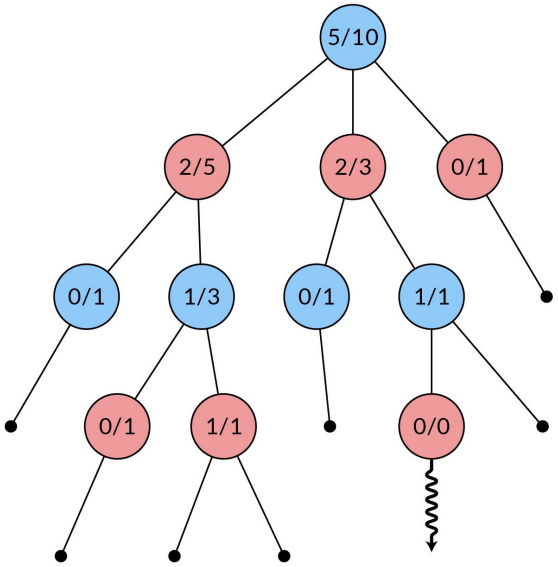
Jakmile je pomocí předešlého výběru nalezen a vybrán vhodný uzel, je tento uzel expandován. Tímto je myšleno nalezení nového následujícího možného tahu hráče, který je zrovna na řadě v dané úrovni zanoření ve stavovém stromě. Tah tohoto nově vygenerovaného uzlu je použit a přechází se do další fáze – Simulace. Na následujícím obrázku 3.6 je opět znázorněna ukázka expanze uzlu, který byl na předchozím obrázku 3.5 vybrán.



Obrázek 3.6: **Monte Carlo Tree Search – Expanze** Zdroj: [3]

Simulace

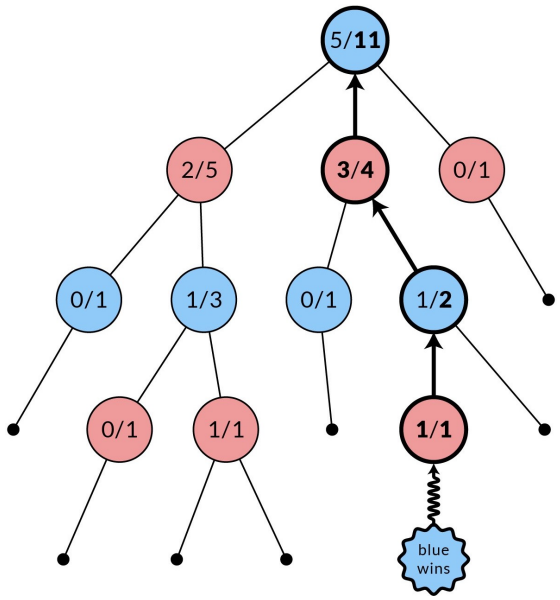
Simulace je provedena dále z již expandovaného uzlu. Náhodně dle pravidel hry je hra zahrána až do konce, kdy je díky tomu vyhodnoceno, který z hráčů vyhrál. Čím lépe jsou v této simulaci zavedeny veškerá pravidla či různé možné taktiky, tím lépe může konečné vyhodnocení být přínosné pro celkový výběr. Tudíž je výhodné klást důraz na logiku simulace a její provedení v závislosti na pravidlech hrané hry. Obrázek 3.7 demonstruje na uzlu, který byl v předchozím obrázku 3.6 expandován provedení jedné takové simulace.



Obrázek 3.7: **Monte Carlo Tree Search – Simulace** Zdroj: [3]

Zpětná propagace

Poslední částí jedné iterace metody Monte Carlo Tree Search je zpětná propagace, která má za úkol získaný výsledek uložit do dříve expandovaného uzlu a poté do všech jeho předků až ke kořeni. Tudíž vždy v závislosti na míře zanoření ve stavovém stromu je určeno, o uzel kterého hráče se jedná, a aktualizuje tak tedy jejich počet výher vůči provedeným simulacím. Tímto jsou tedy poté i ovlivněna ohodnocení uzlů při jejich výpočtu *UCB1* (3.3). Na následujícím obrázku 3.8 je tedy průchod při zpětné propagaci znázorněn.



Obrázek 3.8: **Monte Carlo Tree Search – Zpětná propagace** Zdroj: [3]

Kapitola 4

Aktuálně dosažené výsledky hraní Scotland Yard

Tato kapitola je zaměřena na popis a důkladné rozebrání již dosažených výsledků při pokusech o dosažení dostatečně inteligentního systému vhodného pro hraní hry Scotland Yard. Nejprve je v sekci 4.1 rozebrána implementace takového systému přímo pro celkovou hru Scotland Yard se zapojením metody Monte Carlo Tree Search vycházející z publikace [6]. V dalších sekcích 4.2 a 4.3 je vycházeno z předešlých bakalářských prací na stejné téma, jako má tato práce. Řešení těchto prací jsou tedy zaměřena již na zjednodušenou verzi hry, která byla popsána v sekci 2.3. Nejprve je tedy popsáno řešení metodou Alfa-Beta dle bakalářské práce Adriána Tulušáka [10] a hned po něm v následující sekci řešení metodou Monte Carlo Tree Search dle práce Michala Sovy [8]. V sekci 4.4 je poté popsán problém implementace s Monte Carlo Tree Search (4.3), který vedl k výrazně slabým výsledkům v porovnání této metody s metodou Alfa-Beta (4.2). Tato sekce tedy zhodnocuje předešlé výsledky implementací dostatečně autonomního systému a celkově je v ní popsán návrh možného úspěšnějšího řešení.

4.1 Implementace Monte Carlo Tree Search pro Scotland Yard

Využití této metody bylo v publikaci [6] použito přímo na celkovou hru Scotland Yard s plnými pravidly tak jako tomu je v popisu pravidel v sekci 2.2. Velice dobře ukazuje, že využití metody Monte Carlo Tree Search s důkladným provedením a obohacením o několik vylepšení v závislosti na hře dokáže vybudovat velice silný autonomně hrající systém. Jedním z těchto obohacení je velice dobrá technika, autory nazývána jakožto *Lokační kategorizace*, která bude více popsána níže. Stejně důležitou součástí je poté *Koaliční redukce*, která je také popsána ve své vlastní podsekci.

Lokační kategorizace

Tato technika je důležitým prvkem ve fázi získávání možných lokací Mistra X, kdy je zapojena určitá míra pravděpodobnosti míst, na kterých se bude spíše nacházet už jen například z důvodu toho, že by na jiných pozicích mohl být chycen. V takovém případě, pokud je logika Mistra X dostatečné funkční, dá se s vysokou pravděpodobností počítat, že se například nepohnul přesně na místo, kde může v následujícím tahu stoupnout některý z agentů

a tím vyhrát hru. Míra této pravděpodobnosti je tak dle uvedené zprávy určována dle tří hlavních vlastností:

- Minimální vzdálenost od nejbližšího agenta
- Průměrná vzdálenost všech agentů k možné lokaci
- Typy stanic propojených s danou možnou lokací

Díky těmto vlastnostem je provedeno rozdělení možných lokací do stanovených skupin tak, aby bylo docíleno lepšího vyhodnocování míst, kam jít, jelikož je tam větší pravděpodobnost dopadení Mistra X. Jelikož je tato práce více zaměřena jen na zjednodušenou verzi, důkladnější popis kategorizace pro plná pravidla hry Scotland Yard je popsán v již uvedeném zdroji [6].

Koaliční redukce

Díky této technice mohou agenti hrající v podstatě od samého začátku ve skupině lépe využívat možností skupinových pohybů tak, aby mohli zapojit snahu Mistra X obkličovat a určitým způsobem mu nadbíhat. Toto nadbíhání se jeví jako velice výhodné vzhledem k faktu, že by jinak bez zapojení této techniky mohl nastat řetězový přechod všech agentů za sebou následovně jeden za druhým, což je rozhodně nežádoucí pohyb. V rámci tedy již zmíněného zdroje je taky popsáno, že se v určitých chvílích můžou agenti chovat příliš ledabyle, pokud jeden z nich Mistra X v rámci simulací naleznul. V takové chvíli poté ostatní agenti mohou provést tah, který je naprosto náhodný a neefektivní.

Za účelem zamezení tohoto omezení je ve zdroji uvedeno zapojení různých velikostí ohodnocení stavů při nalezení Mistra X některým agentem tak, aby ostatní fungovali nezávisle na sobě. Tím pádem v některých částech mohou uvažovat jako skupina a Mistra X nahánět, ale někdy je využito i individuálního pohybu každý sám za sebe. Více je Koaliční redukce popsána opět v uvedeném zdroji [6].

