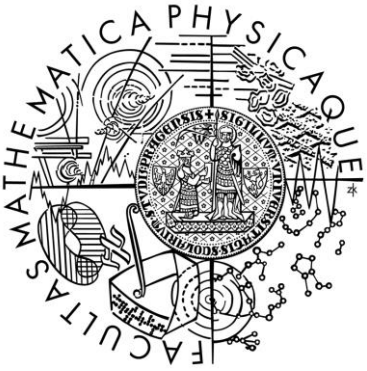
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Ľubomír Ohman

Umělý hráč pro karetní hru Hearthstone

Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Jakub Gemrot

Studijní program: Informatika (B1801)

Studijní obor: Obecná informatika

Praha 2016

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V …..... dne..... podpis

Název práce: Umělý hráč pro karetní hru Hearthstone

Autor: Ľubomír Ohman

Katedra / Ústav: Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Jakub Gemrot, Katedra softwaru a výuky informatiky

Abstrakt: Cieľom tejto práce bolo vytvoriť umelého agenta, ktorý je schopný sa naučiť hrať Hearthstone s daným balíčkom kariet. Na jeho dosiahnutie sme sa rozhodli použiť Q-učenie. Vedľajší efektom tejto práce je transformácia jednoduchého simulátoru hry Hearthstone na framework pre vývoj umelej inteligencie pre túto hru. Pre účely trénovania a vyhodnocovania nám poslúžili bežne hrané stratégie ako inšpirácia pre testovacích agentov, ktorých sme vytvorili. Táto práca obsahuje porovnanie tabuľkovej reprezentácie funkcie Q a jej aproximácie neurónovou sieťou. Pôvodný cieľ bol splnený čiastočne. Boli sme úspešný vo vytvorení umelého agenta, ale ten sa dokáže naučiť len jednu špecifickú stratégiu.

Klíčová slova: Q-učenie, Hearthstone, umelý hráč

Title: Artificial Player for Hearthstone Card Game

Author: Ľubomír Ohman

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Jakub Gemrot, Department of Software and Computer Science Education

Abstract: The goal of this work was to create an artificial agent that is able to learn how to play Hearthstone with given deck of cards. We decided to use Q-learning algorithm to achieve it. The side effect of this work is the transformation of the simple simulator of Hearthstone into the framework for developing Artificial Intelligence in this game. For the purpose of training and evaluation, commonly played strategies served us as inspiration for the testing agents that we developed. This work contains comparison of the table representation of Q-function and the neural network approximation of it. The original goal was fulfilled partially. We were successful in the creation of the learning agent but he is only able to learn one specific strategy.

Keywords: Q-learning, Hearthstone, artificial player

V prvom rade by som sa chcel poďakovať vedúcemu práce, Mgr. Jakubovi Gemrotovi, za jeho nadšenie pre túto prácu, ktoré veľmi pomohlo k jej dokončeniu. Jeho pomoc spočívala nielen v technických radách, ale aj vo skvelých nápadoch a vysokom záujme o priebeh práce.

Moja vďaka patrí takisto mojej rodine, ktorá mi umožnila študovať a počas celého života mi bola mojou nielen materiálnou, ale predovšetkým psychickou podporou.

Obsah

1 Úvod

1.1 Popis práce

Primárnym objektom záujmu tejto práce je vývoj umelého hráča pre hru Hearthstone. Hra Hearthstone je kartová hra pre dvoch hráčov, ktorá bola vydaná v roku 2014. Vzhľadom na jej mladý vek je z hľadiska umelej inteligencie preskúmaná veľmi slabo. Hra sa dá veľmi dobre reprezentovať stavmi a akciami. Okrem toho ponúka veľkú flexibilitu v oblasti cielenia agentovho správania. Tým je myslené, že je možné posielať agentovi pozitívnu alebo negatívnu spätnú väzbu nielen na základe výhier a prehier, ale aj po vykonaní akcií, ktoré nakoniec k výhram alebo prehrám vedú. To jej dáva potenciál byť vhodným prostredím pre agenta, ktorý sa učí, ktoré akcie má na základe odmeny vykonať, za predpokladu, že sa nachádza v nejakom konkrétnom stave. Takýmto parametrom zodpovedá spätnoväzobné učenie. V doterajších prácach venujúcich sa problematike umelej inteligencie pre hru Hearthstone nikto neskúsil vytvoriť agenta, ktorý sa postupne učí hrať. Úmysel spraviť to bol však predstavený v práci, ktorej autorom je David Taralla (1), konkrétne tam bol popísaný úmysel v budúcnosti skúsiť použiť na vývoj agenta pre Hearthstone Q-učenie, čo je technika spätnoväzobného učenia. Toto sú najzásadnejšie dôvody, ktoré viedli k tomu, že táto práca skúma, ako dobre funguje v prostredí hry Hearthstone Q-učenie a jeho rôzne varianty.

1.2 Ciele práce

Cieľom práce je vytvorenie umelého hráča, ktorý je schopný sa učiť hrať hru Hearthstone s daným balíčkom kariet. Voľba Q-učenia prináša ďalšie výzvy, medzi ktoré patrí návrh priestoru stavov a akcií ako aj dizajn odmeňovacej funkcie.

Tento hlavný cieľ produkuje ďalšie požiadavky. Pre učenie je potrebný simulátor hry. Kvôli obmedzeným schopnostiam existujúcich simulátorov sa práca venuje aj rozšíreniu simulátoru FirePlace na evaluačný framework.

Cieľový agent sa potrebuje učiť. Táto potreba učenia spolu s nutnosťou evaluácie vytvára požiadavku mať agentov, pomocou ktorých sa cieľový agent môže učiť aj vyhodnocovať svoje schopnosti. Preto je ďalším vedľajším cieľom vytvorenie testovacích agentov schopných hrať s aspoň jedným pevným balíkom porovnateľne so skutočnými hráčmi.

1.3 Štruktúra textu

V druhej kapitole popisujeme hru Hearthstone. Približuje pravidlá a systém hry aj čitateľovi, ktorý ju nikdy nehral. Tento popis je dôležitý preto, lebo celá práca pracuje s bežne používanými termínmi v hre.

Dôvody kľúčových rozhodnutí je možné nájsť v tretej kapitole. Okrem výberu algoritmu sa v nej nachádza aj popis východiskovej situácie pred začiatkom vývoja. Táto situácia je ovplyvnená najmä dvomi existujúcimi prácami o umelej inteligencii v hre Hearthstone.

Teoretické poznatky o Q-učení sú popísané vo štvrtej kapitole. Je v nej priblížený samotný algoritmus ako aj dve rôzne alternatívy jeho implementácie.

Piata kapitola pokrýva problematiku testovacích agentov. Obsahuje popis techník, ktorých sa títo agenti držia. Rovnako je v nich vyhodnotená aj ich úspešnosť.

Šiesta kapitola prezentuje použitý simulátor. Vysvetľuje, prečo je potrebný a dôležitý, a odôvodňuje výber simulátoru FirePlace. Sú v nej popísané naše rozšírenia, teda grafické záznamy hier, evaluačná funkcionalita, ako aj zjednodušené prostredie pre vývoj umelých hráčov.

Komplexný popis nášho Q-agenta učiaceho sa pomocou neurónovej siete sa nachádza v siedmej kapitole. Je v nej aj podrobné vysvetlenie, aké akcie, stavy a odmenu využíva.

Obsahom ôsmej kapitoly je popis Q-agenta učiaceho sa pomocou tabuľky. Vysvetľuje spôsob obmedzenia stavov z predchádzajúcej kapitoly tak, aby ich bolo možné uchovať v tabuľke.

Deviata a desiata kapitola sú zhrnutím výsledkov dosiahnutých rôznymi variantmi Q-agentov. Je v nich zahrnuté vyhodnotenie týchto výsledkov, ako aj bilancia úspešnosti celej práce.

Posledná kapitola uvádza možnosti rozšírenia tejto práce v budúcnosti vzhľadom na výsledky a nadobudnuté poznatky.

2 Hearthstone

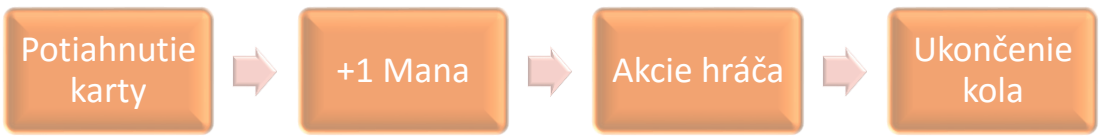
2.1 Priebeh hry

Hearthstone je ťahová, kartová hra pre dvoch hráčov, ktorých cieľom je poraziť protivníka. Každý hráč má hrdinu a každý hrdina má na začiatku tridsať životov a špecifickú schopnosť. V tejto kapitole sa budeme venovať primárne popisu pravidiel hry a všetkých entít na hracej ploche (pozri obrázok 2.2), ktoré ovplyvňujú priebeh hry.

Na docielenie výhry hráči okrem schopnosti hrdinu využívajú aj balíček tridsiatich kariet, ktorý si zvolia ešte pred začatím hry. Množinu kariet, z ktorých hráč môže vyberať, ovplyvňujú komplikované pravidlá, ktoré pre túto prácu nie sú podstatné.

Na začiatku hry je náhodne zvolený hráč, ktorý začína. Tento hráč dostane tri karty zo svojho balíčku. Jeho protivník dostane štyri karty zo svojho balíčku. V tejto fáze hry majú obaja hráči možnosť zamiešať ľubovoľný počet z kariet, ktoré dostali späť do balíčku, a dostať za ne náhodnú náhradu. Karty si hráči navzájom nevidia.

Po ukončení prvej fázy nastáva samotná hra (pozri obrázok 2.1). Na začiatku každého kola sa hráčovi doplní jeho mana na maximálny počet. Tento maximálny počet je v prvom kole jedna a zvyšuje sa v každom kole o jedna až do desiateho kola. Po desiatom kole sa už maximálny počet many nezvyšuje, stále však platí, že každý začiatok kola znamená doplnenie many na tento maximálny počet. Manou hráči platia za hranie kariet. Nové kolo prináša pre hráča tiež novú kartu z vrchu svojho balíčku. Hráč môže mať na ruke maximálne desať kariet. Po prekročení tejto hranice je každá potiahnutá karta zničená a mizne z hry. Ak hráč minul všetky karty z balíčku, namiesto každého potiahnutia sa jeho hrdinovi míňajú životy. Počet minutých životov je rovný počtu pokusov o potiahnutie karty z prázdneho balíčku.



Obrázok 2.1: Priebeh kola

Hra končí v momente, keď aspoň jeden z hrdinov zomrie, teda počet jeho životov je menší ako jedna. Ak sa to stane obom hrdinom naraz, prehrávajú obaja hráči. V prípade, že zomrie len jeden hrdina, prehráva jeho majiteľ a protivník vyhráva.

Obrázok 2.2: Hracia plocha, zdroj: 55[O1]

2.2 Vlastnosti kariet

2.2.1 Všeobecné

Existujú tri spôsoby, ktorými môže hráč vykonávať akcie. Prvým z nich sú útoky postavičiek a hrdinu, druhým použitie schopnosti hrdinu a posledným hranie kariet. Každá karta má svoju cenu, raritu a prevažná väčšina aj textový popis (pozri obrázok 2.3).

Obrázok 2.3: Karta, zdroj: [O2]

Základnou vlastnosťou karty je jej cena. Tá udáva, koľko jednotiek many musí hráč zaplatiť, ak ju chce zahrať. Každá karta má svoju cenu, tá môže byť ovplyvňovaná inými javmi v hre. Hráč môže zahrať len kartu, ktorej cena je nižšia alebo rovná jeho aktuálnemu počtu jednotiek many.

Väčšina kariet má sivý okraj. To určuje, že karta je použiteľná všetkými hrdinami. Ak má karta okraj inej farby, znamená to, že je určená len pre hrdinov, s ktorými je táto farba asociovaná. Okrem toho môžu mať niektoré karty zlaté okraje. Zlatá farba okrajov ale neznamená pre samotnú hru nič. Má funkciu pri vytváraní kariet.

Karty, ktoré tvorcovia hry Hearthstone určili za výnimočné, majú v sebe drahokam. Farba tohto drahokamu určuje ich raritu. Karty s vyššou raritou bývajú silnejšie, lenže v praxi sa kombinujú karty bez ohľadu na ich raritu. Dôvodom je, že dôležitejšie ako toto ohodnotenie je pre hru celková schopnosť kooperácie kariet v balíčku.

Niektoré z kariet majú špeciálny textový popis, ktorý slovne vyjadruje efekt karty. Tieto efekty môžu byť veľmi rozmanité a nedajú sa úplne zovšeobecniť (príklady v podkapitole 2.2.2 nižšie) .

Existujú štyri základné kategórie kariet: postavička, kúzlo, zbraň a tajomstvo, ktoré si predstavíme v nasledujúcich podkapitolách.

2.2.2 Postavičky

Zahranie karty tejto kategórie spôsobuje vytvorenie postavičky na hracej ploche, ktorá bude bojovať za svojho hrdinu. Každá postavička má dve základné vlastnosti: útok a životy (pozri obrázok 2.4).

Postavička zostáva v hre, kým má aspoň jeden život. Ak stratí všetky životy, zomiera a mizne z hry.

Útok vyjadruje množstvo životov, ktoré uberie inej postavičke alebo hrdinovi pri konflikte. Konflikty vznikajú útokmi. Každá postavička môže útočiť raz za kolo, prvýkrát ale až jedno kolo po tom, v ktorom bola zahraná. Útok uberá životy pri každom konflikte, bez ohľadu na to, ktorá postavička útočí. Útočiť môže aj hrdina v prípade, že sa mu nejakým spôsobom podarilo získať útok. V prípade hrdinu funguje útočenie takmer rovnako ako v prípade postavičiek s tým rozdielom, že ak je obranca hrdina, nespôsobuje postavičke stratu životov.

Všetky efekty vyplývajúce z textu na postavičke zostávajú platné aj po jej zahraní. Text môže byť rôznorodý rovnako ako pri ostatných typoch kariet, najdôležitejšie efekty sú však veľmi frekventované, a preto majú aj špeciálne jednoslovné názvy. Tieto efekty sú nasledujúce:

1. Charge. spôsobuje, že postavička môže útočiť už počas kola, v ktorom je zahraná.
2. Taunt. Ak chce hráč útočiť a jeho protivník má jednu alebo viac postavičiek s touto vlastnosťou, musí zaútočiť na jednu z nich.
3. Divine shield. Prvé poškodenie postavičky jej neuberie životy.
4. Battlecry. Efekt textu, ktorý nasleduje po tomto nápise, sa prejaví po zahraní postavičky.
5. Deathrattle. Text, ktorý nasleduje po tomto nápise, zaúčinkuje po smrti postavičky.
6. Inspire. Účinok textu, ktorý nasleduje po tomto nápise, sa prejaví vždy, keď je postavička v hracom poli a priateľský hrdina použije svoju schopnosť.

Príkladom postavičky je „Sneed's Old Shredder" (pozri obrázok 2.4). Má útok päť a sedem životov. Stojí osem jednotiek many. Oranžový drahokam určuje, že jeho rarita dosahuje legendárnu úroveň. Textový popis v jeho prípade znamená, že po jeho

smrti sa na hracej ploche zjaví náhodná legendárna postavička, ktorá bude slúžiť majiteľovi zomretej postavičky.

Obrázok 2.4: Postavička 1 (vľavo podoba na ruke, vpravo podoba na hracej ploche. Zdroj: [O2])

Ďalším zástupcom skupiny postavičiek je „Annoy-o-Tron“ (pozri obrázok 2.5). Má útok jeden a dva životy. Jeho cena sú dve jednotky many. Biely drahokam v strede karty priraďuje tejto karte všednú raritu. Textový popis je stručný, určuje, že postavička má „Taunt“ a „Divine Shield“ (pozri zoznam efektov na predchádzajúcej strane).

Obrázok 2.5: Postavička 2, zdroj: [O2]

2.2.3 Kúzla

Ďalším typom kariet sú kúzla. Táto skupina je najvšeobecnejšou skupinou kariet. Okrem toho, že majú cenu, neexistuje vlastnosť, ktorá by ich spájala. Všetky ich efekty závisia na texte karty. Ich efekt sa prejaví hneď po zahraní.

Jedným z predstaviteľov skupiny kúziel je „Quick Shot“ (pozri obrázok 2.6). Stojí dve jednotky many a biely kryštál uprostred karty indikuje mimoriadnu raritu. Jej efekt je ubratie troch životov cieľovej postavičke alebo hrdinovi. Ak hráčovi po zahraní tejto karty nezostanú na ruke žiadne karty, jednu si potiahne.

Obrázok 2.6: Kúzlo 1, zdroj: [O2]

Karta „Gang up“ (pozri obrázok 2.7) je dobrým príkladom toho, aké rozmanité efekty môžu kúzla spôsobovať. Je to karta s mimoriadnou raritou a cenou dvoch jednotiek many. Hráč sa rozhodne pre postavičku, ktorej tri kópie budú zamiešané do jeho balíčku.

Obrázok 2.7: Kúzlo 2, zdroj: [O2]

2.2.4 Zbrane

Zbrane sú nástrojom hrdinov, pomocou ktorých môžu útočiť. Každý hrdina môže mať v jednom momente maximálne jednu zbraň. Ak už hrdina zbraň má a hráč zahrá ďalšiu, hrdina príde bez náhrady o starú a získa novú. Zbrane majú dve základné vlastnosti: útok a výdrž.

Útok pridáva hrdinovi rovnaký útok, aký bol popísaný pri postavičkách. Tento útok ale nie je permanentný a hrdina ním disponuje len kým má danú zbraň.

Výdrž zbrane určuje počet útokov, ktoré hrdina môže vykonať s danou zbraňou. Tento počet sa po každom útoku znižuje a v momente, keď klesne na nulu, je zbraň zničená a vyradená z hry.

Zbrane môžu mať takisto textový popis. Funguje rovnako ako pri iných typoch kariet. Význam popisov Battlecry, Dearthrattle a Inspire je analogický významu v prípade postavičiek.

Karta „Argent Lance“ je zbraň, ktorá má útok aj výdrž dva (pozri obrázok 2.8). Modrý drahokam v jej strede definuje vzácnosť karty. Stojí dve jednotky many a obsahuje textový popis. Tento popis pridáva karte efekt, ktorý spôsobuje odhalenie náhodnej postavičky z balíčku oboch hráčov. V prípade, že postavička hrdinu, ktorý kartu zbrane zahral, stojí viac ako nepriateľská, zbraň získava jednu výdrž navyše.

Obrázok 2.8: Zbraň 1, zdroj: [O2]

Ďalšou zbraňou v hre je “Death’s Bite” (pozri obrázok 2.9). Za jej zahranie musí hráč zaplatiť štyri jednotky many. Zbraň má mimoriadnu raritu a okrem toho, že má útok štyri a výdrž dva, jej efekt upravuje textový popis. Ten určuje, že zničenie tejto zbrane spôsobí poškodenie vo výške jedna všetkým postavičkám na hracej ploche.

Obrázok 2.9: Zbraň 2, zdroj: [O2]

2.2.5 Tajomstvá

Podobne ako postavičky, zostávajú tajomstvá v hre dlhšie po zahraní. Ich efekty môžu byť rovnako rôznorodé ako účinky kúziel. Po zahraní sa tajomstvo dostane do čakacej fázy. Kým sa nachádza v tejto fáze, protivník nepozná jeho efekt, čo je tiež vysvetlením jeho názvu. Každé tajomstvo má definovanú akciu, ktorá aktivuje jeho efekt. Tajomstvo zostáva v hre, kým nie je vykonaná jeho spúšťacia akcia alebo ho neodstráni nejaká špeciálna karta. Po aktivácii nenávratne mizne z hry. Spoločnou

vlastnosťou tajomstiev je, že môžu byť aktivované vždy len počas ťahu protivníka ich majiteľa.

Karta „Avenge“ je tajomstvom, ktoré stojí jednu jednotku many (pozri obrázok 2.10). Jeho efekt je zvýšenie útoku náhodnej priateľskej postavičky o tri a jej životov o dva. Tento efekt sa prejaví po tom, ako zomrie jedna z priateľských postavičiek. Pri používaní tohto konkrétneho tajomstva je potrebné mať na pamäti, že efekt každého tajomstva sa môže prejaviť len počas ťahu súpera. Dôsledkom toho je, že ak zomrie jedna z priateľských postavičiek počas ťahu vlastníka tohto tajomstva, efekt tajomstva sa neprejaví.

Obrázok 2.10: Tajomstvo 1, zdroj: [O2]

Ďalším tajomstvom je karta “Duplicate” (pozri obrázok 2.11). Stojí tri jednotky many. Spôsobuje, že ak zomrie priateľská postavička, na ruku hráča sa vygenerujú jej dve kópie.

Obrázok 2.11: Tajomstvo 2, zdroj: [O2]

2.3 Vlastnosti prostredia

Hearthstone poskytuje z pohľadu umelej inteligencie zaujímavé prostredie pre vývoj agentov. Jeho vlastnosti sú:

- Čiastočná pozorovateľnosť. Hráč nevidí súperove karty a tajomstvá.
- Čiastočná stochastickosť. Niektoré akcie majú náhodný dopad (niektoré karty majú náhodnosť priamo v textovom popise, pozri obrázok 2.4).
- Sekvenčnosť. Priebeh hry sa síce dá rozdeliť do epizód po jednej akcii, ale tie na sebe nie sú nezávislé.
- Diskrétnosť.
- Jedná sa o hru dvoch hráčov s nulovým súčtom, keďže každý efekt v hre, ktorý je prospešný pre jedného z hráčov, škodí jeho súperovi.

3 Prístupy umelej inteligencie

3.1 Náhodný hráč

Pre účely testovania úspešnosti agentov je nutné poskytnúť agenta s nejakým jednoduchým východiskovým chovaním. Túto úlohu často vykonáva nejaký náhodne hrajúci hráč. Jeho základnou črtou je, že vykonáva náhodne zvolené akcie. V prípade hry Hearthstone existuje viacero možností, ako presne tohto hráča zadefinovať, tieto možnosti rozoberieme v šiestej kapitole, ktorá je popisom chovania testovacích agentov.

Dôležitou súčasťou hry je výber balíčku kariet. Tento výber má svoj zmysel, rôzne karty prinášajú rôzne možnosti, a teda aj rôznorodú kvalitu. Náhodný hráč je vhodným agentom na základné porovnanie kvality dvoch balíčkov.

3.2 Súvisiace práce

Existujú dve zverejnené práce, ktoré sa venujú výskumu umelej inteligencie v hre Hearthstone.

Prvou z nich je práca autora Markus Zopf (2), ktorej účelom bolo porovnanie úspešnosti dvoch algoritmov v tejto hre. Porovnávané boli algoritmy Monte Carlo Tree Search a Monte Carlo bandit approach. Testované boli proti náhodnému hráčovi a proti sebe navzájom. Úspešnosti algoritmov presiahli 90 %. V práci bol pre testovanie používaný náhodne vygenerovaný balíček pre každú hru.

Ďalšou dostupnou publikáciou je práca autora David Taralla (1), ktorej cieľom bola aplikácia strojového učenia na hru Hearthstone. Autori vytvorili Extremely Randomized Trees Binary Classifier. Bol testovaný proti dvom typom hráčov. Prvým z nich bol náhodný hráč. Ďalší hráč je v publikácii nazvaný „Scripted player“ a jeho chovanie nie je detailne popísané. Vieme ale, že jeho účelom je aproximovať stredne dobre hrajúceho hráča, a takisto je v práci popísaný balíček, ktorý používal (pozri prílohu D)). Keďže súčasťou práce sú agenti, ktorých by sa pre svoje chovanie tiež dali nazvať „Scripted players“, premenovali sme si tohto „Scripted player“ pre potreby tejto práce na Priemerného hráča. Extremely Randomized Trees Binary Classifier dosiahol úspešnosť 93 % proti náhodnému hráčovi a 10 % proti Priemernému hráčovi. Náhodný hráč v tejto práci presne zodpovedá nášmu Náhodne akčnému hráčovi (popísaný v podkapitole 6.2 nižšie). Keďže v práci chýba popis chovania ich

Priemerného hráča, nebolo možné toto chovanie zreplikovať a porovnať s ním výsledky.

3.3 Výber algoritmu

Spomínané publikácie zohrali rolu aj pri výbere algoritmu, ktorý sme použili na vývoj umelého hráča. V prvom rade nás ovplyvnili spôsobom, že redukovali počet našich možností, pretože by nemalo zmysel opakovať už použité postupy. V oboch prácach autori odporučili v budúcich prácach použiť spätnoväzobné učenie.

Proces výberu akcií v hre Hearthstone dobre zodpovedá Markovovmu rozhodovaciemu procesu. V každej časovej jednotke je hra v nejakom stave s. Tento stav môže zmeniť hráč vykonaním nejakej akcie. Vyberať môže z akcií, ktoré sú dostupné v stave s. Proces na akciu a odpovedá presunom do stavu s' a dáva hráčovi odmenu $R_a(s, s')$.

Vzhľadom na tieto vlastnosti hry Hearthstone a na odkazy na spätnoväzobné učenie v súvisiacich publikáciách sme sa rozhodli pre použitie Q-učenia. Táto voľba, ako aj povaha celej práce, nás donútila zvoliť hráčov, pomocou ktorých sme nášho agenta učili, a na ktorých sme evaluovali výsledky. Logické východiskové chovanie poskytuje náhodný hráč. Jeho nevýhoda je, že sa chová inak, ako hráči reálne hrajú. Ideálnymi trénovacími entitami z hľadiska reprezentácie reálnej situácie a trendov v hre by teda boli reálni hráči. Hry s reálnymi hráčmi bohužiaľ trvajú príliš dlho na to, aby bolo možné v reálnom čase učiť nášho agenta. Našťastie existujú webové portály ako (3) a (4), ktoré priebežne zaznamenávajú trendy a situácie v hre Hearthstone. Poskytujú taktiež návody, fóra a diskusie, kde sú rozoberané herné stratégie a zloženie balíčkov. Na základe týchto prameňov informácií sme sa pokúsili vytvoriť umelých hráčov, ktorí dobre aproximujú súčasné trendy a hrané taktiky v hre Hearthstone.

4 Q-učenie

4.1 Algoritmus

Q-učenie je technika spätnoväzobného učenia, ktorá je nezávislá na modeli. Podľa (5) dokáže nájsť optimálne správanie pre každý konečný Markov rozhodovací proces. Toto správanie je optimálne v zmysle, že stredná hodnota totálnej odmeny cez všetky nadväzujúce kroky je najväčšia, akú sa dá dosiahnuť. Tento algoritmus funguje na princípe funkcie $Q : S \times A \rightarrow \mathbb{R}$, kde S je množina stavov a A je množina akcií. Agent sa učí chovanie postupným vykonávaním akcií a systémov dostávania a propagácie odmien za ťahy. Jednoduchosť použitia agenta s už naučeným chovaním spočíva v tom, že najlepšou akciou je tá akcia, ktorá má najväčšiu Q-hodnotu v danom stave.

Pred začatím učenia vracia funkcia Q náhodnú pevnú hodnotu pre všetky dvojice $s \in S$ a $a \in A$. Funkcia je aktualizovaná po každom vykonaní akcie a následnom odmenení na základe pozorovania nového stavu. Práve iteratívne opakovanie tejto aktualizácie predstavuje základ algoritmu Q-učenia. Po každom vykonaní akcie sa funkcia Q aktualizuje spôsobom:

$$Q_{t+1}(s,a) \coloneqq Q_t(s,a) + \alpha(s,a) \cdot \left(R_{t+1} + \gamma \cdot \max_{a' \in A} Q_t(s',a') - Q_t(s,a) \right),$$

kde $Q_t(s,a)$ je hodnota funkcie Q v čase t pre stav s a akciu a. Zmenu tejto hodnoty ovplyvňujú najmä okamžitá odmena R_{t+1} a ako dobrý je stav, do ktorého sme sa dostali.

Okamžitú odmenu R_{t+1} agent dostane za vykonanie nejakej akcie. Výška tejto odmeny môže byť kladná, ale aj záporná, a to v prípade, že chceme agentovi prezentovať jeho voľbu ako nevhodnú.

Okrem odmeny sa pri aktualizácii funkcie Q zohľadňuje aj to, do akého stavu sme sa dostali. Keďže chceme vyberať akciu s najväčšou Q hodnotou, budeme tak robiť aj pri učení.

Parameter ovplyvňujúci rýchlosť učenia α určuje, nakoľko novo získané hodnoty prepisujú doteraz naučené hodnoty. Jeho hodnota sa pohybuje v rozsahu $(0, 1]$. Ak by mal hodnotu 0, neučil by sa vôbec. V prípade, že sa tento parameter rovná jednotke, bude mať naposledy naučená hodnota najväčší vplyv. Táto hodnota je ideálna v úplne deterministických prostrediach.

Znižovací faktor γ rozhoduje o dôležitosti budúcich odmien. Naberá hodnoty z rozsahu $[0,1]$. Použitie hodnoty väčšej alebo rovnej ako jedna spôsobuje riziko divergencie Q hodnôt. Čím menšia je jeho hodnota, tým väčšiu váhu pri učení majú okamžité odmeny. Naopak, väčšia hodnota tohto parametru spôsobuje väčšiu propagáciu hodnôt.

4.2 Q-učenie s tabuľkou

Najprirodzenejšou a zároveň implementačne najjednoduchšou formou reprezentácie funkcie Q je tabuľka (pozri tabuľku 4.1). Tabuľka je dvojdimenzionálna s rozmermi $|S| \times |A|$, $|S|$ je veľkosť množiny stavov a $|A|$ vyjadruje počet všetkých možných akcií. Hodnota na pozícii (s, a) udáva hodnotu funkcie Q pre stav s a akciu a. Výhodou tejto reprezentácie je presnosť. Použitie tabuľky garantuje prácu s hodnotami, ktoré sa algoritmus naučí. Ďalšou veľkou výhodou je rýchlosť. Aktualizácia funkcie Q spočíva v jednoduchom prepísaní jedného údaju v tabuľke. Tieto dôvody robia z tabuľky ideálnu dátovú štruktúru vždy, keď je to možné. V prípade veľkého množstva stavov alebo akcií nie je možné použiť tabuľku z pamäťových dôvodov.

Všeobecným riešením pamäťových problémov je aproximácia funkcie Q. Techník na aproximáciu funkcie je veľa, v prípade spätnoväzobného učenia je najpoužívanejšou neurónová sieť.

	$akcia_1$	$akcia_2$
$stav_1$	$Q(stav_1, akcia_1)$	$Q(stav_1, akcia_2)$
$stav_2$	$Q(stav_2, akcia_1)$	$Q(stav_2, akcia_2)$

Tabuľka 4.1: Q-učenie s tabuľkou

4.3 Q-učenie s neurónovou sieťou

Neurónová sieť (pozri obrázok 4.1) je cesta k eliminácii pamäťových problémov v spätnoväzobnom učení. Jej využitie spočíva v nahradení tabuľky, ktorá má veľké pamäťové nároky. Keďže Q je funkcia stavu a akcie, vstupné neuróny reprezentujú stav aj akciu. Zvyčajnou implementáciou je zakódovanie stavu do menšieho počtu neurónov. Neurónov reprezentujúcich akcie je toľko, koľko je akcií, a na vstupe sa vždy aktivuje jediný z nich, podľa toho, ktorá akcia bola vykonaná. Sieť má jediný

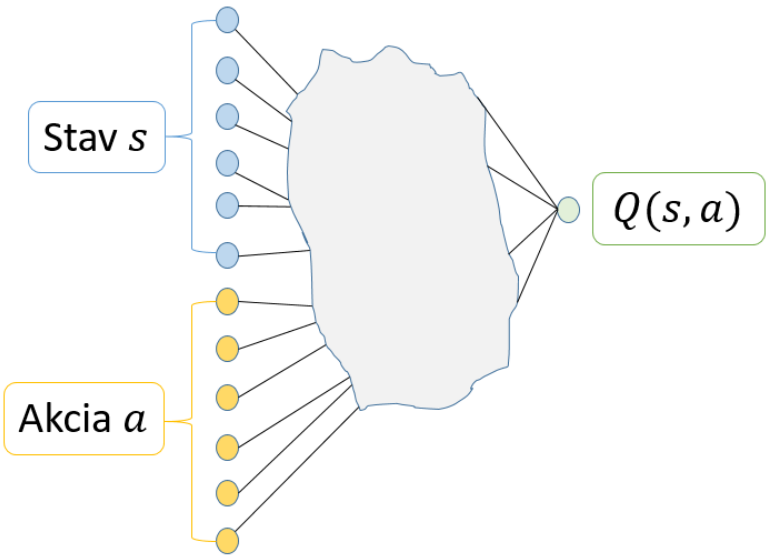
výstupný neurón, ktorého výstup reprezentuje hodnotu funkcie Q pri stave s a akcii a podľa vstupných hodnôt.