Výsledky

Výsledné použití Monte Carlo Tree Search bylo porovnáváno s implementací na herní konzoli *Nintendo DS* kdy bylo dosaženo 33 výher z 50 her. Přičemž se hry skládaly s 25 her za stranu agentů a 25 za stranu Mistra X. Implementace autorů uvedené zprávy z těchto 33 výher vyhrála v 19-ti hrách za agenty a 14 her za Mistra X, čímž celkově potvrdila svou dominanci a silnější zapojení umělé inteligence. Autoři ještě také ke konci dodali, že tento experiment byl jediným možným, protože nebyl nalezen další vhodný soupeř, který by byl dostatečně silný na porovnání či byli soupeři nepraktičtí pro vytvoření dostatečného počtu pokusů.

4.2 Zjednodušená verze s metodou Alfa-Beta prořezávání

Na zjednodušené verzi byla poprvé využita implementace s metodou Alfa-Beta. Obsah této sekce je pořízen na základě bakalářské práce A. Tulušáka [10]. Při vytvoření této verze systému pro hraní hry byly nakonec využity čtyři heuristiky. Jedna byla základní, která byla vytvořena na počátku pro spuštění her a provedení experimentů s lidskými hráči a poté do heuristiky této verze byl zaváděny změny na základě zjištěných nedostatků.

V popisu prvotní verze autor uvedl, že byla podrobena testování se čtyřmi různými lidskými hráči. Jejich cílem bylo jakožto Mistr X unikat agentům, kteří byli ovládání již zmíněnou umělou inteligencí. Dle popisu v dané práci lidští hráči během prvních 10-20 her prohrávali, ale jelikož se po průběhu těchto pár her zvládli adaptovat na styl hry agentů, dokázali poté tahy umělé inteligence odhadovat. V takovém případě se poté již první základní verze mohla jevit jakožto slabá. Hlavní nevýhodou bylo, že se tahy agentů jevily, jako by následovaly Mistra X, ale v konečném důsledku jej nezachytili. V dalších verzích tedy byla heuristika upravována dle daných zjištěných problémů. Pro přehlednost jsou jednotlivé verze a jejich úpravy popsány níže postupně.

Verze 1

Toto tedy byla základní verze, jejíž heuristická funkce prováděla ohodnocení za všechny agenty najednou. Občas se stalo, že se agenti chovali nelogicky, kdy přecházeli z místa tam a zpět, a hlavně lidský hráč po nějakém čase dokázal dostatečně odhadnout chování agentů tak, aby zvládal dle zdroje v 90 % vyhrát nad umělou inteligencí agentů.

Verze 2

V rámci této verze nastalo několik změn heuristické funkce, kdy je tato funkce počítána pro každého agenta zvlášť. V potaz je taky brána vzdálenost jednotlivých agentů od Mistra X, kdy ten, co je blíže, více ovlivňuje hodnotu funkce. Toto zapříčinilo lepší pohyb agentů tak, aby se chovali nezávisle na sobě. Dle uvedených údajů bylo ve výsledku toto vylepšení jen mírné v utkání s lidským hráčem, neboť si opět dokázal po pár hrách uvědomovat slabiny algoritmu, a tak odhadnout následující chování agentů.

Verze 3 a 4

Ve verzi 3 bylo uvedeno jen upravení výpočtu hodnot heuristických funkcí, na tuto úpravu se lze blíže podívat v již uvedené bakalářské práci. Verze 4 oproti tomu zavedla důležitou změnu, kdy upravila hlavně smýšlení, které bylo použito ve verzi 2. Tímto je myšlena úprava, kdy bylo chybné zavedení ohodnocení pouze agenta, co byl blíže Mistru X. Druhý agent tak ve výsledku prováděl pouze náhodný tah a nechoval se tak, aby byl schopen pomoci druhému agentovi s dopadením Mistra X.

Výsledky

Jelikož se jevila nakonec čtvrtá verze jakožto tou nejsilnější, zavedl autor tuto verzi tak, aby mohla jí produkovaná umělá inteligence hrát i za Mistra X. V tabulce 4.1 jsou zobrazeny jednotlivá procenta úspěšnosti verzí aplikovaných na agenty proti čtvrté verzi aplikované na Mistrovi X při zahrání sto her.

Verze agentů	Verze 1	Verze 2	Verze 3	Verze 4
Úspěšnost	92 %	65 %	74 %	89 %

Tabulka 4.1: Úspěšnost jednotlivých verzí agentů proti čtvrté verzi u Mistra X ve 100 hrách

4.3 Zjednodušená verze s metodou Monte Carlo Tree Search

V návaznosti na předešlou implementaci s využitím metody Alfa-Beta byla vytvořena verze s metodou Monte Carlo Tree Search. Tato sekce popisuje využití této metody a vychází z bakalářské práce M. Sovy viz [8]. Tato práce měla již výhodu možnosti porovnání dvou různých návrhů umělé inteligence pro hraní proti sobě. Bohužel však dle výsledků experimentů byla tato verze proti té předešlé slabá a ve většině případů prohrála. V následující podsekci je popsána forma implementace a její výhody či nevýhody.

Implementace

Autor již zmíněné práce se rozhodl, že se při každém vyhodnocování následného tahu bude tvořit stavový strom jako tomu je popsáno v sekci 3.4. V každém uzlu tedy byl obsažen vždy daný stav, jak to vypadá na herní desce, aktuální počet kol a celkový počet kol, počet výher pro daný uzel, a nakonec také ohodnocení tohoto uzlu. Vyhodnocování uzlů (čili výběr) probíhalo v jednotlivých modifikacích autorova algoritmu odlišně. Respektive nejprve bylo vyhodnocování čistě na výhry, kdy výpočet UCB1[1] probíhal pouze v klasickém využití s výhrami. Při dalších úpravách však tuto míru ohodnocení výrazně ovlivnily změny při simulaci a zpětné propagaci, které se týkaly zapojení systému odměn. V situaci, když byl agent blíže Mistru X, získal lepší ohodnocení. To samé platilo pro simulaci, která skončila v dřívějším kole. V tu chvíli dostal uzel s tímto tahem výrazně lepší ohodnocení. Tyto míry úprav provedly rozdíl v procentech úspěšnosti v řádu jednotek procent.

Možné chyby implementace

Prvotní návrh systému popsaný v uvedeném zdroji se jeví správně, jelikož je zapojen vyhledávací strom a správně zvolený průchod a všechny fáze použité metody. Jeho implementace však ukázala, že by někde mohla být nejspíše chyba, jelikož využití této metody vůči metodě předchozí má pro agenty úspěšnost dle zdroje pouze 22,4 %. Základní strom je v literatuře popisován jakožto strom, který jednotlivé vrstvy zanoření rozděluje na tah Mistra X v jedné vrstvě a tah agentů ve vrstvě druhé. Autor této práce se však rozhodl celkové kolo (tudíž všechny tahy jak agentů, tak Mistra X) zapojit do jednoho uzlu najednou. Vyhodnocení se poté děje na fázích kol, kdy navíc nemusí být dostatečně názorné najít, zda není chyba v pořadí tahů agentů či Mistra X v rámci jednoho kola, neboť v takovém případě, když je Mistr X zrovna vidět hned vedle agenta, měl by jej agent okamžitě chytit, protože nejdříve v daném kole táhne Mistr X, poté se zobrazí jeho pozice (pokud se jedná o kolo, kdy nemá zůstat skrytý) a poté táhnou teprve agenti.

Dalším důležitým faktorem, který mohl implementaci vést k tak nízké úspěšnosti může být nevhodné zvolení ohodnocení. V takovém případě by mohlo nastat, že se některý stav, který může být naprosto výrazně výherní, nemusel v rámci provedení simulací projevit, jelikož je ohodnocen méně. Tento fakt však není z popisu dané implementace patrný a je pouze připomenut jako možná chyba.

Výsledky

Z hlediska výsledků, jak již bylo zmíněno, nebyla tato verze vůči verzi s Alfa-Betou úspěšná. V 80 % spíše zvítězila Alfa-Beta. V takovém případě se sice nejspíše povedlo nějakou formou implementovat danou metodu, ale provedení implementace tohoto systému někde selhalo.

[1] *Horní hranice důvěry* viz rovnice 3.3

Je to však ve výsledku vnímáno, že je možné algoritmus a jeho provedení ještě vylepšit, a tím i posunout danou verzi k lepší úspěšnosti. Možnou cestou by mohlo být upravení ohodnocení stavů a zlepšení výběru v rámci tohoto ohodnocování.