V porovnaní s tabuľkou je aktualizácia neurónovej siete zložitejšia. Vhodným postupom je využitie algoritmu back-propagation. Využíva sa väčšinou v prípadoch trénovacích vzorov na natrénovanie neurónovej siete, čo nie je prípad Q-učenia. To ale nie je prekážkou, chyba sa totiž propaguje po každom vykonaní akcie. Hodnota chyby je rozdiel novej a starej hodnoty funkcie Q v danom stave pre danú akciu.

Výber nasledujúceho ťahu prebieha principiálne rovnako ako v prípade tabuľky. Je potrebné zistiť, ktorá akcia je pre daný stav najvýnosnejšia. V praxi to znamená $|A|$ spustení neurónovej siete, kde A je množina akcií. Vykonaná je potom akcia, pri ktorej dá neurónová sieť pre daný stav najväčší výstup.

Alternatívne je možné neurónovú sieť navrhnúť tak, že jej vstupné neuróny reprezentujú len stav. V tom prípade sa namiesto jediného výstupného neurónu využíva toľko neurónov, koľko je akcií. Výstupom je vektor hodnôt funkcie Q pre všetky akcie v danom stave.

Neurónová sieť prináša výhodu ušetrenia priestoru, čo je pre vysoko dimenzionálne problémy nutnosť. Na druhej strane ale prináša aj nevýhody. Sieť ponúka len aproximáciu funkcie a nemôžeme sa na ňu spoliehať tak ako na tabuľku. Negatívne ovplyvnená je aj rýchlosť učenia. Back-propagation algoritmus je v porovnaní s prepísaním údaja v tabuľke výrazne časovo náročnejší.



Obrázok 4.1: Q-učenie s neurónovou sieťou

5 Simulátor

5.1 Účel

Cieľom tejto práce bolo vytvorenie umelej inteligencie pre hru Hearthstone. Kvôli možnosti implementácie a rýchleho hrania hier bol potrebný simulátor hry. Jednou z možností bolo naimplementovať vlastný simulátor. To ale nebolo účelom práce, a tak sme sa pokúsili nájsť projekt, ktorý by ponúkal vhodné prostredie pre vývoj umelej inteligencie.

Medzi existujúcimi simulátormi hry Hearthstone existujú minimálne dva, ktoré sa snažia ponúkať dobre uchopiteľné prostredie pre vývoj umelej inteligencie. Týmito simulátormi sú Hearthbreaker (6) a FirePlace (7). Zo začiatku sme chceli využiť Hearthbreaker. Dôvodom bolo, že podľa dokumentácie ponúkal lepšie zobrazovanie hier a táto dokumentácia bola spracovaná prehľadnejšie. Nanešťastie, tento projekt bol v čase vypracovania tejto práce v technických problémoch a nebol funkčný. Na rozdiel od neho, FirePlace fungoval a bol priebežne udržiavaný. To spôsobilo, že sme pre celý vývoj používali práve jeho.

FirePlace nám ponúkol možnosť odsimulovať ľubovoľný počet hier a obsahoval chovanie náhodného hráča. Výstup simulácií bol text popisujúci akcie hráčov.

5.2 Rozšírenia

Simulátor bol síce plne funkčný, keďže sa na ňom dali odsimulovať hry, ale na vývoj bol príliš neprehľadný a potreboval niekoľko rozšírení. Týmito rozšíreniami sa myslí čitateľnejší záznam hry a štatistický výstup simulácií pre evaluáciu výsledkov. Programátorskú a užívateľskú dokumentáciu je možné nájsť v prílohe H) aprílohe I). Zdrojový kód je voľne dostupný na (8).

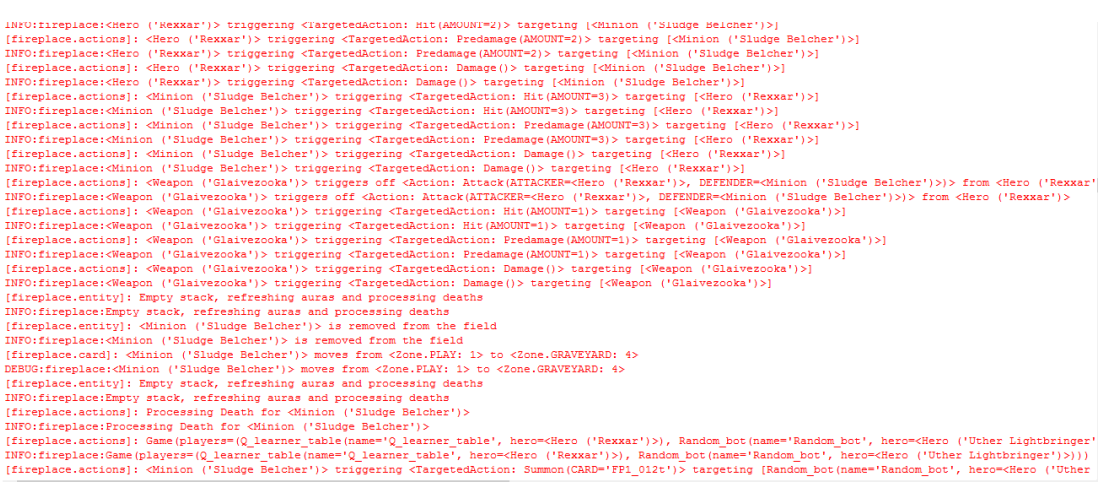
5.2.1 Záznamy hier

Jedným z hlavných nedostatkov simulátoru bolo, že dáva neprehľadný textový výstup (pozri obrázok 5.1). Z neho sa síce dal vyčítať celý priebeh hry, ale trvalo to príliš dlho a brzdilo to samotný vývoj. V tomto bode sme mali znova možnosť naprogramovať nejaký jednoduchý zobrazovač záznamov hier. Už druhýkrát sme sa rozhodli, že nemá zmysel vytvárať duplikáty, a snažili sme sa využiť to, čo už existuje. Hearthstone Deck Tracker je open-source projekt, ktorý ponúka veľa zaujímavých

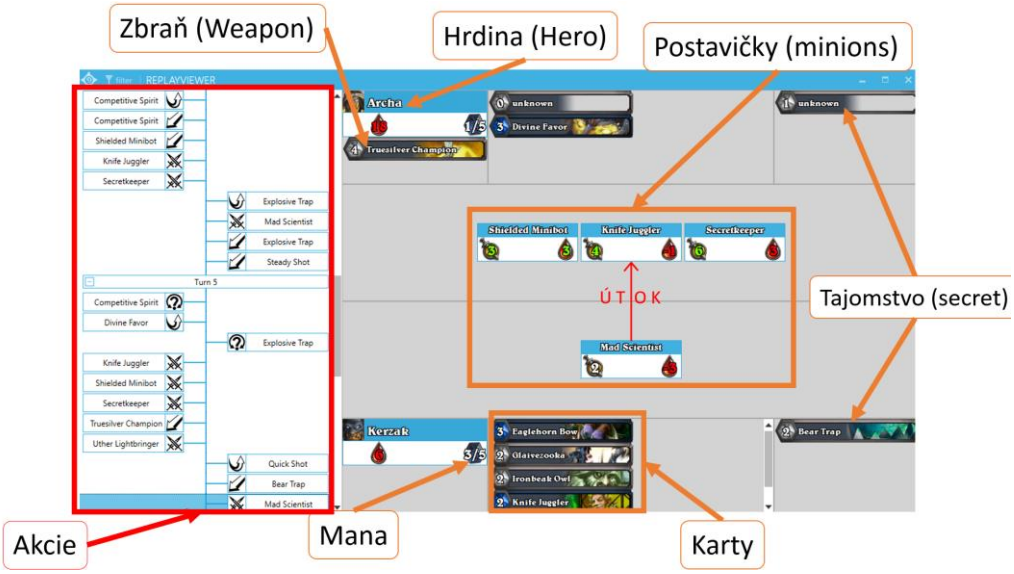
funkčností hráčom Hearthstonu, medzi inými aj vizualizáciu záznamov ich hier (pozri obrázok 5.2).

Aj vďaka tomu, že je tento projekt voľne dostupný, sme dokázali preskúmať, ako zobrazovanie funguje. Zistili sme, že Hearthstone Deck Tracker spracováva textový výstup, ktorý produkuje Hearthstone, a vytvára z neho súbor vo formáte JSON. Ten následne zabalí do formátu ZIP a premenuje jeho príponu na „.hdtreplay“. Kľúčovou úlohou bolo teda rozlúštiť to, ako daný JSON súbor kóduje priebeh hry. Po dôkladnom skúmaní sme prišli na to, že obsahuje jedno pole, do ktorého ukladá po každej akcii obraz stavu hry a informáciu o vykonanej akcii. Obraz stavu hry je tiež zoznamom, ktorého položky reprezentujú každú kartu, postavičku, tajomstvo či hrdinu v hre. Akcia je reprezentovaná typom a odkazom na entity, ktoré s ňou súvisia.

Na základe tohto skúmania sme rozšírili simulátor tak, aby produkoval súbory “.hdtreplay“, v ktorých sú zabalené súbory typu JSON. Tieto súbory sú zobraziteľné programom Hearthstone Deck Tracker v rovnakom formáte ako tie, ktoré produkuje on sám (pozri obrázok 5.2). Toto rozšírenie ponúka vývojárovi lepšiu predstavu o tom, ako sa agenti chovajú.



Obrázok 5.1: Textový výstup



Obrázok 5.2: Záznam hry (Hearthstone Deck Tracker), zdroj: [O3]

5.2.2 Štatistické súbory

Grafický záznam hry je vhodný vývojový nástroj na vizuálnu kontrolu toho, či hra bola odsimulovaná správne, ako aj toho, či hra agenta zodpovedá definovanej stratégii. Na väčšie štatistické evaluácie sme potrebovali výstup, ktorý bude spracovateľný štatistickými nástrojmi. Rozhodli sme sa, že na vyhodnocovanie budeme používať programovací jazyk R, pretože je to open-source projekt určený predovšetkým na štatistické výpočty a tvorbu grafov. Pre štatistické súbory sme vybrali formát „csv“.

Každá hra produkuje svoj štatistický záznam. Tento záznam obsahuje údaje o stave hry na začiatku každého kola. Všetky hry navyše po svojom ukončení produkujú súhrnný záznam o výsledku. Tento záznam predstavuje jeden riadok v súbore, ktorý tieto výsledky obsahuje. Vývojár má na výber, s akým súborom chce pracovať. Môže každú sériu simulácií ukladať do vlastného súboru, ale taktiež má možnosť dopĺňať výsledky do jediného súboru.

6 Testovací agenti

6.1 Motivácia

Doterajšie práce, ktoré sa zaoberali umelou inteligenciou v hre Hearthstone, využívali na tréning alebo vývoj väčšinou náhodne hrajúcich hráčov. Práca autora David Taralla (1) využíva taktiež agenta, ktorý hrá podľa definovaných pravidiel. Chovanie tohto hráča je v práci popísané tak, že reprezentuje priemerne hrajúceho hráča. Takýto popis nie je dostatočný na to, aby sa dal hodnoverne napodobniť. Vývoj umelej inteligencie pre Hearthstone by mal smerovať k agentovi schopnému porážať reálnych hráčov. Z tohto hľadiska by boli reálni hráči ideálnymi protivníkmi v priebehu tréningu aj v priebehu evaluácie. Okrem komplikovanejšej realizácie sme takýto postup zamietli aj preto, že reálne hry trvajú rádovo minúty. To znemožňuje flexibilný tréning s dostatočným množstvom hier.

V dobe vypracovávania tejto práce existovali taktiky, ktoré boli hrané veľkým množstvom hráčov (podľa (3) a (4), 15. 02. 2016). Tieto taktiky sa dajú popísať malým počtom (rádovo desiatkami) pravidiel, preto sme sa rozhodli, že ich v práci využijeme. Takýchto stratégií existuje veľa, my sme vybrali tri z nich a naimplementovali podľa nich agentov (pozri podkapitoly 6.5, 6.6 a 6.7). Našim cieľom bolo, aby sme mali k dispozícii nejakých hráčov chovajúcich sa podľa rozumne podložených pravidiel, a zároveň aby každý z nich používal iný štýl hry. Ich účelom nebolo reprezentovať najoptimálnejšie možné chovanie, ktoré myslí na každú okrajovú situáciu, ale vystihnúť podstatu potenciálneho spôsobu hrania hráča riadiaceho sa danými taktikami. Preto ani ich implementácia neošetruje obrovské množstvá možných situácií v hre, ale snaží sa vystihnúť kľúčové črty stratégie, ktorou sa riadia. Učenie a evaluácia na týchto agentoch má väčšiu výpovednú hodnotu o kvalite agenta, ako keby boli vykonávané len na náhodnom hráčovi.

Každý z troch testovacích agentov používal počas celého vývoja jediný balíček kariet, špecifický pre taktiku, ktorú agent nasledoval. Balíček bol vybraný podľa expertov (weby (3) a (4), 15. 02. 2016), ktorí okrem popisovania úspešných stratégii zostavovali najvhodnejšie balíčky pre tieto stratégie. Pevný balíček priniesol možnosť naimplementovať veľmi špecifické chovanie, ktoré je garanciou kvality agenta.

6.2 Náhodne akčný hráč

Náhodný hráč je základným typom trénovacieho ako aj evaluačného agenta. Ide o jediného agenta, ktorý je porovnateľný naprieč rôznymi prácami, ak aj nemáme prístup k ich zdrojovým kódom daných prác. Ako sme už zmienili v podkapitole 3.1, v rámci hry Hearthstone existuje viacero možností, ako takéhoto hráča zostrojiť.

Prvý z prístupov, ktoré sme využili, bol použitý aj v prácach autorov David Taralla (1) a Markus Zopf (2). Jeho počínanie je priamočiare. Vykonáva náhodne zvolenú akciu z množiny všetkých akcií, ktoré môže v danom stave vykonávať. Toto opakuje, kým existuje akcia, ktorú môže vykonať, následne ukončí ťah. Je schopný hrať s ľubovoľným regulárnym balíčkom kariet. Tohto agenta budeme ďalej nazývať Náhodne akčný hráč.

6.3 Náhodný hráč

Náhodne akčný hráč je dobrým východiskovým bodom pre testovanie. Medzi akciami, ktoré náhodne vyberá, ale nemá zahrnutú akciu ukončenia kola. Intuitívne predpokladáme, že hráčovi sa vždy oplatí vykonať nejakú akciu v prípade, že ju môže vykonať, než neurobiť nič. Existujú ale herné stratégie (pozri podkapitolu 6.7), ktoré fungujú na princípe uchovávania si nejakej dobrej kombinácie kariet, ktorá vedie k víťazstvu. Náhodne akčný hráč implementuje chovanie, ktoré mu nedovolí uchovávať si karty, pretože vždy vykonáva akcie, kým môže. To bolo dôvodom toho, že sme sa rozhodli naimplementovať ďalší typ náhodne hrajúceho hráča, ktorý taktiež vykonáva náhodné akcie, ale tentoraz medzi nimi má aj ukončenie kola, ktoré teda niekedy môže byť predčasné. Tohto agenta budeme pre možnosť rozlíšenia od Náhodne akčného hráča nazývať Náhodný hráč.

6.4 Priemerný hráč

V práci autora David Taralla (1) použili na testovanie agenta, ktorý mal reprezentovať stredne dobre hrajúceho hráča. Autori bohužiaľ nepopísali jeho správanie nijak podrobnejšie. Bolo žiadúce, aby bolo možné testovať hráča proti agentovi, ktorý reprezentuje skutočných hráčov. Keďže nám neboli známe pravidlá, ktorými sa riadi Priemerný hráč, nemohli sme jeho chovanie skopírovať či napodobniť. Skutočnosť, že naši testovací agenti majú chovanie prispôsobené na mieru svojim pevným balíčkom, ich kvalitu veľmi zvyšuje. Preto sa dá predpokladať, že hrajú lepšie

ako stredne dobre hrajúci hráč, aj keď toto tvrdenie nie je možné overiť z časových dôvodov.

6.5 Face hunter

Jednou z najrozšírenejších taktík v hre Hearthstone je takzvaná „face“ taktika. Spočíva v maximalizácii útočenia na súperovho hrdinu. Táto taktika ignoruje všetko, čo jej nebráni útočiť na súperovho hrdinu.

Taktika je veľmi jednoduchá, na prvý pohľad až príliš. Pri kombinácii správnych kariet a hrdinu je však jednou z najúčinnejších. Je kombinovaná výlučne s jedným hrdinom, a tým je hunter. Dôvodom je, že práve hunter má na hranie tejto taktiky vhodné karty aj schopnosť. Jeho schopnosť je totiž ubrať dva životy súperovmu hrdinovi. Spotrebuje dve jednotky many ako každá iná schopnosť hrdinu. Existuje takisto veľa kariet, ktoré môže použiť len hunter, a sú vhodné pre túto taktiku, či už kvôli tomu, že sú to postavičky, ktoré majú „Charge“ (pozri podkapitolu 2.2.2), alebo kúzla spôsobujúce významné poškodenie.

Face hunter sa v každom kole snaží hrať karty tak, aby využil čo najviac jednotiek many. Všetky postavičky útočia primárne na súperovho hrdinu. Ak to nie je možné, útočia na postavičky s vlastnosťou „Taunt“, čím si opäť vytvárajú priestor na útočenie na hrdinu. Všetky kúzla, ktoré spôsobujú poškodenie, mieria rovnako na protivníkovho hrdinu.

Okrem tohto chovania, ktoré je základom „face“ taktiky, sme agenta vycibrili niekoľkými vylepšeniami. Pevný balíček (pozri prílohu A)) nám umožnil zostrojiť preferencie pre isté karty, dokonca nám umožnil tieto preferencie prispôsobiť stavu hry. Jeho presné chovanie popisuje príloha E).

6.6 Secret paladin

Hearthstone poskytuje príliš široké spektrum kariet na to, aby sa všetky oplatili hrať „face“ taktikou. V komunite hráčov je zaužívaným výrazom „výmena“ (trade). V širšom zmysle sa ním myslí strata jednej alebo viacerých kariet výmenou za elimináciu jednej alebo viacerých súperových kariet. Najčastejšie sa jedná o stratu a elimináciu postavičiek, no môže ísť aj o kúzla. Ak sa nejakej postavičke podarí útokom zničiť súperovu postavičku bez toho, aby útočiaca zomrela, hovoríme o výmene „2 za 1“. Dôvodom je, že súper bude pravdepodobne nútený použiť ďalšiu postavičku alebo kúzlo na zabitie danej útočiacej postavičky. Takáto výmena

predstavuje pre hráča výhodu, pretože počet kariet na ruke aj v balíčku je obmedzený, a vďaka tejto výmene hráč eliminuje dve súperove karty za cenu jednej svojej.

Taktika Secret paladin je reprezentantom štýlu hry, ktorý sa snaží využívať istú formu výmeny „2 za 1“. Svoje ciele si hráč vyberá tak, aby z toho mal čo najväčší prospech. Namiesto bezhlavého útočenia na hrdinu hráč útočí aj na postavičky, ak sa mu to oplatí. To, či sa mu to oplatí, posudzuje hráč individuálne. Jeho ústrednou kartou je „Mysterious challenger“, ktorý pri zahraní presunie jedno z každého druhu tajomstiev v balíčku do čakacej fázy. Preto má v balíčku (pozri prílohu B)) veľa tajomstiev.

Rozhodnutie, ktoré karty zahrá, závisí na konkrétnej hernej situácii. Podobne ako Face hunter má definované skóre pre jednotlivé karty, ktoré sa menia podľa stavu hry. Nesnaží sa primárne priamo využiť maximálne množstvo many, snaží sa o to, aby celkové skóre kariet zahraných za kolo bolo čo najväčšie.

V momente, keď sa agent rozhodne, že už v tomto kole nechce zahrať žiadnu kartu, začne útočiť svojimi postavičkami. Rekurzívne si sám pre seba spočíta všetky možné scenáre a vyberie najlepší z nich. Na výber najlepšieho potrebuje ohodnotenie stavu hry. Pre tento účel sme skonštruovali funkciu (pozri prílohu F)), ktorá lineárne rastie s počtom zničených útokov nepriateľa a s menšou váhou aj s počtom znížených životov nepriateľa. Klesá lineárne s počtom stratených útokov. Toto vyhľadávanie je možné interpretovať ako snahu udržať si prevahu útoku na hracej ploche. Sekundárne spôsobuje útočenie postavičiek na hrdinu. To sa deje v prípade, že agent vyhodnotí, že by útokmi na postavičku znížil prevahu útoku.

6.7 Malygos freeze mage

Posledný z trojice testovacích agentov je reprezentantom skupiny taktík, ktorá sa dajú nazvať „combo“. Charakteristickým znakom tejto skupiny je deštruktívnosť. Na rozdiel od taktík podobných tej, ktorú hrá Secret paladin, sa nesnaží udržať si prevahu v hernom poli, ale zabrániť získaniu prevahy súperovi. Okrem toho si na ruke zbiera špecifickú kombináciu kariet. Táto kombinácia je veľmi často schopná vyhrať hru za jediné kolo. Úspešnosť tejto taktiky závisí na schopnosti udržať sa nažive dostatočne dlho. Ak sa to hráčovi alebo agentovi podarí dovtedy, kým je jeho kľúčová kombinácia použiteľná, vo väčšine prípadov vyhráva.

Finálna kombinácia tohto agenta (pozri obrázok 6.1) je tvorená štyrmi až šiestimi kartami. Obsahuje karty „Malygos“, „Emperor Thaurissan“, „Frostbolt“ a „Ice

Lance“. Jej použitie trvá dve kolá a dokáže ubrať súperovmu hrdinovi až 34 životov v druhom kole použitia. Počas čakania na túto kombináciu používa predovšetkým kúzla, ktoré plošne zamrazujú alebo ničia súperove postavičky. Ďalším typom kariet, ktoré používa, sú karty obnovujúce životy.

Zo všetkých testovacích agentov je práve Malygos freeze mage najviac zviazaný s jeho balíčkom (pozri prílohu C)). Aj v tomto prípade sme hranie jeho kariet naimplementovali pomocou priorizácie kariet na základe hernej situácie (pozri prílohu G)). Postavičky preferujú útoky na hrdinu, pretože o postavičky protivníka sa starajú kúzla s plošným účinkom.

Obrázok 6.1: Malygos kombo, zdroj: [O2]

6.8 Kvalita testovacích agentov

Testovacích agentov sme najprv otestovali proti sebe a proti obom náhodným hráčom (pozri tabuľku 6.1), aby sme získali prehľad o ich kvalite. Malygos freeze mage, Secret paladin a Face hunter boli testovaní so svojimi balíčkami, Náhodný hráč bol testovaný s balíčkami testovacích agentov a s balíčkom, ktorý bol použitý aj v práci autora David Taralla (1). Pre každú dvojicu týchto variantov bolo odsimulovaných tisíc hier.

Prvým pozorovaním je, že všetci testovací agenti dokázali porážať oba typy náhodných hráčov. Face hunter a Secret paladin dokonca proti nim dosiahli úspešnosť aspoň 93 %, nech už mali ktorýkoľvek balíček. Proti Náhodnému hráčovi dosiahol Malygos freeze mage rovnako vysokú úspešnosť, avšak Náhodne akčného hráča dokázal Malygos freeze mage porážať, len ak Náhodne akčný hráč hral s balíčkom Malygos (pozri prílohu C)) alebo s balíčkom Liège (pozri prílohu D)). V prípade, že Náhodne akčný hráč hral s balíčkom Face (pozri prílohu A)) alebo s balíčkom Secret (pozri prílohu B)), dokázal Malygos freeze mage zvíťaziť len v 64 %, resp. 61 % prípadov. To je spôsobené tým, že náhodné vykonávanie akcií môže viesť v prípade vhodného balíčku k relatívne dobrým výsledkom.

Zo všetkých testovacích agentov vyšiel z testovania jednoznačne najlepšie Face hunter (pozri tabuľku 6.1). Prekonal 90 % úspešnosť nielen proti Náhodnému, či Náhodne akčnému hráčovi, ale aj proti agentovi Malygos freeze mage dosiahol 94 % úspešnosť. Jediným agentom, ktorý mu dokázal konkurovať, bol Secret paladin, proti ktorému dosiahol 70 % úspešnosť. Ďalším výsledkom potvrdzujúcim dominanciu agenta Face hunter je skutočnosť, že takmer proti každému oponentovi dokázal byť najúspešnejším zo všetkých agentov. Jediným prípadom, v ktorom ho niekto dokázal prekonať, bolo hranie proti Náhodne akčnému hráčovi s balíčkom Face. Proti nemu dosiahol 93 % úspešnosť, zatiaľ čo úspešnosť agenta Secret paladin sa vyšplhala až na 94 %. Okrem tejto výnimky je Face hunter najúspešnejším agentom proti svojim súperom, čo z neho robí najlepšieho testovacieho agenta.

Druhým najúspešnejším agentom je Secret paladin. Jediným agentom, ktorý ho dokázal poraziť, bol Face hunter. V porovaní s ním dosiahol aj výrazne nižšiu úspešnosť proti agentovi Malygos freeze mage. Táto úspešnosť je 74 %, a teda vyznieva stále jednoznačne v prospech Secret paladina.

Malygos freeze mage je síce najhorším z trojice testovacích agentov, ale je stále lepší ako Náhodný hráč. To, že je pre testovanie relevantným, dokazuje aj fakt, že proti Secret paladinovi dosiahol 26 % úspešnosť, čo je výrazne lepší výsledok ako Náhodný hráč.

Agent		Náhodný hráč				Náhodne akčný hráč				Face hunter	Secret paladin	Malygos freeze mage	Priemerný hráč
	Balíček	Face	Secret	Malygos	Liège	Face	Secret	Malygos	Liège	Face	Secret	Malygos	Liège
Náhodný hráč	Face		0,25	0,77	0,60	0,10	0,05	0,51	0,26	0,01	0,01	0,02	
	Secret	0,75		0,95	0,80	0,30	0,12	0,85	0,50	0,01	0,01	0,03	
	Malygos	0,23	0,05		0,25	0,03	0,01	0,39	0,10	0,00	0,00	0,01	
	Liège	0,40	0,20	0,75		0,05	0,03	0,49	0,16	0,00	0,01	0,01	
Náhodne akčný hráč	Face	0,90	0,70	0,97	0,95		0,26	0,90	0,77	0,07	0,06	0,36	
	Secret	0,95	0,88	0,99	0,97	0,74		0,99	0,89	0,06	0,06	0,39	
	Malygos	0,49	0,15	0,61	0,51	0,10	0,01		0,29	0,01	0,01	0,03	
	Liège	0,74	0,50	0,90	0,84	0,23	0,11	0,71		0,01	0,01	0,04	0,01
Face hunter	Face	0,99	0,99	1,00	1,00	0,93	0,94	0,99	0,99		0,70	0,94	
Secret paladin	Secret	0,99	0,99	1,00	0,99	0,94	0,94	0,99	0,99	0,30		0,74	
Malygos freeze mage	Malygos	0,98	0,97	0,99	0,99	0,64	0,61	0,97	0,96	0,06	0,26		
Priemerný hráč	Liège								0,99				

Tabuľka 6.1: Výsledky testovacích agentov.

Tabuľka zobrazuje úspešnosť jednotlivých testovacích agentov proti sebe navzájom ako aj proti obom náhodným hráčom. Vnútorné políčka tabuľky ukazujú úspešnosť agenta s balíčkom, ktorého určuje riadok proti agentovi s balíčkom, ktorého definuje stĺpec. Hlavná diagonála je začiernená, pretože nemalo zmysel jednotlivých agentov testovať proti sebe samým. Červeným písmom je vyznačený agent (pozri podkapitolu 6.4) a balíček z práce autora David Taralla (1). Výsledky v červenom riadku (a teda aj stĺpci) boli dosiahnuté práve v tejto práci ich Priemerným hráčom (pozri podkapitolu 6.4). Všetky dvojice boli testované na 1000 hrách.

6.9 Sila balíčku

Zaujímavé výsledky prinieslo taktiež testovanie dvoch Náhodne akčných hráčov proti sebe (pozri tabuľku 6.1). Prvým z nich je predpokladané tvrdenie, že na balíčku záleží. Dôkazom je, že Náhodne akčný hráč dokázal dosiahnuť s balíčkom Face úspešnosť 77 % proti Náhodnému akčnému hráčovi s balíčkom Liège, a dokonca až 90 % proti Náhodne akčnému hráčovi s balíčkom Malygos. Ak Náhodne akčný hráč dostal balíček Secret, dokázal proti týmto dvom balíčkom dosiahnuť ešte vyššiu úspešnosť, a to 89 % v prípade balíčku Liège a 99 % v prípade balíčku Malygos (pozri tabuľku 6.1).

Ďalším zaujímavým pozorovaním je to, že úspešnosť balíčku Face proti balíčku Secret dosiahla len 26 % v prípade, že boli hrané Náhodne akčným hráčom. Pokiaľ ale boli hrané proti sebe agentmi Face hunter a Secret paladin, úspešnosť sa preklopila a nadobudla hodnotu 70 % v prospech balíčku Face (pozri tabuľku 6.1). Plynie z toho záver, že výber balíčku nie je jediným faktorom úspešnosti agenta.

Tieto dva závery v praxi znamenajú, že má zmysel skúmať tak výber balíčku, ako aj samotnú hru. V tejto práci sme výber balíčku neskúmali, ale na základe týchto výsledkov vieme, že je nutné zohľadňovať, s akým balíčkom agenti hrajú.

V práci autora David Taralla (1) použili balíček Liège (pozri prílohu D)) Z výsledkov vyplýva, že tento balíček dosahuje jednoznačne slabšie výsledky [1] ako balíčky Secret (pozri prílohu B)) a Face (pozri prílohu A)). Náhodný hráč s balíčkom Liège prehral Náhodným hráčom s balíčkom Face v 60 % prípadov, s balíčkom Secret dokonca v 80 % prípadov. Náhodne akčný hráč s balíčkom Face dosiahol proti Náhodne akčnému hráčovi s balíčkom Liège 77 % úspešnosť a s balíčkom Secret proti tomu s balíčkom Liège až 89 % úspešnosť. Z týchto úspešností vyplýva, že porážať agentov s balíčkom Face alebo Secret by malo byť ťažšie ako tých s balíčkom Liège.

[1] To platí len pre náhodných hráčov, dobrá stratégia by mohla úspešnosť výrazne zvýšiť.

7 Aplikácia Q-učenia s neurónovou sieťou

7.1 Stavy

Stav hry je komplexný popis hry, ktorý sa skladá zo stavov jednotlivých entít v hre. Tieto entity sú karty na ruke, postavičky, hrdinovia, mana, tajomstvá a zbrane. Týchto stavov je potenciálne nekonečne veľa, pretože niektoré hodnoty sú teoreticky neobmedzené. Pri vyčísľovaní počtu stavov sme však zanedbali stavy, ktoré sa v hre vyskytujú veľmi zriedkavo, aby bolo možné spočítať veľkosť potrebnej tabuľky pre Q-učenie.

7.1.1 Stav kariet na ruke

Reprezentovať stav kariet na ruke sa môže zdať na prvý pohľad komplikované, pretože ich text môže určovať efekty, ktoré je ťažké popísať. V skutočnosti je jedinou potrebnou vecou množina jednoznačných identifikátorov kariet, ktoré máme na ruke. Dôvodom toho je, že nezáleží na ich poradí na ruke, a keďže identifikátor je pre každú kartu jedinečný, nie je potrebné nijak inak kódovať text či cenu karty. Q-agent by si totiž mal zapamätať účinky karty podľa jej identifikátoru.