4.4 Zhodnocení a návrh řešení

Vzhledem k zadání této práce byly již provedeny dva pokusy o realizaci systému, který by mohl hrát hru Scotland Yard autonomně. Oba dva pokusy se zaměřovaly na vhodnou implementaci takového systému pro hraní zjednodušené verze této hry s myšlenkou, že poté bude do této verze přidáváno několik jejich dalších vlastností tak, aby se v konečném důsledku dosáhlo verze plné. Volba takového postupu řešení nabízí rozdělení na podproblémy, které lze postupným přidáváním vylepšení eliminovat. Obě dvě řešení na zjednodušené verzi hry jsou tomu důkazem, jelikož implementovaly i možnost poté hrací mapu zvětšovat, přidávat počet agentů a celkově škálovat pravidla hry tak, aby mohl být implementovaný algoritmus testován při složitějších případech.

V sekci 4.1 bylo dle zdroje nastíněno, že využití metody Monte Carlo Tree Search je vhodnou volbou pro vytvoření systému pro hraní verze plné. Zároveň však bylo dokázáno, že mnoho takových implementací neexistuje, jelikož sám zdroj uvádí, že bylo nalezeno prozatím pouze pár důstojných soupeřů, z toho jen jeden mohl být využit k poměření sil. I v takovém případě pro plnou verzi bylo jednoznačně vhodno použít dostatečně vysoký výpočetní výkon, aby zvládnul pracovat na celé mapě o 200 polích, které navíc spojují dohromady tři přes sebe překryté mapy různých dopravních prostředků.

Sekce 4.2 a 4.3 popisovaly, jak byly jednotlivé návrhy metod pro hraní zjednodušené verze úspěšné. V konečném důsledku se jeví jako úspěšná hlavně verze s využitím metody Alfa-Beta, jelikož dosáhla proti následující verzi s Monte Carlo Tree Search mnohem lepších výsledků. Při jejich porovnání jí nebyla implementace této metody schopná konkurovat. Jelikož bylo využití metody Monte Carlo Tree Search vnímáno jako velice vhodné již v sekci 4.1 při využití na plná pravidla hry, je velice pravděpodobné, že by mělo využití této metody na zjednodušené verzi být alespoň vyrovnané, tudíž úspěšnost agentů, které by ovládal algoritmus Monte Carlo Tree Search by měla být proti Mistru X (ovládaném Alfa-Betou) dle předpokladů alespoň 50 %.

Nutno podotknout, že implementace v sekci 4.3 by mohla obsahovat několik chyb, které mohly její úspěšnost velice ovlivnit. Pokud by se opravdu jednalo o chyby implementace a byly by dostatečně opraveny, mělo by se to projevit na úspěšnosti při postavení proti předchozí metodě. Následující tabulka 4.2 poskytuje zobrazení souhrnu úspěšností, které byly získány na základě bakalářské práce M. Sovy [8], kde k porovnání systémů s využitím těchto dvou metod došlo.

Mistr X	Agenti	Úspěšnost agentů
Alfa-Beta	Monte Carlo Tree Search	20,8 %
Monte Carlo Tree Search	Alfa-Beta	91 %

Tabulka 4.2: Procentuální míra úspěšnosti agentů při použití metod Monte Carlo Tree Search (Michal Sova 2021 [8]) a Alfa-Beta (Adrián Tulušák 2020 [10]) při počtu 1000 provedených her

Návrh řešení

Dle získaných hodnot úspěšnosti z pohledu jak strany hrající za agenty, tak strany Mistra X je patrné, že by bylo dobré pokusit se o napravení a zlepšení implementace systému hrajícího s využitím metody Monte Carlo Tree Search. Toto zapojení metody by mělo být buď opraveno nebo nejlépe naprogramováno úplně na novo, čímž by se mohla naskytnout možnost porovnat tři různé druhy implementace. Již v předpokladu využití zapojení stavového stromu, kdy bude tento strom rozdělen v jednotlivých vrstvách zanoření na dvě strany hráčů a nebude každá vrstva sloužit pro jedno celé kolo hry, by se mohla taková implementace velice odlišit v míře úspěšnosti. I tak by bylo tedy výhodné implementaci zapojit samotnou zvlášť, protože už v takto základním bodu, jako je stavový strom, se budou tyto dvě implementace metody Monte Carlo Tree Search dostatečně lišit.

Vzhledem k získaným informacím uvedeným v sekci 4.1 by bylo určitě pro vyšší úspěšnost algoritmu výhodné zapojit rozlišnosti v ohodnocování tahů dle počtu kol, které byly k dosažení Mistra X zahrány a stejně tak dle vzdálenosti Mistra X od agentů. V prvotní fázi by však mohlo být vysokým přínosem zjistit, jak by si vedl algoritmus tak, jako je popsaný základně s pouhým využitím počtu výher daných uzlů při provedených simulacích. Důkladnější popis metody, ze kterého vychází tento odstavec se vyskytuje viz 3.4.

Řešením by tedy mohla být implementace algoritmu dle základního popisu metody Monte Carlo Tree Search, které by bylo odkázáno v prvotní fázi pouze na výhry a prohry, přičemž by se poté dalo toto řešení rozšiřovat o prvky rozmanitějšího ohodnocování stavů. Pokud by se tato implementace v porovnání s předešlými ukázala jako nejsilnější, mohlo by se v rámci řešení přejít i k rozšiřování hracího pole, přidávání agentů a jiným možným úpravám, kterým byla jen velice mírně vystavena i předešlá řešení.

Kapitola 5

Implementace algoritmů řízení pro hraní hry Scotland Yard

Následující kapitola popisuje celkovou vlastní implementaci systému pro autonomní hraní zjednodušené verze hry Scotland Yard. V prvních dvou sekcích je popsán podrobně celkový koncept aplikace a systému tak, aby bylo jasno, za jakým účelem jsou ve výsledném programu obsaženy dané vlastnosti, které pomáhají k následnému zhodnocení. Toto zhodnocení bylo poté dosaženo díky provedení experimentů a celkově je i s nimi popsáno v sekci 5.3. Díky experimentům se povedlo nalézt mnoho dalších vylepšení a odstranit různé chyby, které v prvotním návrhu nebyly dobře identifikovatelné. Souhrn těchto úprav i s vylepšeními je detailně rozebrán jak u popisů jednotlivých experimentů, tak v sekci 5.4, po které poté následuje celkové zhodnocení celé implementace a porovnání výsledků. V této poslední sekci jsou také projednány některé nedostatky a návrh snížení jejich dopadu na celkový systém hrající autonomně.

Jelikož již bylo v rámci řešení vycházeno z předešlých dvou zmíněných bakalářských prací [8] a [10] byla jednotlivá řešení obou kolegů použita jakožto základ, ze kterého se dalo určitým způsobem vycházet. Nakonec však byly obě řešení použity jakožto jednotlivé moduly, kdy každý provádí tahy dle své implementace s určenou metodou a poté je propaguje zpět do systému, který tah použije, zobrazí a vyzve modul, který ovládá oponenta k tahu.

5.1 Zaměření prvotních úprav implementace

Jelikož bylo v rámci realizace implementace vycházeno z již dvou vytvořených, bylo nejprve nutno se pořádně do hloubky zorientovat ve fungování aplikace jako takové. Nakonec jsem se rozhodl původní soubory aplikace rozdělit do dvou modulů, každý pro jednu implementaci a nad nimi vytvořit své řešení hlavní funkce, které dle zadaných argumentů využívalo některý z modulů, či přímo mou implementaci. Celkový rozpis rozložení souborů programu je popsán v příloze A. Po prvotním rozložení tedy bylo možno využívat tří verzí systému, který určoval, jak se budou Mistr X či agenti pohybovat v hracím poli. Díky vhodnému nastavení herního prostředí bylo tak možno vždy jednoduše předat jednomu z modulů aktuální situaci ve hře a ten poté vrátil následující tah, který byl proveden a vyhodnocen vůči základním pravidlům (zda byl či nebyl Mistr X chycen). Jednoduše tak poté šlo hru hrát autonomně jedna verze proti některé ze dvou ostatních a mohlo být provedeno několik her za sebou tak, aby se poté dala určit míra úspěšnosti dané verze systému.

Již zmíněnými třemi verzemi byly:

1. Systém s využitím metody Alfa-Beta (Implementace Adriána Tulušáka z roku 2020 [10])
2. Systém s využitím metody Monte Carlo Tree Search (Implementace Michala Sovy z roku 2021 [8])
3. Systém s využitím metody Monte Carlo Tree Search (Vlastní implementace této práce)

Ačkoliv jsou ve verzích 2 a 3 využity stejné metody, jejich implementace je provedena velice odlišně. Důvodem pro provedení implementace s využitím stejné metody byla hlavně velice nízká úspěšnost verze 2 proti verzi 1 s využitím metody Alfa-Beta. V následující podsekci jsou popsány zásadní změny verzí s využitím metody Monte Carlo Tree Search, kdy bylo jejich provedení v konečném důsledku úplně odlišné.

Odlišnosti ve verzích s využitím Monte Carlo Tree Search

Autor verze 2 v případě tvorby vyhledávacího stromu má v každé hloubce tohoto stromu rovnou celkové jedno kolo hry. Je tak tedy učiněno, že tah je proveden jak Mistrem X, tak všemi agenty a poté je z tohoto uzlu provedena simulace. Problém ovšem nastává hned při začátku generování tohoto stromu. Jelikož kořen v případě skrytého Mistra X byl zkrátka vytvořen s náhodně vybraným možným pohybem Mistra X. Nebyl tak vůbec brán v potaz jakýkoliv z dalších možných tahů. V danou chvíli tedy již takto bylo vzato v potaz při expandování i simulacích, že se zkrátka Mistr X pohnul na náhodně vybranou pozici a poté se vytvořil z tohoto kořene strom.

Vlastní implementace (verze 3) tedy již v tomto ohledu měla za cíl snažit se zaměřit všechny možné pozice Mistra X při jeho skrytí. V takovém případě bylo nutno všechna tato místa získat již při tvoření kořenu. Nejprve jsem měl za cíl vygenerovat tato místa každé zvlášť jakožto několik možných kořenů a z nich poté vybrat ten, který byl nejlepší. Nakonec však bylo provedeno nespočet úprav a experimentů s nimi. Celkový popis je tedy více rozepsán v následujících dvou sekcích.

Další důležitou odlišností je vyhodnocení výběru uzlů. Autor předešlé verze (2) zmínil, že rovnou využil implementace výběru uzlů dle rovnice se započítáním ohodnocení daných stavů po simulaci. Toto ohodnocení nestavěl tedy na již zmíněné rovnici 3.3, ale rovnou použil složitější tvar, kdy místo uvažování poměru výher vůči počtu průchodů tímto uzlem nahradil tuto část ($\frac{w_i}{s_i}$) za nepřímo pojmenovanou hodnotu (v implementaci figuruje slovo *value*). Tuto hodnotu poté používá jakožto zmíněnou náhradu při výpočtu UCB1 3.3 a dle výsledku je prováděn výběr uzlů.

Sám jsem se ve své implementaci zamyslel nad myšlenkou takového využití v některých případech, kdy je vhodné zvýšit míru použití některého z tahů ve stavovém stromě. Nakonec se však i díky experimentům ukázalo, že poměr výher s počtem průchodů může být bohatě dostačující vzhledem k požadavkům této hry. O to více bylo však nutnější se zaměřit na správné expandování jen opravdu možných stavů a správnou simulaci.

5.2 Popis implementace

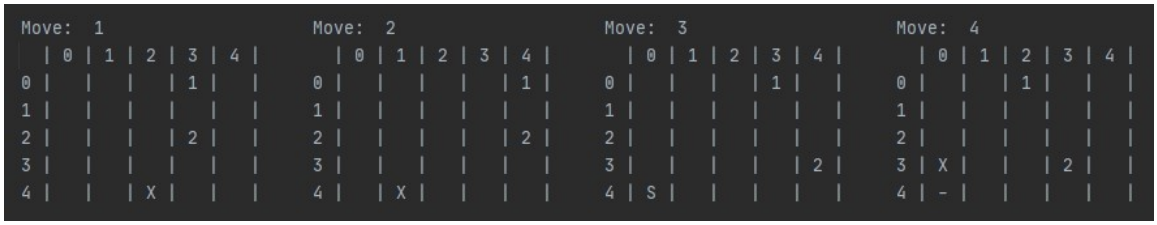
Implementace byla provedena na základě již dvou předešlých implementací tudíž byla vytvořena v jazyce Python, který byl použit i dříve. Tyto dvě předešlé verze jsou využity jako moduly, které slouží pro získávání nových tahů v případě jejich využití. Samotná vlastní

implementace poté má hlavní funkci *main*, která rozjede simulování hraní her dle zadaných argumentů. V rámci souboru obsahujícího tuto funkci jsou také zapojeny funkce pro generování určitých statistických údajů a funkce, které volají jednotlivé moduly při snaze získat následující tah jedné ze stran.

Vzhledem k vhodnému získání všech argumentů byla využita knihovna *argparse*, kdy je daný analyzátor argumentů nastaven v souboru *argparser.py*. V souboru *mc_environment.py* je provedena veškerá práce s herním prostředím jako takovým. Tento soubor byl převzat z předešlé implementace kolegů, ale v některých částech upraven (úpravy jsou v souboru vhodně označeny a okomentovány).

V poslední řadě je zde hlavní soubor, kterým je vlastní implementace metody Monte Carlo Tree Search a všech nutných prostředků k jejímu provedení. Celý tento systém je poté implementován v souboru *MG_monte_carlo.py*. Zde je uvedena jak třída *Node*, která vyjadřuje jeden z uzlů daného stavového stromu, tak její veškeré metody. V poslední řadě jsou poté v tomto souboru uvedeny i vlastní funkce, které slouží ke spuštění chodu daného systému při zjišťování následujícího tahu a k jednotlivým krokům dané metody (výběr, expanze, simulace a zpětná propagace).

Herní prostředí při výpisu je znázorněno na následujícím obrázku 5.1, kde lze vidět, jak jsou jednotlivé stavy hry vypisovány do konzole. Agenti jsou označeni čísly **1** a **2**, přičemž v případě více agentů toto číslování jednoduše pokračuje. Mistr X je v případě odhalení označen písmenem **S** a v případě skrytí označen písmenem **X** a jeho historická pozice odhalení jako **-**.



Obrázek 5.1: **Ukázka stavu herního pole zobrazeného v konzoli** Zdroj: Vlastní snímek

Takto je tedy hra vypisována, pokud si to uživatel přeje vidět, či pokud chce využít hraní za agenty či Mistra X pomocí zadávání směrů tahu z klávesnice. Jelikož jsem v rámci experimentů chtěl i v některých chvílích pouze spustit simulaci mnoho her za sebou a nepotřeboval jsem při tom mít zobrazen průběh hry, byl přidán argument *--no-print*, který jednoduše toto vypisování vypne a pouze provádí hry na pozadí. S využitím tohoto argumentu spolu s argumentem *-f*, který poté jen uvádí výsledky jednotlivých her do souboru bylo možno provést mnoho her pro experimenty za kratší čas.

Vlastní základní implementace Monte Carlo Tree Search

V předešlé části sekce bylo popsáno, jak celý koncept aplikace funguje. V této části bych se rád zaměřil na popis vlastní implementace metody Monte Carlo Tree Search tak, jak vypadala po prvotním vytvoření, kdy sloužila jakožto odrazový můstek k provedení prvních experimentů. Z výsledků těchto experimentů pak vznikaly návrhy na úpravy implementace tak, aby dosahovala vlastní metoda co nejvyšší úspěšnosti z pohledu hry agentů proti doposud nejsilnější metodě Alfa-Beta v roli Mistra X.

Hlavními funkcemi v souboru vlastní implementace *monte_carlo.py* jsou funkce *mc_agents_move* a *mc_mrX_move*. Tyto funkce jsou volány dle požadovaného vyhodno-

cení tahu. Jak už název napovídá, buď pro Mistra X nebo pro agenty. Následný popis bude tedy veden pro agenty, přičemž pro Mistra X se takřka shoduje až na opačné vyhodnocování výher a proher. U Mistra X navíc není zapojena neurčitost, jelikož pozice agentů neustále vidí, tím pádem se snaží jen vzhledem k nastavení metody neustále unikat.
Tah agentů je tedy po zavolání příslušné funkce vyhodnocován v prvotní implementaci následovně:

1. Pokud je tah prvním či druhým, vrať náhodný pohyb agentů, jelikož zatím nemáme žádnou informaci, kde by se mohl Mistr X vyskytovat.

2. Vytvoření kořenového uzlu stavového stromu, ve kterém je uložena aktuální pozice Mistra X (v případě, že je viditelný, jinak jeho poslední pozice, kdy byl odhalený), aktuální pozice agentů, číslo aktuálního kola (tahu) a počet navštívení tohoto uzlu. V tomto případě bude číslo navštívení nastaveno na 1, jelikož jsme v kořenovém uzlu, který je již vytvořen a má být dle metody rovnou v následující akci vybrán a také expandován.