Hráč nemusí mať na ruke žiadnu kartu, maximálne ich môže mať desať. Pre obrovský počet kariet v hre sme sa rozhodli predpokladať, že budeme brať do úvahy len karty z hráčovho balíčku. Keďže je v balíčku tridsať kariet, počet stavov kariet na ruke je

$$\sum_{i=0}^{10} \binom{30}{i} = 53\,009\,102.$$

7.1.2 Stav postavičiek

Aj pri postavičkách platí, že Q-agent by sa mal naučiť efekty popísané v texte na základe ich identifikátoru. Lenže to nestačí, pretože stav postavičiek sa môže meniť, na rozdiel od statického stavu kariet na ruke, a tieto zmeny už identifikátor nedokáže zachytiť. Postavičkám sa môže meniť veľkosť útoku a počet životov. Okrem toho ich stav ovplyvňuje to, či majú alebo nemajú Taunt a Divine shield (definícia v podkapitole 2.2.2) a či môžu útočiť alebo nie. Poslednou premenlivou vlastnosťou postavičky je, či platí, že je tzv. „Silenced“. Postavička, ktorá je Silenced, stráca všetky efekty vyplývajúce z textu, Taunt aj Divine shield.

Keďže veľkosť útoku a počet životov postavičky sú teoreticky neobmedzené, počet jej stavov je potenciálne nekonečný. Vzhľadom na reálnu situáciu v hre sme určili, že ak životy alebo útok presiahnu hodnotu deväť, budeme to považovať vždy za rovnakú situáciu.

Obmedzený počet stavov pre jednu postavičku je $52\,800$ (pozri tabuľku 7.1). Na hracej ploche sa môže nachádzať až štrnásť postavičiek. Na rozdiel od kariet na ruke môže mať ich vzájomná poloha význam pre hru. Preto je celkový počet stavov postavičiek $52\,801^{14} = 1{,}30915198 \cdot 10^{66}$.

Použili sme hodnotu $52\,801$ namiesto $52\,800$, pretože na nejakých miestach sa nemusí vyskytovať žiadna postavička.

	Hodnoty	**Obmedzené hodnoty**	**Počet obmedzených hodnôt**
Identifikátor	$[1{,}30]$	$[1{,}30]$	30
Počet životov	$[1, \infty]$	$[1{,}10]$	10
Veľkosť útoku	$[0, \infty]$	$[0{,}10]$	11
Taunt	$\{Áno, Nie\}$	$\{Áno, Nie\}$	2
Divine Shield	$\{Áno, Nie\}$	$\{Áno, Nie\}$	2
Silenced	$\{Áno, Nie\}$	$\{Áno, Nie\}$	2
Môže útočiť	$\{Áno, Nie\}$	$\{Áno, Nie\}$	2
Spolu			52 800

Tabuľka 7.1: Stavy postavičky

Ľavý stĺpec uvádza parametre ovplyvňujúce stav postavičky. V stĺpci „Hodnoty“ je potenciálny rozsah hodnôt týchto parametrov. V stĺpci „Obmedzené hodnoty“ je obmedzený rozsah týchto hodnôt na základe reálneho výskytu v hre a stĺpec „Počet obmedzených hodnôt“ uvádza ich počet.

7.1.3 Stav hrdinov

V hre sú vždy dvaja hrdinovia. Stav každého z nich určuje identifikátor, počet životov, veľkosť útoku a veľkosť brnenia. Potenciálne nekonečnými hodnotami sú v prípade hrdinu veľkosť útoku a veľkosť brnenia. Obe sme obmedzili hodnotou desať.

Po obmedzení počtu stavov je ich počet pre jedného hrdinu $10\,890$ (pozri tabuľku 7.2), a teda pre oboch je počet stavov $21\,780^2 = 474\,638\,400$.

	Hodnoty	**Obmedzené hodnoty**	**Počet obmedzených hodnôt**
Identifikátor	$[1,9]$	$[1,9]$	9
Počet životov	$[1, \infty]$	$[1,10]$	10
Veľkosť útoku	$[0, \infty]$	$[0,10]$	11
Veľkosť brnenia	$[0, \infty]$	$[0,10]$	11
Môže útočiť	$\{Áno, Nie\}$	$\{Áno, Nie\}$	2
Spolu			21 780

Tabuľka 7.2: Stavy hrdinu

Ľavý stĺpec uvádza parametre ovplyvňujúce stav hrdinu. V stĺpci „Hodnoty“ je potenciálny rozsah hodnôt týchto parametrov. V stĺpci „Obmedzené hodnoty“ je obmedzený rozsah týchto hodnôt na základe reálneho výskytu v hre a stĺpec „Počet obmedzených hodnôt“ uvádza ich počet.

7.1.4 Stav zbraní

Stav zbrane určuje hlavne veľkosť jej útoku a výdrže. Vzhľadom na to, že niektoré zbrane majú textový popis, musíme brať do úvahy aj identifikátor zbrane. Maximálny útok zbrane sme obmedzili na hodnotu desať, maximálnu výdrž na hodnotu päť.

Počet stavov priateľskej zbrane je 1500 (pozri tabuľku 7.3). Pri nepriateľskej zbrani zanedbávame identifikátor, pretože nepriateľ môže mať aj neznáme zbrane. Preto je počet stavov nepriateľskej zbrane 50, a teda celkový počet stavov zbraní je $1500 \cdot 50 = 75\,000$.

	Hodnoty	Obmedzené hodnoty	Počet obmedzených hodnôt
Identifikátor	[1,30]	[1,30]	30
Útok	[1,10]	[1,10]	10
Výdrž	[1,5]	[1,5]	5
Spolu			1 500

Tabuľka 7.3: Stav zbrane

Ľavý stĺpec uvádza parametre ovplyvňujúce stav zbrane. V stĺpci „Hodnoty“ je potenciálny rozsah hodnôt týchto parametrov. V stĺpci „Obmedzené hodnoty“ je obmedzený rozsah týchto hodnôt na základe reálneho výskytu v hre a stĺpec „Počet obmedzených hodnôt“ uvádza ich počet.

7.1.5 Stav tajomstiev

Stav priateľských tajomstiev sa dá vyjadriť podobne ako stav kariet na ruke. Na ich poradí nezáleží a ich efekt určuje identifikátor, ale na rozdiel od kariet na ruke ich môže byť teoreticky nekonečne veľa. Ich maximálny počet sme obmedzili na päť. Stav súperových tajomstiev sa dá vyjadriť len počtom (6 možností), keďže nemáme informácie o ich identifikátoroch. Celkový počet stavov tajomstiev je teda

$$6 \cdot \sum_{i=0}^{5} \binom{30}{i} = 6 \cdot 174\,437 = 1\,046\,622.$$

7.1.6 Ostatné stavy

Okrem stavov postavičiek, zbraní, tajomstiev, kariet na ruke a hrdinov, je celkový stav hry ovplyvňovaný aj počtom použiteľných jednotiek many, maximálnym počtom many v danom kole a počtom súperových kariet. Počet stavov týchto parametrov je 1100 (pozri tabuľku 7.4).

	Hodnoty	Obmedzené hodnoty	Počet obmedzených hodnôt
Mana	[1,10]	[1,10]	10
Maximálna mana	[1,10]	[1,10]	10
Súperove karty	[0,10]	[0,10]	11
Spolu			1 100

Tabuľka 7.4: Ostatné stavy

Ľavý stĺpec uvádza parametre ovplyvňujúce zvyšné stavy. V stĺpci „Hodnoty“ je potenciálny rozsah hodnôt týchto parametrov. V stĺpci „Obmedzené hodnoty“ je obmedzený rozsah týchto hodnôt na základe reálneho výskytu v hre a stĺpec „Počet obmedzených hodnôt“ uvádza ich počet.

7.1.7 Celkový počet stavov

Z predchádzajúcich výpočtov je možné jednoduchým prenásobením čiastkových počtov stavov získať ich celkový počet. Tento počet je
$53\,009\,102 \cdot\ 1{,}30915198 \cdot 10^{66} \cdot\ 474\,638\,400 \cdot\ 75\,000 \cdot\ 1\,046\,622 \cdot 1\,100 = 2.8441153 \cdot 10^{96}$.

Pre nášho agenta to znamená, že ak by používal pre učenie tabuľku, jeho tabuľka by musel mať pri tejto reprezentácii práve $2.8441153 * 10^{96}$ riadkov.

Tento počet je vysoký aj v porovnaní s inými všeobecne známymi hrami. Desiatkový logaritmus počtu stavov pre šach je 47, pre hru Go na ploche 13×13 je to 79. Ak by sme chceli nájsť hru s väčším počtom stavov, opäť nám môže poslúžiť hra Go, tentokrát na ploche 19×19. Vtedy je desiatkový logaritmus počtu jej stavov 170.

7.2 Akcie

Existujú tri základné typy akcií – zahranie karty, útok a použitie schopnosti hrdinu. Za akciu by sa dal považovať aj výber cieľa, ale zvolili sme prístup, v ktorom zahranie karty s iným cieľom predstavuje dve rôzne akcie, pretože výber cieľa je vždy spätý s nejakou inou akciou, ktorá jej bezprostredne predchádza.

7.2.1 Ciele

Každá akcia môže mať maximálne sedemnásť rôznych cieľov. Tými sú obaja hrdinovia, jedna zo siedmich priateľských alebo siedmich nepriateľských postavičiek, a keďže niektoré karty nemieria na nikoho, cieľ nemusí byť žiaden. Keďže druhov postavičiek je veľa, budeme ich rozlišovať podľa umiestnenia na hracej ploche. Môžeme tak robiť preto, lebo stavy zahŕňajú pozíciu jednotlivých postavičiek.

7.2.2 Hranie kariet

Pri definovaní akcie reprezentujúcej zahranie karty sme prihliadli na to, ako je reprezentovaný stav kariet. Keďže nezáleží na poradí kariet, rozhodli sme, že akcie kariet budeme rozlišovať podľa ich identifikátorov a nie podľa ich pozície na ruke. Okrem tridsiatich identifikátorov kariet v balíčku bolo nutné zaviesť aj akciu pre zahranie neznámej karty, pretože taká karta sa na ruke môže vyskytnúť. Vzhľadom na to, že každá karta môže mieriť na jeden zo sedemnástich vyššie definovaných cieľov, je počet akcií reprezentujúcich hranie kariet $31 \cdot 17 = 527$.

7.2.3 Útoky

Na rozdiel od hrania kariet na umiestnení postavičiek záleží. Preto bolo nutné definovať útočné akcie podobne ako vyberanie cieľa. Útočiť môže sedem priateľských postavičiek a priateľský hrdina. Mieriť môže len na polovicu cieľov, a tými sú tak súperov hrdina ako aj jeho postavičky. Počet útočných akcií je teda
$8 \cdot 8 = 64$.

7.2.4 Schopnosť hrdinu

Akciu schopnosti hrdinu je možné zadefinovať podobne ako akciu zahrania karty. Schopnosť má každý hrdina len jednu a počet jej cieľov je sedemnásť, pretože rôzne schopnosti hrdinov pokrývajú všetky možné ciele (pozri podkapitolu 7.2.1).

7.2.5 Celkový počet akcií

Z predchádzajúcich čiastkových výsledkom je možné získať celkový počet akcií ich priamočiarym sčítaním a pripočítaním jednotky, preto, že aj ukončenie kola je akcia. Celkový počet akcií je $527 + 64 + 17 + 1 = 609$.

7.3 Nepoužiteľnosť tabuľky

V predchádzajúcich podkapitolách (7.1 a 7.2) sme sa dopočítali k celkovým počtom stavov a akcií. Pri tejto podrobnej a kompletnej analýze sme dospeli k hodnotám $2.8441153 \cdot 10^{96}$ pre počet stavov a 609 pre počet akcií. V prípade, že by sme sa rozhodli naimplementovať Q-agenta s využitím tabuľky, táto tabuľka by musela mať rozmery $2.8441153 \cdot 10^{96} \times 609$, a teda obsahovať $2.8441153 \cdot 10^{96} \cdot 609 = 1.7320662 \cdot 10^{99}$ údajov. Ak by nám aj na každý údaj stačil jeden bajt, potrebovali by sme na uloženie tabuľky $1.7320662 * 10^{99}\ B = 1.57530503 \times 10^{87}\ TiB$.

Takýto počet je pri dnešných technologických možnostiach príliš veľký pre všetky dostupné pamäte. Tianhe-2, ktorý bol v roku 2015, podľa prieskumu TOP500 venujúcemu sa porovnávaniu superpočítačov, najlepší na svete, má pamäť 1375 TiB. Nehovoriac o tom, že bežné domáce počítače dosahujú úroveň pamäte rádovo v gigabajtoch.

Z dôvodu takého obrovského stavového priestoru sme boli nútení nejakým spôsobom zmeniť prístup. Namiesto tabuľky sme sa preto rozhodli pre použitie neurónovej siete kódujúcej stavový priestor, ktorú popíšeme v nasledujúcej podkapitole.

7.4 Kompozícia neurónovej siete

Rozhodnutie použiť pre učenie neurónovú sieť, ktoré padlo v podkapitole 7.3, prinieslo potrebu jej dizajnu. Našim cieľom bolo čo najvernejšie sa držať teórie popísanej v podkapitolách 7.1 a 7.2. Vďaka neurónovej sieti sme sa vo viacerých prípadoch nemuseli zaoberať reguláciou hodnôt, pretože tieto hodnoty boli zadávané neurónom ako vstupy, a teda potenciálne nekonečné množstvo hodnôt nebolo prekážkou. Popis jednotlivých zložiek neurónovej siete je popísaný v nasledujúcich podkapitolách.

7.4.1 Stavy

Množstvo stavov bol rozhodujúci dôvod, ktorý znemožnil zostrojenie tabuľky pre Q-učenie. Preto je to práve stavový priestor, kde je nutné ušetriť veľa pamäte.

Pre reprezentáciu stavu kariet bolo použitých desať vstupných neurónov. Tieto neuróny brali ako vstup identifikátor kariet na ruke. Keďže v prípade kariet na ruke nezáleží na poradí, ich identifikátory boli neurónovej vždy predávané vo vzostupnom

poradí, aby sa zmenšil celkový možný počet vstupov. V prípade, že agent mal na ruke menej ako desať kariet, neurónom, ktoré boli navyše, bola zadaná špeciálna vstupná hodnota -1.

Ďalších deväťdesiatosem neurónov reprezentovalo priateľské aj nepriateľské postavičky. Maximálny počet postavičiek na hracej ploche je štrnásť, každú z nich zastupovalo sedem vstupných neurónov. Prvé tri z týchto neurónov tvorili identifikátor postavičky, počet životov postavičky a útok postavičky. Ďalšie štyri naberali hodnoty 1 a 0, a to podľa toho, či platilo, že príslušná postavička mala Taunt, Divine shield, bola v stave Silenced a či mohla útočiť. Postavičky boli do vstupných neurónov siete zadávané v rovnakom poradí, v akom boli rozostavené na hracej ploche. Toto poradie bolo dôležité preto, že útoky a mierenie boli vyberané s ohľadom na umiestnenie postavičiek. V prípade neúplného počtu postavičiek na hracej ploche boli chýbajúce neuróny doplnené hodnotami -1 podobne ako chýbajúce karty.

Každého z hrdinov reprezentovalo päť neurónov reprezentujúcich identifikátor hrdinu, počet jeho životov, veľkosť jeho útoku, brnenia a to, či môže útočiť.

Priateľská zbraň bola reprezentovaná tromi neurónmi vyjadrujúcimi jej identifikátor, útok a výdrž. Súperovu reprezentovali len dva neuróny, oproti priateľskej sme vynechali identifikátor. V prípade absencie aspoň jednej zo zbraní boli použité hodnoty -1 ako obvykle.

Jedinou entitou na hracej ploche, ktorú sme obmedzili aj v prípade neurónovej siete, boli tajomstvá. Dôvodom bolo, že každé tajomstvo sme reprezentovali jedným neurónom, ktorý bral za vstup identifikátor daného tajomstva. Neurónov ale nie je možné mať nekonečne veľa, preto sme ich obmedzili na päť (ako inšpirácia poslúžila podkapitola 7.1.5). Súperove tajomstvá boli reprezentované jediným neurónom, ktorý bral za vstup ich počet, keďže to bola jediná informácia, ktorú o nich agent mal.

Táto reprezentácia spolu predstavuje 132 vstupných neurónov reprezentujúcich stav.