3. V tomto kroku probíhá dle nastaveného argumentu času pro iterace celého algoritmu největší počet iterací, který se povede za tuto dobu stihnout. Tudíž je opakování rozděleno na tyto kroky:

   (a) Nad kořenem zavolej rekurzivní funkci pro výběr a expandování vybraného uzlu (jak je tato funkce implementována je více popsáno níže).

   (b) Proveď z vybraného a vráceného uzlu simulaci jedné hry (opět simulace a její konkrétně zvolená implementace rozepsána níže).

   (c) Proveď zpětnou propagaci výsledku simulace (tudíž výhru či prohru) až po kořen stromu.

4. V případě zvolení argumentu pro tisknutí rozhodovacího stromu, vytiskni do konzole tento strom po určený level zanoření.

5. Vyber u kořene nejlepší následující uzel, jehož tah poté vrať jakožto tah, který se zahraje.

Vzhledem k výběru bylo zvoleno tedy již zmíněné rovnice 3.3, kdy vždy na volaném uzlu je zkontrolováno, zda byl již navštíven. Pokud nebyl, tak je vrácen přímo sám, aby na něm byla provedena simulace. V případě že byl navštíven, zatím nemá potomkovské uzly a jeho číslo aktuálního kola není větší než maximální nastavený počet kol (v základním nastavení 15), je tento stav expandován. Expandování probíhá tak, že rovnou pod tento uzel jsou přidány všechny možné potomkovské uzly. Z nich je následně tedy vrácen první vytvořený uzel expanzí a s tím se opět postupuje do simulace. V poslední možné situaci, kdy byly všechny potomkovské uzly alespoň jednou navštíveny, je konečně využito dané rovnice, která určí uzel, ze kterého je opět zavolán výběr. Tím je dosaženo rekurze v dané funkci pro výběr.

Funkce pro expandování je jednoduše vytvořena tak, že pokaždé, když má z daného uzlu vytvořit všechny možné potomkovské uzly, zavolá pomocné funkce pro generování tahů Mistra X a agentů v závislosti na tom, zda rodičovský uzel obsahuje tah jedné či druhé strany. Tyto pomocné funkce vrátí všechny možnosti pozic, na kterých se mohou jednotlivé postavy vyskytnout a vytvoří všechny možné kombinace těchto tahů. V případě rodičovského uzlu s tahem Mistra X jsou tak tomuto uzlu přiřazeny všechny potomkovské

uzly se všemi kombinacemi tahů agentů. Na následujícím obrázku 5.2 je takto vygenerovaný strom s výpisem vygenerovaných přímých potomků z kořene pro lepší představu znázorněn.

```
Move:  3
|__ Move: 3 Mr.X: [0, 1]   Agents: [[4, 2], [3, 4]] wins:  1886 visited: 1602 UCB:  0.0
  |__ Move: 3 Mr.X: [0, 1]   Agents: [[3, 2], [2, 4]] wins:  97   visited: 157  UCB:  1.7017632971516794
  |__ Move: 3 Mr.X: [0, 1]   Agents: [[3, 2], [3, 3]] wins:  146  visited: 199  UCB:  1.6964417456104648
  |__ Move: 3 Mr.X: [0, 1]   Agents: [[4, 1], [2, 4]] wins:  148  visited: 200  UCB:  1.7003634541800252
  |__ Move: 3 Mr.X: [0, 1]   Agents: [[4, 1], [3, 3]] wins:  1495 visited: 1045 UCB:  1.8427534578275173
Chosen: [[4, 1], [3, 3]]
  | 0 | 1 | 2 | 3 | 4 |
0 |   | S |   |   |   |
1 |   |   |   |   |   |
2 |   |   |   |   |   |
3 |   |   |   | 2 |   |
4 |   | 1 |   |   |   |
```

Obrázek 5.2: **Ukázka expandovaného kořene přímo z implementace** Zdroj: Vlastní snímek

Funkce simulace je nejspíše tou nejdůležitější při provedení této počáteční verze metody, jelikož na ní stojí, zda správně simuluje jednotlivé hry tak, aby byl poté několikrát vybrán a vyzkoušen správný uzel, který vede k nejlepšímu možnému tahu agentů (případně Mistra X). Funkce vrací pouze pravdivostní hodnotu dle výhry či prohry. V prvotní implementaci byla navržena pouze tak, že náhodně generuje nové uzly s tahy jak agentů, tak Mistra X dle toho, jakým typem byl uzel rodičovský. Pokaždé pak pouze vyhodnotí, zda nebyl Mistr X obklíčen, či neproběhlo postavení agentů na pole, kde zrovna Mistr X stojí. V takovém případě funkce vracela pravdu. Pokud však proběhla simulace až do čísla kola, které bylo určeno jakožto poslední, byla vrácena lež, jelikož agenti prohráli. Opačné pravdivostní hodnoty vracela funkce v případě provádění simulace v rámci tahu Mistra X. Tento způsob implementace jsem jednoznačně nepovažoval za nejsilnější, ale jak se již při prvních experimentech ukázalo, tak i tak byl v konečném důsledku překvapivě hodně úspěšným.

5.3 Popis experimentů

Tato sekce je zaměřena na popis všech experimentů, které byly postupně provedeny. Vždy v rámci nich proběhla aktualizace systému tak, aby byl schopný dosahovat vyšší úspěšnosti. Tato úspěšnost se celkově získávala z pohledu agentů ovládaných vlastní implementací systému s metodou Monte Carlo Tree Search, kdy byli agenti postaveni proti doposud nejlepší implementaci systému pro Mistra X, a to verzi s Alfa-Betou.

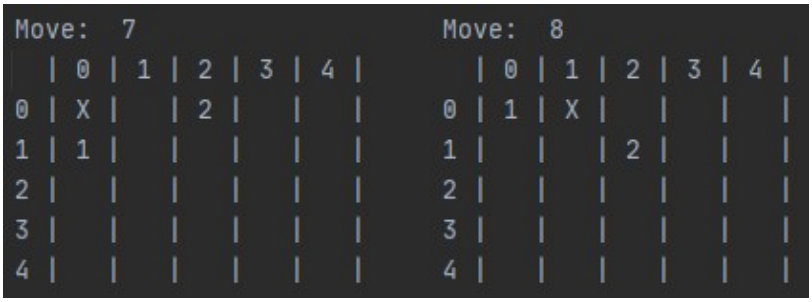
Základní náležitosti

Jeden experiment si vždy zakládal na provedení 10 000 her zahraných po sobě, kdy se tímto získala pomocí počtu výher agentů procentuální úspěšnost. Pro agenty byl zvolen čas na provedení iterací při rozhodování následujícího tahu na 1 sekundu. Bylo tak učiněno z důvodu, že předešlá implementace Monte Carlo Tree Search od kolegy Sovy měla také zvolen tento čas na vyhodnocení následujícího tahu. V takovém případě poté bylo velice dobře porovnatelné, která implementace je úspěšnější vzhledem ke stejným základním podmínkám. Vzhledem ke konstantě c (viz rovnice 3.3), díky které je zajištěna dostatečná míra objevování nových uzlů zároveň s použitím uzlů již objevených, se obě hodnoty lišily, jelikož

kolega upravoval danou rovnici dle jeho různých experimentů. Já jsem se rozhodl využít tuto rovnici tak jak je popsána v 3.3 a raději upravovat části implementace. Bylo to učiněno z toho důvodu, že již při prvním experimentu s mou implementací již popsanou v sekci 5.2 bylo dosaženo lepší úspěšnosti agentů, což je také lépe popsáno v následující podsekci.

Experiment číslo 1

Tento experiment byl proveden se základní implementací systému tak, jako byla popsána v sekci 5.2. Při hraní jednotlivých her byl prováděn výpis tahů tak, jak byly zahrány. Díky tomu bylo možno vidět, jak se v jednotlivých situacích agenti chovají. Na první pohled se jevili agenti podstatně inteligentně, jelikož již v několika tazích, měli možnost Mistra X obklíčit. Tím je myšleno že se dostali do blízkosti jednoho z rohů tak, že po následujícím tahu byla vysoká šance na chycení Mistra X. Ostatně je tento typ postavení v jednom kole zobrazen na obrázku 5.3.

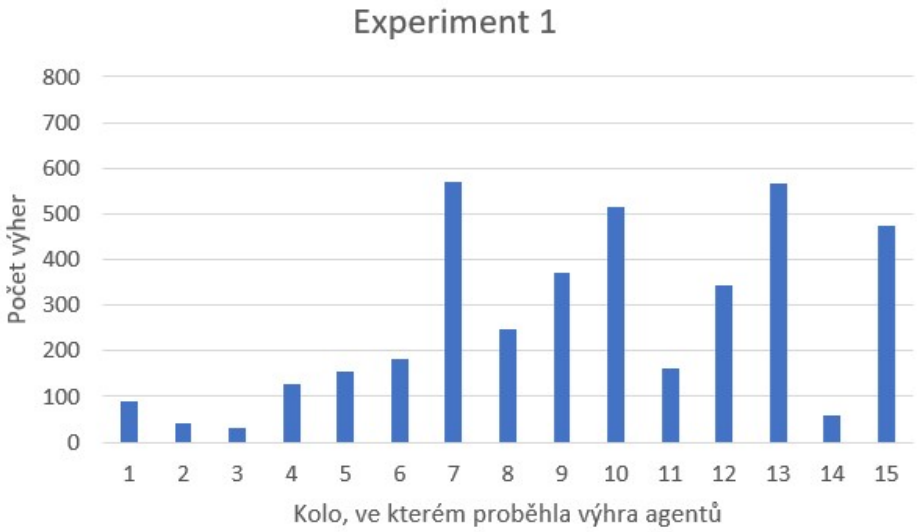


Obrázek 5.3: **Ukázka postavení pro možné obklíčení a následný únik při hře** Zdroj: Vlastní snímek

Problémem se však v této situaci stala možnost Mistra X být skryt. Jelikož již agenti mají několik možností, kde by mohl Mistr X stát a aktuální verze implementace nijak hlouběji tuto situaci neřeší, rozhodnou se v tomto stavu spíše pro různé pohyby, které ve výsledku jedno z míst Mistru X uvolní. Vzhledem ke generovanému rozhodovacímu stromu totiž vznikne několik uzlů s různými místy, kde zrovna může Mistr X stát. V takovém případě se při simulaci vyhodnotily tyto stavy, kde zároveň byl chycen lépe. Bohužel však pouze vzhledem k jednomu z agentů. V danou chvíli tak jeden agent sice stoupl na pole, kde Mistru X zablokoval cestu, ale druhý agent občas šel naprosto jinam. V takovém případě bylo tedy nutné začít uvažovat, jak zvýšit v rámci simulace úspěšnost tahů agentů, kde budou oba stát na místech možného výskytu Mistra X (potažmo místech, kde by mu zablokovali následující tah).

Bylo tedy provedeno 10 000 her a dosaženo počtu 3 932 výher pro agenty, tudíž úspěšnosti 39, 32 %. Ohledně časové náročnosti toto zahrání 10 000 her trvalo 41 hodin a 28 minut. Na následujícím grafu 5.4 jsou znázorněny čísla kol, ve kterých dopadení Mistra X proběhlo. Nutno podotknout, že první dva tahy agentů byly aktuálně prováděny zcela náhodně. V případě takového kola dopadení se tedy jednalo obvykle o náhodné postavení agenta na místo, kde zrovna skrytý Mistr X stál. Již po provedení tohoto experimentu bylo tedy dosaženo v celku dobrého výsledku. V porovnání s prvotním pokusem využití Monte Carlo Tree Search kolegy Sovy [8] již bylo dosaženo lepšího výsledku. Vzhledem k implementaci s využitím Alfa-Bety však stále bylo co zdokonalovat tak, aby byl Mistr X ovládaný touto metodou poražen. Cílem se tedy pro další experimenty jevilo vylepšit implementaci

tak, aby byla úspěšnost alespoň 50 %, tudíž by si obě metody v tomto vzájemném postavení Mistra X za Alfa-Betu a agenti za Monte Carlo Tree Search konkurovaly.



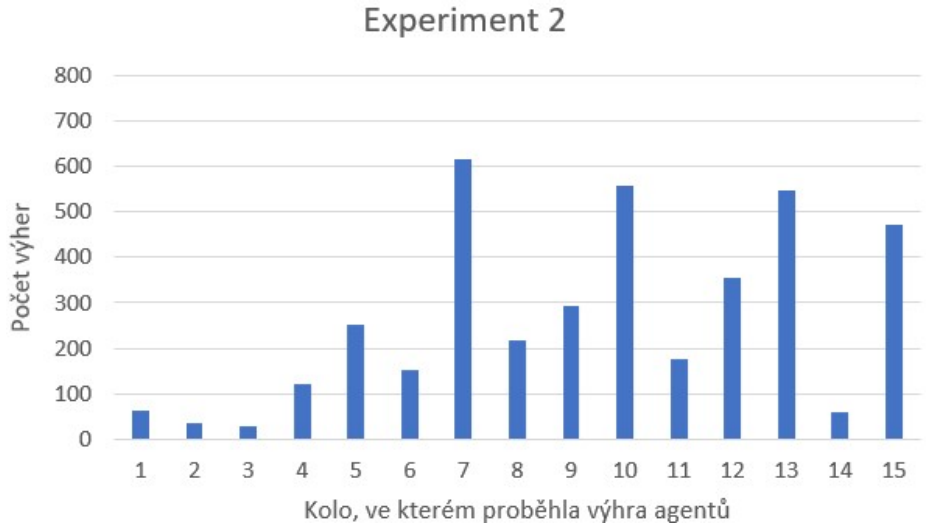
Obrázek 5.4: **Počet výher agentů dle daných kol v experimentu 1** Zdroj: Vlastní snímek

Experiment číslo 2

Úkolem druhého experimentu byl pokus o zvýšení úspěšnosti pomocí zaměření se na počáteční rozestavení agentů při prvních dvou kolech, kdy nemají k dispozici žádnou informaci, kde by se mohl Mistr X nacházet. Důležitým faktorem tedy bylo jakkoliv agenty rozprostřít po mapě tak, aby ve třetím kole mohli již podle zobrazení aktuální polohy Mistra X postupovat z výhodnějších pozic. Logicky se tato úprava jevila jako vylepšení, tudíž jsem se pokusil o zavedení této úpravy do implementace.

Implementace byla tedy změněna tak, že v případě prvního či druhého kola byli agenti dle jejich pozice navigování vždy kolem středu hracího pole tak, aby se z něj poté mohli ve třetím kole vydat již správným směrem za Mistrem X. Tato volba byla provedena na základě faktu, že bylo hráno na poli o velikosti 5x5, takže dosah agentů na Mistra X mohl být při výhodném postavení klidně jen jedno pole. Při provedení na větším poli by však tato myšlenka musela být rozvedena více v závislosti postavení agentů vůči sobě.

Při opětovném počtu 10 000 provedených her bylo tentokrát 3 937 her zakončeno výhrou agentů. Úspěšnost tohoto vylepšení se tedy navýšila pouze o $0,05\,\%$ na $39,37\,\%$. Časová náročnost této verze oproti té v experimentu 1 však vzrostla o 5 hodin a 29 minut na celkových 46 hodin a 57 minut. Vzhledem k této skutečnosti by se tedy dalo uznat, že byly verze obou experimentů úspěšností totožné, ale druhá verze byla rozhodně výrazně časově náročnější. Na následujícím grafu 5.5 jsou opět znázorněny čísla kol dopadení Mistra X, přičemž v něm není dosaženo nějakých větších vychýlení oproti experimentu 1.



Obrázek 5.5: **Počet výher agentů dle daných kol v experimentu 2** Zdroj: Vlastní snímek

Experiment číslo 3

Tento experiment se věnoval druhému zmíněnému problému z experimentu 1. Tímto problémem bylo omezení vzájemné spolupráce agentů jako takových. V mnoha situacích ve vyhledávacím stromě metody vyhrávaly tahy, kdy jeden z agentů sice šel na jednu z nejvýhodnějších pozic, ale druhý tak neučinil a šel v naprosto nelogickém směru od Mistra X. V takovém případě se jevily jako možné dvě vylepšení. V prvním případě by šlo agenty nechat rozmýšlet každého samotného za sebe. V tom druhém čistě podporovat možnosti, kde agenti v následujícím tahu zaberou více míst, kde by mohl Mistr X stát. Vzhledem k již aktuálnímu stavu implementace vyhledávacího stromu pro všechny agenty najednou jsem se rozhodl pro možnost druhou.