7.4.2 Akcie

Situácia s reprezentovaním akcií bola jednoduchšia, ako tomu bolo v prípade stavov, pretože celkový počet akcií je len 609 (pozri podkapitolu 7.2.5). Do neurónovej siete sme akcie aplikovali tak, že sme vytvorili neurón pre každú z nich (držali sme sa presne teórie popísanej v podkapitole 7.2). Tieto neuróny dostávali ako vstup nulu, jednotku dostal iba neurón reprezentujúci vykonanú akciu.

7.4.3 Vnútorné vrstvy

Komplikovanejšia vnútorná štruktúra siete zvyšuje časovú náročnosť jej aktualizácie. Z dôvodu tejto časovej náročnosti sme zvolili pomerne jednoduchú vnútornú štruktúru siete, s jedinou skrytou vrstvou obsahujúcou 800 vstupných neurónov.

7.4.4 Odmena

Počet spôsobov návrhu odmeňovacej funkcie je prakticky neobmedzený. My sme chceli začať jednoduchou stratégiou a v prípade veľkého úspechu skúšať zložitejšie.

Využili sme flexibilitu odmeňovacej funkcie, ktoré ponúka Q-učenie, a určili sme, že sa budeme snažiť agenta naučiť hrať podobnú taktiku, akou sa riadi náš najúspešnejší testovací agent, Face hunter. Bol odmeňovaný za útočenie na súperovho hrdinu, hranie kariet a využívanie many. Konkrétne dostával odmenu:

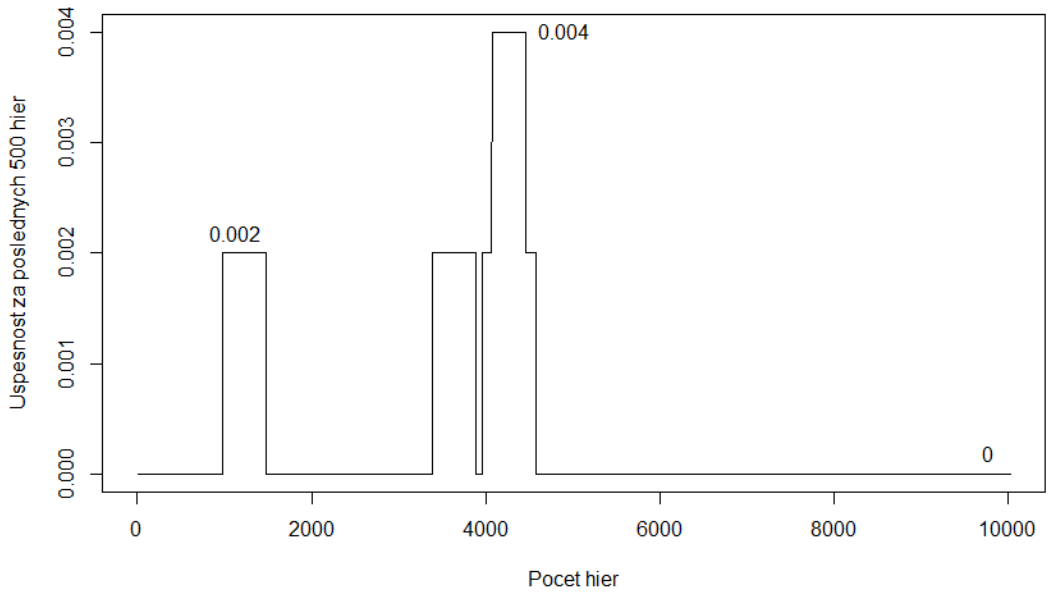
- $+10$ za útok na súperovho hrdinu alebo postavičku s Tauntom,
- $+10$ za zahranie karty,
- $+1$ za každú spôsobenú škodu súperovmu hrdinovi
- $+\frac{použitá\ mana}{maximálna\ mana} \cdot 10$ po ukončení ťahu, ak už agent nemohol zahrať žiadnu kartu ani ničím zaútočiť sa súperovho hrdinu,
- -10 za útok na postavičku bez Tauntu,
- -10 za spôsobenie škody priateľskému hrdinovi,
- -100 za ukončenie ťahu v prípade, že agent ešte mohol zahrať nejaké karty alebo niečím zaútočiť sa súperovho hrdinu.

Na aktualizáciu neurónovej siete bol použitý algoritmus back-propagation, ktorý aktualizoval váhy v neurónovej sieti na základe aktualizovanej Q-hodnoty pre vykonanú akciu.

7.4.5 Výsledky

S agentom využívajúcim neurónovú sieť z podkapitol 7.4.1, 7.4.2 a 7.4.3, ktorý bol odmeňovaný na základe funkcie popísanej v podkapitole 7.4.4, sa nám podarilo dosiahnuť len veľmi slabú úspešnosť. Po 10 000 simuláciách proti Face hunterovi agent dosiahol úspešnosť len 0,04 %, navyše počas posledných 5000 hier nevyhral ani raz (pozri obrázok 7.1). Z časových dôvodov nebolo možné preukázať, že táto úspešnosť by sa po oveľa väčšom počte simulácii nezvýšila.

Potrebovali sme nájsť príčinu zlyhania a znova nejakým spôsobom zmeniť prístup. Usúdili sme, že tento agent síce veľmi detailne a presne pokrýva priestor stavov a akcií hry, ale práve táto podrobnosť môže byť príčinou neúspechu. Skúsili sme preto zmenšiť stavový priestor a počet akcií. V nasledujúcej podkapitole je definovaný agent s redukovaným priestorom stavov aj akcií, navrhnutý tak, aby tieto priestory boli čo najmenšie, ale stále dostatočne veľké na to, aby sa agent dokázal naučiť chovanie požadované odmeňovacou funkciou (definovanou v podkapitole 7.4.4).



Obrázok 7.1: Učenie Q-agenta

7.5 Kompozícia redukovanej neurónovej siete

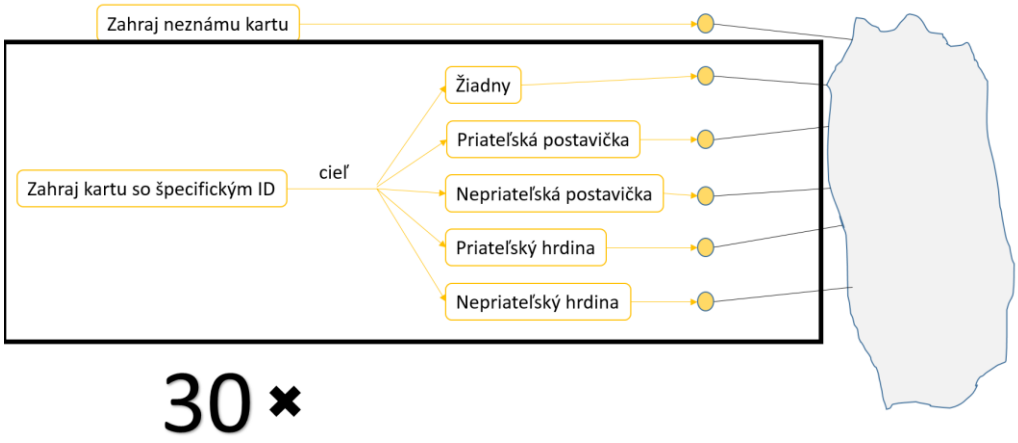
7.5.1 Redukované stavy

Ak by sme chceli slovne vyjadriť, aké chovanie nášho agenta sme chceli dosiahnuť, dalo by sa povedať, že sme chceli, aby čo naviac využíval manu a spôsobil súperovmu hrdinovi maximálne možné poškodenie. Pre účel maximalizácie poškodenia súperovho hrdinu sme nevyčlenili ani jeden neurón, pretože sme usúdili, že agent by sa mal naučiť mieriť na hrdinu bez ohľadu na stav. Využívať efektívne manu sa ale nemohol naučiť efektívne bez poznania stavu, preto sme určili jeden neurón za reprezentanta súčasného počtu jednotiek many a ďalších desať neurónov ako reprezentantov ceny kariet na ruke (pozri obrázok 7.2).

Obrázok 7.2: Redukované stavy

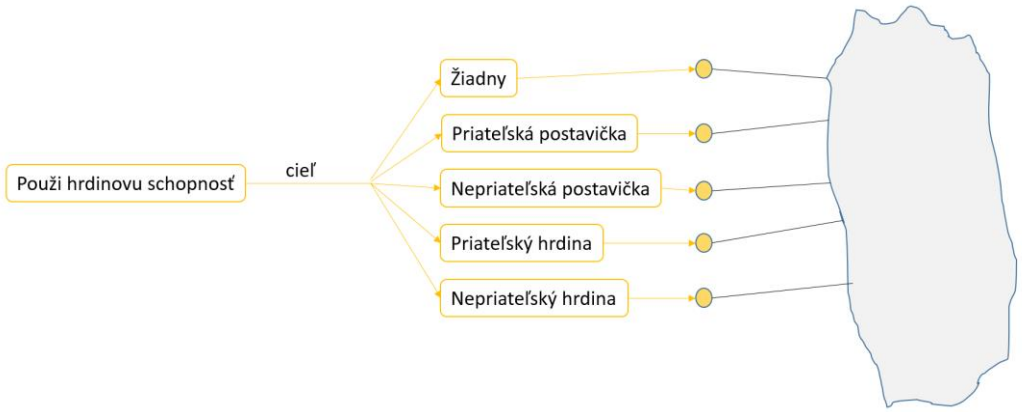
7.5.2 Redukované akcie

Pre hranie kariet sme zachovali prístup rozhodovania podľa identifikátoru karty. Dôvodom bolo, že to malo umožniť agentovi vytvoriť si akési preferencie pre karty, podobne ako to robia testovací agenti. Zmenšili sme však možný počet cieľov, čo bolo nutným dôsledkom redukcie stavov, ktoré už ďalej neposkytovali informáciu o umiestnení postavičiek. Týchto cieľov bolo päť, konkrétne priateľský hrdina, nepriateľský hrdina, priateľská postavička, nepriateľská postavička a pre potreby kariet bez cieľa aj žiaden cieľ. Tomuto zúženiu predchádzala úvaha, že veľké množstvo kariet s cieľom spôsobuje nejaký všeobecný nepodmienený pozitívny alebo negatívny efekt, a preto agentovi stačí rozlišovať týchto päť cieľov. V praxi týmto akciám zodpovedalo päť vstupných neurónov pre každú kartu z balíčku a pre kompletné pokrytie priestoru akcií, jeden ďalší vstupný neurón pre hranie neznámej karty. Spolu sme teda použili pre hranie kariet $30 * 5 + 1 = 151$ neurónov (pozri obrázok 7.3).



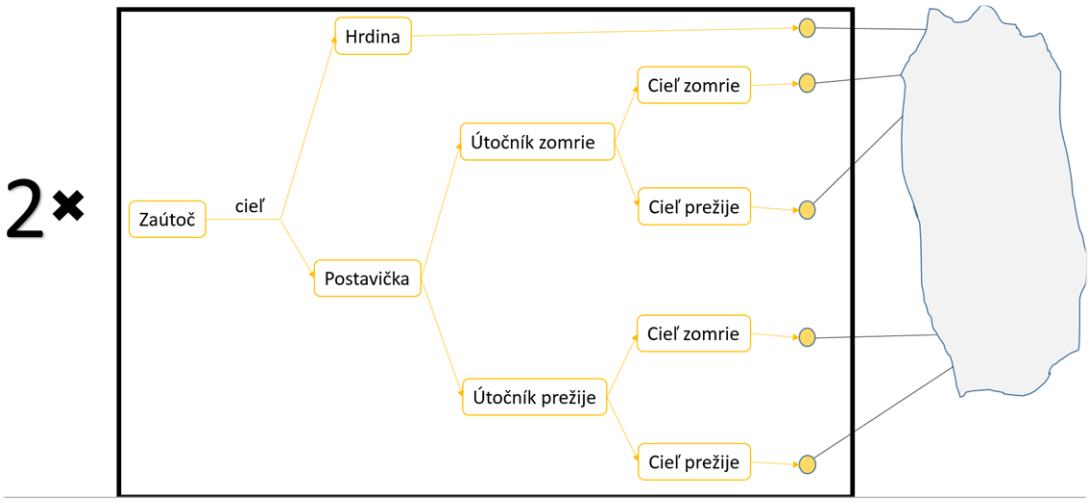
Obrázok 7.3: Redukované akcie hrania kariet

Obmedzili sme aj počet cieľov schopnosti hrdinu na rovnakých päť ako v prípade hrania kariet z rovnakých dôvodov, čo prakticky predstavovalo ďalších päť vstupných neurónov (pozri obrázok 7.4).



Obrázok 7.4: Redukované akcie schopnosti hrdinu

Útoky sme najprv rozdelili podľa toho, či je útočníkom hrdina alebo postavička. Dôvodom bolo, že útočenie často prináša aj stratu životov a životy hrdinu bývajú cennejšie ako životy postavičiek, preto sme chceli, aby to agent rozlišoval. Keďže sme už ciele v dôsledku redukcie stavov nemohli určovať na základe pozície, rozhodli sme sa ich rozdeliť primárne podľa toho, či išlo o hrdinu alebo postavičku, čo bolo nutnosťou pre maximalizáciu poškodenia hrdinu. Rozdelili sme ešte aj prípad útočenia na postavičku podľa toho, či sa predpokladalo, že útočník a cieľ prežijú. Účelom tohto delenia bolo prejsť na taktiku výmeny „2 za 1“ (pozri podkapitolu 6.6) v prípade nemožnosti útočiť na hrdinu. To spolu predstavuje päť vstupných neurónov pre prípad útočiaceho hrdinu na jeden z piatich cieľov a rovnako ďalších päť vstupných neurónov pre prípad útočiacej postavičky, spolu teda desať vstupných neurónov (pozri obrázok 7.5).



Obrázok 7.5: Redukované akcie útokov

Neurónová sieť mala teda spolu 166 vstupných neurónov na reprezentáciu akcií a 11 vstupných neurónov na reprezentáciu stavu, čo spolu dáva 177 vstupnúch neurónov.

7.5.3 Vnútorná štruktúra siete

Neurónovej sieti sme dali jedinú vnútornú vrstvu obsahujúcu 200 neurónov, pretože vysoká zložitosť neurónovej siete výraznou mierou zvyšovala časovú zložitosť výpočtov.

8 Aplikácia Q-učenia s tabuľkou

8.1 Motivácia

Zistenie, že je efektívnejšie použiť zjednodušený priestor stavov a akcií pre neurónovú sieť (pozri kapitolu 7), nás priviedlo k obnoveniu myšlienky použitia tabuľky pre učenie agenta. Redukovanú neurónovú sieť (pozri kapitolu 7.5) sme previedli s niekoľkými úpravami na tabuľku (pozri podkapitoly 8.2 a 8.3 nižšie).

8.2 Stavy

Ak by sme sa riadili priamo konštrukciou stavov v redukovanej neurónovej sieti (pozri podkapitolu 7.5.1), počet stavov by dosiahol hodnotu $11^{10} * 10$, pretože na ruke je možné mať nula až desať kariet ceny jedna až desať a počet jednotiek many sa môže pohybovať medzi jednou až desiatimi. Pre tento vysoký počet sme sa rozhodli obmedziť niektoré stavy (pozri tabuľku 8.1). Pre ceny šesť a vyššie sme zaviedli len dva stavy a to, či hráč má na ruke kartu s danou cenou alebo nie. Dôvodom je, že maximálny jednotiek many je desať, a teda hráč nemá možnosť využiť viac ako jednu z týchto kariet. Podobnou úvahou sme obmedzili počet kariet, ktoré stoja päť jednotiek many na dve, rovnako tie, ktoré stoja štyri. Pri kartách ceny tri je toto obmedzenie na počet tri a pri kartách s cenou dve jednotky many je obmedzenie päť. Kariet s cenou jedna je možné naozaj zahrať za kolo až desať, preto ich počet nie je obmedzený.