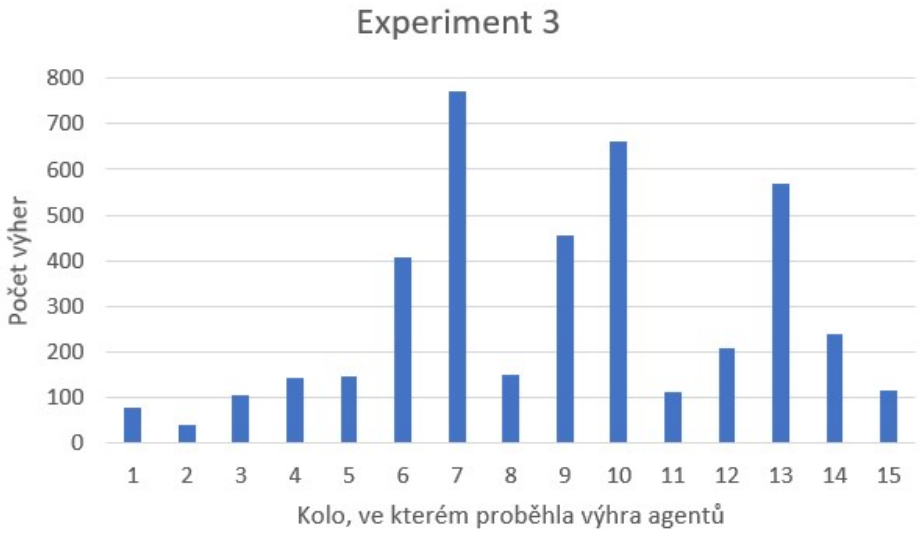
Při této situaci bylo nutno hlavně změnit generování vyhledávacího stromu v počáteční fázi, kdy je expandován kořenový uzel. Zároveň bylo také nutno zlepšit vyhodnocování skrz simulaci, a to tak, aby byly výrazně lépe ohodnoceny tahy, které byly obsaženy hned v potomkovských uzlech kořene a zároveň by Mistra X dopadly. V takovém případě tato úprava hlavně ovlivnila situaci, kdy byl Mistr X viděn, a tak byl jasně ohodnocen tah, kdy mohl agent v následujícím tahu stoupnout přesně na toto pole Mistra X, pokud mu to jeho pozice dovolovala.

Další úpravou byl stav, kdy nebyl pohyb agentů již jen zaměřen na poslední pozici Mistra X, jako tomu bylo doposud. V případě druhého skrytého tahu Mistra X tak přicházela v potaz všechna možná místa Mistra X a agenti prováděli simulace nad všemi těmito pozicemi. Přibylo tak rozhodně více uzlů ve vyhledávacím stromu, ale byla díky tomu razantně zvýšena pravděpodobnost dopadení Mistra X.

Díky tomuto vylepšení se tedy povedlo docílit lepších možností dopadení Mistra X. V některých předešlých případech totiž nastávala situace, kdy agenti směřovali na místo, kde byl Mistr X naposledy spatřen, ale ten již stihl uniknout a přesunout se podél hrany hrací plochy úplně na opačnou stranu. Zamýšlení všech možných pozic tak této situaci zabránilo.

Z opětovného počtu 10 000 her v tomto experimentu dopadlo 4 207 her jako výherních pro agenty. Celkově bylo tedy dosaženo doposud nejvyšší úspěšnosti $42,07\,\%$. Tento experi-

ment byl doposud časově nejrychlejším a trval 40 hodin a 29 minut. Zároveň se projevil počet výher již v dřívějších kolech, než tomu bylo v experimentech předešlých, což je opět vyobrazeno na následujícím grafu 5.6.



Obrázek 5.6: **Počet výher agentů dle daných kol v experimentu 3** Zdroj: Vlastní snímek

5.4 Popis úprav a zlepšení implementace

Jak již tedy bylo popsáno v předešlé sekci, díky experimentům se povedlo zdokonalit původní implementaci systému. Jelikož však nebylo dosaženo překonání hranice 50 % úspěšnosti, bylo provedeno ještě několik úprav a následných testů s těmito úpravami. Obvykle se jednalo o malé úpravy v rámci ošetření některých situací, které nastávaly pouze v několika případech, a proto nebyly lehce odhalitelné.

Při finálních úpravách bylo tedy zvoleno, že se agenti budou v prvních dvou tazích pohybovat náhodně, jelikož tato úprava neposkytovala nijak výrazný rozdíl v úspěšnosti a pouze zvyšovala časovou náročnost. Vzhledem ke spuštění mnoha her za sebou to poté prodlužovalo dobu testu, ale nebylo tím dosaženo žádné vyšší úspěšnosti. Detailnější popis finálních úprav je poté popsán v následující sekci.

Popis finálních úprav

Při finálních úpravách jsem se ještě zaměřil na výpis pohybu agentů jako takový a snažil se přijít na příčiny některých tahů, které agenti prováděli, i když se jevily jako méně výhodné. Bylo tomu tak například při tahu agentů do protisměru od aktuálně zobrazeného Mistra X při vzdálenějších pozicích. Při takovéto situaci metoda zaostávala, jelikož měli agenti na výběr všechny směry, kdy simulace mohla vyhodnotit i ty ve směru od Mistra X jako výhodné. V takovém případě, bylo při získání ohodnocení možné, že v těchto případech občas agenti jednoduše našli Mistra X až ve velice pozdí fázi hry v rámci jedné simulace daného uzlu, a tak nebylo vůbec jasné, kam se mají v celkovém daném tahu pohnout.

Prvotní úpravou bylo při simulaci ohodnocení těch výher, které proběhly v dřívějších kolech. Toto vylepšení se týkalo procesu simulace, kdy se hned při získání výhry během

simulace navrátila i poměrová hodnota dané výhry. Tím pádem v případě výher, které nastaly hned z uzlu, který byl expandován jakožto druhý, byla navrácena nejen výhra, ale i hodnota pojmenována jako *value*, která byla vypočítána jakožto poměr aktuálního kola, ve kterém výhra proběhla a maximálního počtu kol. V takovém případě se poté tato hodnota jednoduše přičetla k počtu výher. Díky ní poté probíhal výběr daných tahů o něco lépe. Ve výsledné verzi však bylo toto mírné zlepšení úspěšnosti naprosto zanedbatelné a bylo tedy nutno se zaměřit na další možná vylepšení.

Jako další proběhla úprava ohledně zaměření se na expandované stavy. Jelikož jich bylo obvykle vygenerováno velké množství, neproběhl občas průchod celým stromem dostatečně vůči správnému rozhodnutí. V některých chvílích nastala situace, že byly vygenerovány potomci kořene vyhledávacího stromu pouze do hloubky 3 až 4. V takovém případě se tedy jevilo jako příhodné zaměřit se na myšlenku odstranění některých stavů, které nemohly v daných případech nastat. Bylo tomu tak například pro stavy, kdy by byl Mistr X hned vedle agenta. Mohlo se tím pádem počítat s jeho základní inteligencí, že nevstoupí na pole agenta, jelikož by okamžitě hru prohrál. Proběhlo tak vždy odstranění stavů, které z hlediska pravidel nemohly nastat.

Poslední a zároveň hlavní úpravou, která nastala se stalo ještě větší vylepšení generování stavů při expanzi jako tomu bylo v předešlém odstavci. Základem bylo zaměření se na logiku agentů netáhnout vůbec na místa, která jsou od Mistra X více vzdálená než jejich dosavadní pozice. Toto vylepšení tedy v rámci generování všech možných tahů agentů bralo v potaz pouze ty směry, které byly vůči všem možným pozicím Mistra X nejhůře stejně daleko jako poloha jednotlivých agentů. Takto nastavená úprava pak tedy zaručovala neustálý pohyb agentů buď směrem k Mistru X anebo v daných směrech tak, aby byl pořád obkličován.

Jednotlivé úpravy se tedy zaměřovaly na zdokonalení různých situací hry, které nenastávaly pokaždé, ale i tak celkem výrazně ovlivňovaly hru. Nakonec se hlavně díky poslední zmíněné úpravě podařilo implementaci metody Alfa-Beta pro Mistra X touto implementací Monte Carlo Tree Search pro agenty překonat a dosáhnout tak úspěšnosti agentů přes 50 %. Ve finálním experimentu bylo tedy pro výše popsanou upravenou implementaci opět provedeno 10 000 her přičemž 7 057 her skončilo výhrou agentů. Bylo tak dosaženo úspěšnosti 70, 57 %. Tento experiment při porovnání s předešlými trval pouze 35 hodin a 23 minut a finální verze se tak stala tou nejúspěšnější.

5.5 Zhodnocení

Vzhledem k původnímu cíli, kterým bylo překonat úspěšnost agentů verze s využitím metody Monte Carlo Tree Search M. Sovy viz [8], bylo již při prvním experimentu dosaženo výborného výsledku. Jeho úspěšnost 20, 8 % tak byla za použití naprosto stejných podmínek překonána takřka dvojnásobně. Hlavním cílem dalších úprav implementace se tedy stalo zvýšení vlastní úspěšnosti alespoň na 50 %. Realizace systému pro hraní zjednodušené verze hry Scotland Yard byla tedy zdokonalena finálními úpravami, díky kterým byla hranice 50 % nejen dosažena, ale i překonána díky nejvyššímu výsledku úspěšnosti 70, 57 %.

Za účelem porovnání všech třech systémů byl proveden ještě finální test, při kterém byly tyto systémy postaveny proti sobě jak v rolích agentů, tak Mistra X. Bylo tedy naposledy provedeno 10 000 her. Sledována byla tak jako v předešlých experimentech úspěšnost agentů při snaze o dopadení Mistra X. V tabulce 5.1 jsou uvedeny míry úspěšnosti agentů proti Mistru X při vzájemném postavení všech jednotlivých systémů proti sobě.
Vzhledem k tomuto porovnání lze říci, že bylo dosaženo výrazného zlepšení metody Monte Carlo Tree Search. Důležitým faktorem zadání bylo také dosáhnout možnosti hraní této

	MCTS Gerža 2022	MCTS Sova 2021	A-B Tulušák 2020
MCTS Gerža 2022	–	94,67 %	70,57 %
MCTS Sova 2021	25,71 %	–	16,88 %
A-B Tulušák 2020	77,64 %	99,02 %	–

Tabulka 5.1: Úspěšnost jednotlivých verzí metod agentů (řádky) proti verzím metod Mistra X (sloupce) (vždy provedeno 10 000 her)

hry autonomně tak, aby bylo možno jakožto hráč hrát proti počítači jak za agenta, tak za Mistra X. V mnoha ohledech při hře proti hráči i tento systém tak jako systém s využitím metody Alfa-Beta stále zaostává, jelikož je jeho logika po pár desítkách her pro lidského hráče lehce přemožitelná. Oproti systému s využitím Alfa-Bety však ve více hrách nastává stav, kdy se systém s metodou Monte Carlo Tree Search rozhoduje mírně náhodně z důvodu využití simulací. Díky tomu tak nenastává v dané situaci vždy tah stejný, a tak je pro lidského hráče styl hry tohoto systému rozhodně hůře zapamatovatelný.

Kapitola 6

Závěr

Cílem této práce bylo navrhnout a implementovat systém pro autonomní hraní zjednodušené verze hry Scotland Yard a následně poměřit danou implementaci s již předešlými řešeními, které byly doposud zkonstruovány.

Tento cíl jsem splnil na základě porovnání všech systémů vůči sobě, kdy bylo dosaženo nadprůměrné úspěšnosti z role agentů mého systému. Z pohledu druhé role (Mistra X) byl však můj systém umístěn úspěšností mezi ostatní dva systémy, tudíž by zde ještě šlo rozhodně docílit zlepšení.

Na počátku práce jsem se seznámil s možnostmi výskytu neurčitosti v pravidlech některých her, přičemž jsem se hned v další fázi zaměřil na pravidla zadané hry Scotland Yard, a hlavně její zjednodušené verze, která byla použita již v předešlých pracích. Pravidla této zjednodušené verze jsem nastudoval vzhledem k výskytu neurčitosti v nich. Poté jsem se zaměřil na důkladné prostudování možných metod, které by byly vhodné pro použití při tvoření systému pro autonomní hraní zjednodušené verze této hry. Při vybírání takové metody jsem se zároveň zaměřil na již předešlé dvě realizace s využitím metod Alfa-Beta a Monte Carlo Tree Search, kdy jsem zjistil, že konkrétně druhá z nich byla výrazně neúspěšná. Nejen z toho důvodu jsem se rozhodl implementovat systém s využitím metody Monte Carlo Tree Search.

Vzhledem k implementaci jsem využil již předešlých dvou řešení jakožto dvou systémů, se kterými jsem celkově při implementaci spolupracoval a povedlo se mi je spojit i s mou implementací systému do jednoho programu, který využívá jak hraní autonomně jeden systém proti druhému, tak reálného hráče proti jednomu ze systémů. S touto realizací jsem tedy poté prováděl experimenty za účelem zdokonalení implementace vlastního systému. Při experimentech se mi podařilo dosáhnout zvýšení úspěšnosti při hraní mého systému za stranu agentů skoro na 50 %. Ve finálních úpravách se poté povedlo tuto hodnotu překonat a dosáhnout výborné 70 % úspěšnosti.

S finálními úpravami bylo tedy také provedeno velké porovnání všech systémů proti sobě, kdy bylo dosaženo výborných výsledků hlavně vůči předešlé realizaci, která také využívala metody Monte Carlo Tree Search. Povedlo se tak také poukázat na některé možné chyby předešlé realizace, které byly v této práci konzultovány.

Práce pro mě byla přínosná vzhledem k poznání různých metod řešení tohoto či podobných úkolů. Vzhledem k již dosaženým výsledkům by se mohlo v daném řešení navázat jeho úpravami za cílem dosažení systému pro hraní hry Scotland Yard s celkovými pravidly.

Literatura

[1] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P. et al. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*. Březen 2012, 4:1, s. 1–43. DOI: 10.1109/TCIAIG.2012.2186810. Dostupné z: https://www.researchgate.net/publication/235985858_A_Survey_of_Monte_Carlo_Tree_Search_Methods.

[2] Kramer, W. *6 bere!* - *pravidla hry* [online]. 2019. Dostupné z: https://www.zatrolene-hry.cz/soubory/349/7152.pdf.

[3] Liu, M. *General Game-Playing With Monte Carlo Tree Search* [online]. medium.com, říjen 2017 [cit. 2022-25-04]. Dostupné z: https://medium.com/@quasimik/monte-carlo-tree-search-applied-to-letterpress-34f41c86e238.

[4] Neuvedeno. *Stratego* - *pravidla hry* [online]. 1947. Dostupné z: https://www.zatrolene-hry.cz/soubory/182/77.pdf.

[5] Neuvedeno. *Scotland yard* - *pravidla hry* [online]. 1983. Dostupné z: https://www.zatrolene-hry.cz/soubory/696/2787.pdf.

[6] Nijssen, J. P. A. M. a Winands, M. Monte-Carlo Tree Search for the Game of Scotland Yard. In:. říjen 2011, s. 158–165. DOI: 10.1109/CIG.2011.6032002. Dostupné z: https://www.researchgate.net/publication/224259872_Monte-Carlo_Tree_Search_for_the_Game_of_Scotland_Yard.

[7] Russell, S., Norvig, P. et al. *Artificial Intelligence, A Modern Approach*. 3. vyd. Pearson, 2010. ISBN 978-0-13-604259-4.

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

[9] Stegmaier, J. *Charterstone* - *pravidla hry* [online]. 2017. Dostupné z: https://www.zatrolene-hry.cz/soubory/7265/8645.pdf.

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

[11] Zbořil, F. V. a Zbořil, F. *Základy umělé inteligence: Studijní opora*. Duben 2021.

Příloha A

Obsah přiloženého paměť. média

Přiložené paměťové médium obsahuje soubor s textem této práce ve formátu pdf a adresář s názvem *BP_MartinGerza_xgerza00*. V tomto adresáři se vyskytují vlastní skripty programu napsány v jazyce Python a dva adresáře jménem *AT_2020* a *MS_2021*. Obsahem těchto dvou adresářů jsou taktéž skripty v jazyce Python, sloužící pro chody dvou systémů z bakalářských prací A. Tulušáka [10] a M. Sovy [8]. Soubory v těchto adresářích tedy byly převzaty z původních implementací a byly pouze uloženy do jednotlivých modulů, aby byl celý zdrojový kód programu přehlednější. Jediným zásahem do souborů těchto adresářů bylo jejich vzájemné provázání tak, aby bylo z hlavní funkce (souboru *main.py*) možno využívat všech tří systémů dle vlastní volby.

V adresáři *AT_2020* se tedy nachází tyto převzaté soubory ([10]):

- alfa_beta.py
- alfa_beta_mrx.py
- alfabeta_node.py
- environment.py
- player_input.py

V adresáři *MS_2021* potom tyto převzaté soubory ([8]):

- alfa_beta_wrapper.py
- mc_environment.py
- monte_carlo.py
- player.py

V nadřazeném adresáři dvou předchozích, tedy v *BP_MartinGerza_xgerza00* jsou poté obsaženy tyto vlastní (2 z nich převzaty a upraveny) skripty a soubor *README.md*:

- argparser.py
- main.py – převzat pouze koncept z *MS_2021* [8] a takřka celý předělán (nevyznačeno)
- mc_environment.py – převzat z *MS_2021* [8] a upraven dle potřeb (vše vyznačeno)
- MG_monte_carlo.py
- README.md