	Hodnoty	Obmedzené hodnoty	Počet obmedzených hodnôt
Mana	[1,10]	[1,10]	10
Počet kariet ceny 1	[1,10]	[0,10]	11
Počet kariet ceny 2	[0,10]	{0, 1, 2, 3, 4, (5 a viac)}	6
Počet kariet ceny 3	[0,10]	{0, 1, 2, (3 a viac)}	4
Počet kariet ceny 4	[0,10]	{0, 1, (2 a viac)}	3
Počet kariet ceny 5	[0,10]	{0, 1, (2 a viac)}	3
Počet kariet ceny 6	[0,10]	{0, (1 a viac)}	2
Počet kariet ceny 7	[0,10]	{0, (1 a viac)}	2
Počet kariet ceny 8	[0,10]	{0, (1 a viac)}	2
Počet kariet ceny 9	[0,10]	{0, (1 a viac)}	2
Počet kariet ceny 10	[0,10]	{0, (1 a viac)}	2
Spolu			760 320

Tabuľka 8.1: Stavy v tabuľke

Ľavý stĺpec uvádza parametre ovplyvňujúce stav cien kariet na ruke. V stĺpci „Hodnoty“ je potenciálny rozsah hodnôt týchto parametrov. V stĺpci „Obmedzené hodnoty“ je obmedzený rozsah týchto hodnôt na základe toho, ktoré hodnoty sú pre dobré využívanie many dôležité a stĺpec „Počet obmedzených hodnôt“ uvádza ich počet.

8.3 Akcie

Akcie v tabuľke presne zodpovedajú tým, ktoré boli použité pre redukovanú neurónovú sieť (pozri podkapitolu 7.5.2). Dôvodom je, že ich počet je len 166, čo je pre tabuľku veľmi prijateľný počet.

9 Výsledky a vyhodnotenie

V tejto práci sme zostrojili spolu pätnásť variantov umelých hráčov. Osem z nich predstavovali Náhodný a Náhodne akčný hráč (pozri podkapitoly 6.2 a 6.3) s balíčkami Face, Secret, Malygos a Liège (pozri prílohy A), B), C) a D)). Ako ďalších sme naimplementovali troch testovacích agentov (popísaní v podkapitolách 6.5, 6.6 a 6.7). Nakoniec sme vytvorili Q-agenta na štyri rôzne spôsoby. Jedna dvojica z nich sa učila pomocou neurónovej siete (pozri podkapitolu 7.5), ďalšia zas pomocou tabuľky (pozri kapitolu 8). V každej dvojici sa jeden z agentov učil s parametrami $\alpha = 0{,}8$ a $\gamma = 0{,}8$ a ďalší s parametrami $\alpha = 0{,}1$ a $\gamma = 0{,}01$.

Všetky varianty Q-agenta sa naučili porážať oba typy náhodných hráčov a testovacieho agenta Malygos freeze mage. Proti ostatným dvom testovacím agentom to už neplatilo, najbližšie k tomu bol tabuľkový agent s parametrami $\alpha = 0{,}1$ a $\gamma = 0{,}01$, ktorý dosiahol proti Secret paladinovi 42 % úspešnosť a proti Face hunterovi 26 % úspešnosť (pozri tabuľku 9.1). Úspešnosť proti Face hunterovi je najporovnateľnejším údajom s výsledkami v práci autora David Taralla (1), kde dosiahol so svojím prediktorom 10 % úspešnosť proti Priemernému hráčovi (pozri podkapitolu 6.4) s rovnakým balíčkom. Dôvodom dobrej porovnateľnosti je fakt, že obaja agenti hrajú v oboch prípadoch s rovnakým balíčkom a súperom je v oboch prípadoch naskriptovaný hráč.

Proti testovacím agentom dosiahol pri rovnakých parametroch agent s tabuľkou v piatich zo šiestich prípadov vyššiu úspešnosť ako ten s neurónovou sieťou. Výnimkou bolo testovanie s parametrami $\alpha = 0{,}8$ a $\gamma = 0{,}8$ proti Face hunterovi. Vtedy zvíťazil agent s tabuľkou len v 17 % prípadov, zatiaľ čo agent s neurónovou sieťou až v 20 % prípadov. Pri vzájomných súbojoch Q-agentov sa táto prevaha tabuľky nepotvrdila. Pri parametroch $\alpha = 0{,}1$ a $\gamma = 0{,}01$ síce bola úspešnejšia tabuľka v 56 %, ale pri parametroch $\alpha = 0{,}8$ a $\gamma = 0{,}8$ dosiahol agent s tabuľkou len 48 % úspešnosť (pozri tabuľku 9.2).

Z týchto výsledkov vyplýva, že prispôsobenie stavov, akcií a odmeňovacej funkcie Q-učenie viedlo k zvýšeniu konkurencieschopnosti Q-agentov proti testovacím agentom. Taktiež sa ukázalo, že výsledky s neurónovou sieťou príliš nezaostávajú za tými s tabuľkou.

				Náhodný hráč				Náhodne akčný hráč				Face hunter	Secret paladin	Malygos freeze mage	Priemerný hráč
				Balíček Face	Balíček Secret	Balíček Malygos	Balíček Liège	Balíček Face	Balíček Secret	Balíček Malygos	Balíček Liège	Balíček Face	Balíček Secret	Balíček Malygos	Balíček Liège
Q	Neurónová sieť	$\alpha = 0{,}8$ $\gamma = 0{,}8$	Balíček Face	0,96	0,97	0,93	0,99	0,72	0,67	0,98	0,93	0,20	0,26	0,70	
		$\alpha = 0{,}1$ $\gamma = 0{,}01$	Balíček Face	0,98	0,99	1,00	1,00	0,85	0,74	0,99	0,98	0,21	0,20	0,83	
	Tabuľka	$\alpha = 0{,}8$ $\gamma = 0{,}8$	Balíček Face	0,90	0,86	0,99	0,91	0,75	0,75	0,95	0,89	0,17	0,37	0,71	
		$\alpha = 0{,}1$ $\gamma = 0{,}01$	Balíček Face	0,99	0,97	0,99	0,99	0,77	0,77	0,98	0,98	**0,26**	0,42	0,82	
Liège prediktor			Balíček Liège								0,93				0,1

Tabuľka 9.1: Úspešnosť Q-agentov

Táto tabuľka ukazuje úspešnosť Q-agenta s rôznymi parametrami proti testovacím agentom a náhodným hráčom. Agenti boli trénovaní na 5000 hrách a testovaní na 500 hrách. Každé políčko v tabuľke uvádza úspešnosť agenta, ktorého reprezentuje riadok proti agentovi, ktorého reprezentuje stĺpec. Červenou farbou sú uvedení agenti a balíčky z práce autora David Taralla (1). V poslednom riadku chýbajú údaje, pretože sa jedná o agenta z tejto práce, a ten nemohol byť testovaní proti našim agentom. V poslednom stĺpci chýbajú údaje, pretože sa jedná o testovacieho agenta z tej istej práce, ktorého pre nedostatok informácií nebolo možné zreplikovať.

Dátová štruktúra			Neurónová sieť		Tabuľka	
	Parametre Q-učenia		$\alpha = 0{,}8$ $\gamma = 0{,}8$	$\alpha = 0{,}1$ $\gamma = 0{,}01$	$\alpha = 0{,}8$ $\gamma = 0{,}8$	$\alpha = 0{,}1$ $\gamma = 0{,}01$
		Balíček	Balíček Face	Balíček Face	Balíček Face	Balíček Face
Neurónová sieť	$\alpha = 0{,}8$ $\gamma = 0{,}8$	Balíček Face		0,53	0,54	0,46
	$\alpha = 0{,}1$ $\gamma = 0{,}01$	Balíček Face	0,47		0,52	0,44
Tabuľka	$\alpha = 0{,}8$ $\gamma = 0{,}8$	Balíček Face	0,46	0,48		0,53
	$\alpha = 0{,}1$ $\gamma = 0{,}01$	Balíček Face	0,54	0,56	0,47	

Tabuľka 9.2: Vzájomná úspešnosť Q-agentov

Tabuľka uvádza úspešnosti rôznych variantov Q-agenta proti sebe. Agenti boli trénovaní na 5000 hrách a testovaní na 500 hrách. Políčko tabuľky znamená úspešnosť agenta reprezentovaného riadkom proti agentovi reprezentovaným stĺpcom. Diagonála je začiernená preto, že by ukazovala úspešnosť agenta proti identickému agentovi.

10 Záver

Našim pôvodným cieľom bolo vytvoriť umelého hráča, ktorý je schopný sa učiť hrať hru Hearthstone. Podarilo sa nám ho splniť čiastočne, len pre jednu špecifickú taktiku, pomocou Q-učenia. Nami zostrojení agenti dokázali porážať Náhodných hráčov a konkurovať testovacím agentom, ktorí sa riadia súborom pravidiel prispôsobených konkrétnemu balíčku. Preto je tento prístup sľubný aj pre iné taktiky.

Táto konkurencieschopnosť bola dosiahnutá až po významnom zredukovaní priestoru stavov a akcií. Agenti používajúci neurónovú sieť síce v niektorých prípadoch za agentom používajúcim tabuľku zaostával, ale často mu bol vyrovnaným súperom a niekoľkokrát ho dokonca aj prekonal. Tieto poznatky ukazujú, že neurónová sieť je solídnou náhradou tabuľky, ale pri zložitých prostrediach je vhodné zredukovať priestory stavov a akcií tak, aby neboli zbytočne väčšie, ako to vyžaduje odmeňovacia funkcia.

Ďalší záver je špecifický pre prostredie hry Hearthstone. Potvrdil sa predpoklad, že zloženie balíčku veľmi ovplyvňuje výsledky hier a pri interpretácii výsledkov je nutné tento faktor nezanedbávať. Zároveň sme ukázali vzájomnú závislosť použitého balíčku a hranej stratégie.

11 Rozšíriteľnosť

Táto práca síce priniesla solídne výsledky v rámci vývoja umelej inteligencie pre hru Hearthstone, stále však chýba veľký krok k vývoju agenta schopného sa všeobecne naučiť vyhrávať nad súpermi v tejto hre. Uvádzame preto niekoľko návrhov do budúcnosti, ktoré majú potenciál zlepšiť výsledky agentov.

11.1 Q-učenie inak

V našej práci je ukázané, ako veľmi komplexné je prostredie hry Hearthstone. Existuje preto široké spektrum možností, ako navrhnúť priestor stavov, akcií, ale aj odmeňovaciu funkciu pre Q-učenie.

Bolo by vhodné zovšeobecniť pravidlá stratégií, podľa ktorých sa riadia testovací agenti. Potom by bolo možné vytvoriť agentov, ktorí vedia hrať určenú stratégiu s každým balíčkom na rozdiel od našich testovacích agentov, ktorých chovanie je definované špecificky pre pevný balíček.

11.2 Double Q-learning

Hado van Hasselt vo svojom článku (9) uvádza, že slabší výkon algoritmu Q-učenie býva spôsobený tým, že preceňuje niektoré akcie. Ukazuje taktiež alternatívu, „Double Q-learning“. Funguje tak, že namiesto jednej funkcie má dve funkcie Q^A a Q^B. Po každej akcii sa aktualizuje len jedna náhodne vybraná tabuľka, a to nasledujúcim spôsobom:

$Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\big(r + \gamma Q^B(s', a^*) - Q^A(s,a)\big)$ v prípade tabuľky pre Q^A, v opačnom prípade analogicky.

Tento prístup vedie podľa autora k eliminácii preceňovania hodnôt akcií.

11.3 SARSA

Ďalšou alternatívou je použitie algoritmu SARSA (State-Action-Reward-State-Action), ktorý funguje podobne ako Q-učenie, ale jeho učiaci krok je pozmenený na:

$Q_{t+1}(s,a) \coloneqq Q_t(s,a) + \alpha(s,a) \cdot \big(R_{t+1} + \gamma\, Q_t(s',a') - Q_t(s,a)\big)$.

Význam parametra a' je „akcia vykonaná v stave s'“. Ostatné parametre majú rovnaký význam, aký bol popísaný v 4.1.

11.4 Skúmanie balíčkov

Jedným zo záverov tejto práce je pozorovanie, že zloženie balíčku má veľký vplyv na hru agenta. Výber balíčku sme ale nenechávali na umelú inteligenciu. Nepreskúmaným územím v rámci umelej inteligencie pre Hearthstone je práve tento výber kariet pre hru. V práci sme tiež pozorovali, že naskriptovaní hráči dosahujú vysoké úspešnosti, dokonca aj keď ich chovanie nie je ošetrené pre všetky hraničné situácie. Preto by mohol zaujímavé výsledky dosiahnuť agent hrajúci s nejakým zadefinovaným chovaním optimalizujúci balíček pre toto chovanie. Alternatívou je komplexný agent schopný ako vyberať balíček, tak s ním aj hrať.

Zoznam použitej literatúry

1. **Taralla, David.** Learning Artificial Intelligence in Large-Scale Video Games. *University of Liège.* [Online] 23. 6 2015. http://orbi.ulg.ac.be/handle/2268/183266.

2. **Zopf, Markus.** A Comparison Between the Usage of Flat and Structured Game Trees for Move Evaluationin Hearthstone. *Technische Universität Darmstadt.* [Online] 2015. http://www.ke.tu-darmstadt.de/lehre/arbeiten/master/2015/Zopf_Markus.pdf.

3. **Hearthpwn.** *http://www.hearthpwn.com/.* [Online] http://www.hearthpwn.com/.

4. **IcyVeins.** Icy Veins. [Online] http://www.icy-veins.com/hearthstone/.

5. **Richard S. Sutton, Andrew G. Barto.** Optimal Value Functions. [aut. knihy] Andrew G. Barto Richard S. Sutton. *Reinforcement learning: An Introduction.* London : MIT Press, 1998.

6. **Hearthbreaker. [Online] https://github.com/danielyule/hearthbreaker.**

7. FirePlace. [Online] https://github.com/jleclanche/fireplace.

8. HearthstoneAI. [Online] https://github.com/Kerzak1408/HearthstoneAI.

9. *Double Q-learning.* van Hasselt, Hado. Vancouver, Canada : s.n., 2010.

10. Face Hunter, zdrojový kód. [Online] https://github.com/Kerzak1408/HearthstoneAI/blob/master/HearthstoneAI/AI/bots/face_hunter.py.

11. Secret paladin. [Online] https://github.com/Kerzak1408/HearthstoneAI/blob/master/HearthstoneAI/AI/bots/secret_paladin.py.

12. Malygos Freeze Mage. [Online] https://github.com/Kerzak1408/HearthstoneAI/blob/master/HearthstoneAI/AI/bots/malygos_freeze_mage.py.

13. HEarthstone Deck Tracker. [Online] https://hsdecktracker.net/.

14. SourceForge. [Online] https://sourceforge.net/projects/numpy/files/NumPy/.

15. PyBrain. [Online] http://pybrain.org/docs/.

16. Python. [Online] https://www.python.org/.

Zdroje obrázkov

[O1] *Screenshot hry Hearthstone*

[O2] *Blizzard Press Center, http://blizzard.gamespress.com/Hearthstone*

[O3] *Hearthstone Deck Tracker*

Zoznam tabuliek

Prílohy

A) Balíček Face

2x Glaivezooka

1x Bear Trap

1x Explosive Trap

2x Quick Shot

2x Eaglehorn Bow

2x Animal Companion

2x Kill Command

2x Unleash the Hounds

2x Abusive Sergeant

2x Leper Gnome

1x Worgen Infiltrator

2x Haunted Creeper

1x Ironbeak Owl

2x Knife Juggler

2x Mad Scientist

2x Arcane Golem

2x Argent Horserider

B) Balíček Secret

2x Avenge

1x Competitive Spirit

2x Noble Sacrifice

1x Redemption

2x Shielded Minibot

1x Coghammer

2x Muster for Battle

1x Truesilver Champion

2x Blessing of Kings

2x Keeper of Uldaman

2x Mysterious Challenger

1x Tirion Fordring

2x Secretkeeper

1x Haunted Creeper

2x Knife Juggler

2x Piloted Shredder

1x Loatheb

2x Sludge Belcher

1x Dr. Boom

C) Balíček Malygos

2x Ice Lance

2x Frostbolt

2x Arcane Intellect

2x Frost Nova

2x Ice Barrier

2x Ice Block

2x Fireball

2x Blizzard

1x Flamestrike

1x Pyroblast

1x Bloodmage Thalnos

2x Doomsayer

2x Loot Hoarder

2x Mad Scientist

1x Acolyte of Pain

1x Antique Healbot

1x Emperor Thaurissan

1x Alexstrasza

1x Malygos

D) Balíček Liège

2x Abomination

2x Abusive Sergeant

1x Acolyte of Pain

1x Arcane Explosion
2x Arcane Intellect
1x Arcane Missiles
2x Bloodfen Raptor
1x Bluegill Warrior
2x Boulderfist Ogre
2x Chillwind Yeti
2x Fireball
2x Flamestrike
2x Frostbolt
2x Gurubashi Berserker
1x Ironfur Grizzly
2x Mana Wyrm
2x Senjin Shieldmasta
2x Shattered Sun Cleric

E) Face hunter (chovanie)

V každom ťahu rekurzívne vyberie množinu kariet, ktoré zahrá tak, aby maximalizoval ich celkovú cenu. To robí za účelom čo najväčšieho využitia many. V prípade, že existuje viacero množín kariet maximalizujúcich ich cenu, agent vyberie jednu z nich na základe ich celkového skóre. Fukcia $skóre: Karty \rightarrow \mathbb{R}$, priraďujúca skóre jednotlivým kartám, je definovaná nasledovne:

- $skóre(Leper\ Gnome) = 3.$
- $skóre(Worgen\ Infiltrator) = 2.$
- $skóre(Abusive\ Sergeant) = 1.$
- $skóre(Mad\ Scientist) = 8.$
- $skóre(Haunted\ Creeper) = 7.$
- $skóre(Knife\ Juggler) = 6.$
- $skóre(Bear\ Trap) = 5.$
- $skóre(Glaivezooka) = 3$ ak agentov hrdina nemá zbraň.
  $= 0$ ak agentov hrdina má zbraň.
- $skóre(Explosive\ Trap) = 3.$
- $skóre(Ironbeak\ Owl) = 2.$
- $skóre(Steady\ Shot) = 1.$

- $skóre(Eaglehorn\ Bow) = 7$ ak agentov hrdina nemá zbraň.
  $= 0$ ak agentov hrdina má zbraň.
- $skóre(Ironbeak\ Owl) = 2.$
- $skóre(Animal\ Companion) = 6.$
- $skóre(Arcane\ Golem) = 5.$
- $skóre(Argent\ Horserider) = 4.$
- $skóre(Quick\ Shot) = 3.$
- $skóre(Kill\ Command) = 8$ ak je na ploche priateľská Beast postavička.
  $= 2$ inak.
- $skóre(Unleash\ the\ Hounds) = 9$ ak sú na ploche aspoň 3 postavičky súpera.
  $= 1$ inak.
- $skóre(Iná\ karta) = 0.$

Po zahraní kariet útočí všetkým, čím môže, na súperovho hrdinu. V prípade, že na súperovho hrdinu zaútočiť nemôže, náhodne útočí na postavičky s Tauntom (pozri podkapitolu 2.2.2).

Zdrojový kód tohto agenta je verejný a voľne dostupný (10).

F) Secret Paladin (chovanie)

Na rozdiel od Face huntera sa nesnaží o maximálne využitie many, ale o čo najväčšie navýšenie celkového skóre kariet, ktoré v danom kole zahrá. Skóre postavičiek je dané funkciou $skóre: Karty \rightarrow \mathbb{R}$ nasledovne:

- $skóre(Mysterious\ Challenger) = 15.$
- $skóre(Dr.Boom) = 8.$
- $skóre(Keeper\ of\ Uldaman)$ sa spočíta tak, že pre každú nepriateľskú postavičku sa spočíta rozdiel súčtu jej životov a útoku a šestky, pre každú priateľskú sa spočíta rovnaká hodnota s opačným znamienkom. Zo všetkých týchto hodnôt sa vyberie maximum. V prípade prázdnej plochy je skóre -100. Dôvodom tohto výpočtu je snaha o maximalizácia efektu tejto karty, ktorý zmení životy aj útok jednej vybranej postavičky na tri.
- $skóre(Mysterious\ Challenger) = 15.$
- $skóre(Coghammer) = 3$ ak má agent aspoň jednu postavičku na ploche.
- $skóre(Muster\ for\ Battle) = 10.$

$= 60$ ak má agent na ploche postavičku Knife Juggler.

- $skóre(Blessing\ of\ Kings) = 8$ ak má agent aspoň jednu postavičku,
  $= -8$ inak.
- $skóre(Avenge) = 5.$
- $skóre(Noble\ Sacrifice) = 3.$
- $skóre(Redemption)$ je počítané ako súčet útokov priateľských postavičiek na ploche v prípade, že je na ploche aspoň jedna postavička a každá z postavičiek na ploche má aspoň jeden útok. V prípade, že je na ploche nejaká postavička s útokom menším ako dva, toto skóre je -1. V prípade, že na ploche nemá agent žiadnu priateľskú postavičku, je toto skóre -2.
- $skóre(Competitive\ Spirit)$ predstavuje rozdiel počtu priateľských postavičiek a počtu nepriateľských postavičiek znížený o 2.
- $skóre(Iná\ karta) = 1.$

K všetkým skóre týchto kariet sa pripočítava ešte $bonus : (Typy\ kariet) \rightarrow \mathbb{R}$ definovaný nasledovne:

- $bonus(Postavička)$ je súčtom jej útoku a životov. V prípade, že má postavička Taunt, je tento súčet navýšený o jednotku. V prípade, že má Divine Shield, je celý bonus ešte prenásobený 1,5-krát, rovnako aj v prípade, že má Deathrattle (vysvetlivky k pojmom v podkapitole 2.2.2).
- $bonus(Zbraň) = -10$ v prípade, že hrdina už má nejakú inú zbraň.
- $bonus(Iný\ typ) = 0.$

Po zahraní kariet začne agent útočiť. Pred útočením si odsimuluje sám pre seba všetky možné spôsoby, ako môže zaútočiť, a vyberie ten, ktorý maximalizuje funkciu $skórePlochy : StavyHracejPlochy \rightarrow \mathbb{R}$ definovanú tak, že každá zničená súperova postavička do nej prispieva súčtom veľkosti svojho útoku a desatiny svojich životov navýšeným o 0,5, zatiaľ čo každá zničená priateľská postavička ju znižuje o veľkosť svojho útoku. Zdrojový kód je voľne dostupný na (11).

G) Malygos freeze mage (chovanie)

Stratégia tohto agenta spočíva v zahraní kombo kombinácie počas dvoch kôl. Táto kombo kombinácia je zložená z kariet Emperor Thaurissan, Malygos a čo najviac kariet Frostbolt a Ice Lance. Hra agenta je rozdelená na tri fázy: čakaciu fázu, fázu

Emperor a fázu Malygos. Začína v čakacej fáze, do fázy Emperor prechádza, ak má na ruke karty Malygos, Emperor Thaurissan, minimálne jeden Frostbolt a platí podmienka, že celkové poškodenie, ktoré dokáže spôsobiť jeho kombo kombinácia, je väčšie alebo rovné počtu životov súperovho hrdinu. Ak sa nachádza v čakacej fáze, na začiatku každého kola skontroluje, či súčet útokov nepriateľských postavičiek presahuje 9. V prípade, že áno a má k dispozícii aspoň nejaké z kariet Blizzard, Flame Strike a Frost Nova, použije jednu z nich. Inak hrá v čakacej fáze karty tak, aby počas každého kola maximalizoval využitie many. Niektoré karty má však dovolené použiť len počas určitých herných situácií:

- Kombo karty nemôže použiť v čakacej fáze za žiadnych okolností.
- Kartu Antique Healbot môže použiť len ak má jeho hrdina menej ako 26 životov.
- Kartu Doomsayer môže použiť len ak je súčet útokov nepriateľských postavičiek väčší ako 4 a na ploche nie je iný Doomsayer.
- Kartu Arcane Intellect môže použiť len ak má na ruke menej ako 10 kariet.

V momente, keď sa dostane do Emperor fázy, zahrá kartu Emperor Thaurissa, prípadne kartu Ice Block, ak ju má k dispozícii.

Keď sa dostane do Malygos fázy, zahrá víťaznú kombináciu: Malygosa nasledovaného Frost Boltom a ďalšími kartami Frost Bolt a Ice Lance.

Všetky útoky mieri na súperovho hrdinu.

Zdrojový kód so všetkými detailmi je k dispozícii na (12).

H) Programátorská dokumentácia

1 Východisková situácia

Pred začatím vývoja sme mali k dispozícii FirePlace – simulátor hry Hearthstone. Tento simulátor bol pomocou modulu full_game.py (v balíčku tests) schopný korektne odsimulovať ľubovoľný počet hier. Textový výstup hry značne komplikoval čitateľnosť priebehu hry. Z pohľadu umelej inteligencie bol východiskovým bodom takisto modul full_game.py. Nachádzala sa v ňom implementácia chovania dvoch rovnakých náhodných hráčov proti sebe.

2 Simulácie

Pre spúšťanie samotných simulácií sme v priečinku tests vytvorili modul general_game.py. Jeho main metóda spracováva argumenty a volá metódu

test_full_game, ktorá vytvára priečinky a súbory pre výsledky. Okrem toho dynamicky inicializuje agentov určených vstupnými argumentmi a volá metódu play_full_game, ktorá simuluje jednu hru. Metóda play_full_game vyžaduje akciu od hráča, ktorý je na ťahu. Túto akciu následne spracováva a postupne rozširuje hashmapu, ktorá je na konci programu použitá na vytvorenie súboru replay.json. Ten je následne zabalený do formátu ZIP a jeho koncovka pozmenená na „.hdtreplay“. Výsledkom je záznam hry zobraziteľný programom Hearthstone Deck Tracker (13).

3 Agenti

Všetci agenti sú triedy v moduloch, ktoré sú umiestnené v priečinku AI/bots. Takáto trieda musí implementovať všetky metódy, ktoré sa nachádzajú v module bot_template.py. Konštruktor __init__ obsahuje inicializáciu hrdinu a balíčku kariet podľa parametrov. Metóda get_id vracia identifikátor agenta. Tento identifikátor je tvorený menom agenta a jeho verziou. Pôvodný balíček kariet agenta vracia metóda get_deck a názov tohto balíčku metóda get_deck_id. Metódy __init__ get_id, get_deck a get_deck_id sú spoločné naprieč všetkými agentmi, a preto ich ďalej nebudeme spomínať. Metóda get_mulligans vracia zoznam kariet, ktoré agent chce zamiešať do balíčka v prvej fáze hry. Najdôležitejšou metódou je metóda get_next_turn, ktorá vracia ďalšiu akciu agenta vždy, keď je na ťahu.

4 Náhodný hráč

Implementácia Náhodného hráča sa nachádza v module random_bot_2.py. Jeho metóda get_mulligans vracia náhodnú podmnožinu z kariet, ktoré dostane na výber. Ako ďalší ťah vracia v metóde get_next_turn vždy jednu náhodnú akciu z tých, ktoré môže urobiť. Medzi tieto akcie sa zaraďuje aj ukončenie kola.

5 Náhodne akčný hráč

Modul obsahujúci Náhodne akčného hráča sa volá random_bot.py. Jeho metóda get_mulligans je identická s rovnomennou metódou Náhodného hráča (pozri 4). Toto tvrdenie už neplatí pre metódu get_next_turn, pretože Náhodne akčný hráč vracia ako ďalší ťah náhodnú akciu okrem ukončenia kola, pokiaľ môže. Keď už nemôže vykonať žiadnu akciu okrem ukončenia kola, ukončí ho.

6 Face hunter

Umiestnenie implementácie Face huntera je v module face_hunter.py. Jeho metóda get_mulligans vracia všetky karty zo vstupu s výnimkou kariet Leper Gnome a Worgen Infiltrator. Ako ďalší ťah vracia v metóde get_next_turn najprv karty, ktoré si vyberie pomocou metódy get_cards_to_play na základe ich ceny, ktorú minimalizuje metóda get_min_loss_card_cost. Metóda get_secondary_score rozhoduje, ktoré karty použiť v prípade rovnakej cenovej výhodnosti viacerých kombinácií. Po tom, ako agent zahrá karty, metóda get_next_turn začne vracať útoky primárne na súperovho hrdinu, sekundárne na postavičky s Tauntom.

7 Secret paladin

Secret paladin je umiestnený v module secret_paladin.py. Metódu get_mulligans má identickú s náhodnými hráčmi (pozri 4 a 5). O ďalšom ťahu rozhoduje v metóde get_next_turn primárne na základe metódy get_cards_to_play. To je metóda, ktorá rekurzívne vyberá najlepšie možné karty na základe ohodnotenia, ktoré poskytuje metóda get_cards_score. Metódy get_redemption_score a get_keeper_bonus_score slúžia na ohodnotenie dvoch špecifických kariet (Redemption, Keeper of Uldaman) pre účely výberu hraných kariet. V prípade zahrania karty Keeper of Uldaman je volaná metóda get_keeper_target, ktorá určuje, na koho karta zamieri. Po zahraní kariet nasleduje výber útoku pomocou metódy get_next_attack. Výber útoku prebieha rekurzívne cez všetky možné útoky a snaží sa o prevahu útokov postavičiek v poli.

8 Malygos freeze mage

V module malygos_freze_mage.py je implementácia agenta Malygos freeze mage. Jeho metóda get_mulligans vracia všetky karty, ktoré dostane ako argument, okrem kariet Loot Hoarder, Mad Scientist, Arcane Intellect, Bloodmage Thalnos, Doomsayer a Acolyte of Pain. Metóda get_next_turn vracia najprv karty na zahranie a potom útočí s takmer rovnakými preferenciami ako Face hunter (pozri 6) podľa metódy get_next_attack. Karty vracia podľa fázy, v ktorej sa agent nachádza. V čakacej fáze sú to buď karty podľa metódy get_playable_freeze_card (pri veľkej prevahe súpera v poli), alebo karty podľa metódy get_next_waiting_phase_turn (keď súper nemá veľkú prevahu v poli alebo metóda has_playable freeze_card vracia False, čo značí, že agentova metóda get_playable_freeze_card by vrátila None). Metóda

get_playable_freeze_card vracia niektorú z dostupných plošných zamrazovacích kariet na ruke. V prípade volania metódy get_next_waiting_phase_turn sa agent snaží maximalizovať ich cenu pomocou metódy get_min_loss_card_costs. V prípade rovnosti cien viacerých kombinácií kariet medzi nimi rozhoduje skóre definované metódou get_secondary_score. V tejto fáze je výhodnosť hrania každej karty overovaná metódou is_allowed_as_pawn. Metóda has_emperor vracia informáciu, či agent na ruke má kartu Emperor Thaurissan. Potenciálne poškodenie kombo kombináciou vracia metóda get_available_combo power. V prípade, že je toto poškodenie dostatočné na zničenie súperovho hrdinu, prechádza agent do fázy Emperor. V tejto fáze hranie kariet určuje metóda get_next_emperor_phase_turn. Fáza Emperor trvá len jedno kolo, hneď po ňom prechádza agent do fázy Malygos, kedy zahrané karty určuje metóda get_next_malygos_phase_turn.

9 Q-agent s tabuľkou

Q-agent s tabuľkou je naimplementovaný v module q_learner_table.py. Jeho metóda get_mulligans vracia náhodnú podmnožinu kariet zo vstupu. V metóde get_next_turn sa rozhoduje pre ďalšiu akciu pomocou metódy get_best_action. Okrem toho aktualizuje tabuľku funkcie Q pomocou metód get_reward (vracia odmenu pre Q-učenie) a update_table (samotná aktualizácia tabuľky). Index súčasného stavu pre tabuľku získava pomocou metódy get_state, index akcie zas pomocou metódy get_action. Metóda get_best_action vracia najlepšiu možnú akciu v danom stave a jej hodnotu v tabuľke funkcie Q.

10 Q-agent s neurónovou sieťou

Implementácia Q-agenta s neurónovou sieťou je umiestnená v module Q_learner_super_makro.py. Jeho metóda get_mulligans vracia náhodnú podmnožinu kariet zo vstupu. Metóda get_next_turn vracia ďalší ťah obdobne ako tomu je u Q-agenta s tabuľkou (pozri 9) s tým rozdielom, že neaktualizuje tabuľku, ale neurónovú sieť pomocou metódy update_neural_network. Analógia platí aj pri metódach get_state a get_action. Namiesto indexu v tabuľke však vracajú vstupné vektory pre neurónovú sieť. Metóda create_dataset vytvára trénovaciu množinu pre algoritmus back-propagation na základe stavu a akcie.

I) Užívateľská dokumentácia

1 Požiadavky

Pre používanie frameworku je nutné mať nainštalovaný Python verzie 3.5 alebo novší, vrátane balíčku NumPy (14). Odporúčanou je 64-bitová verzia Pythonu kvôli agentom s veľkými pamäťovými požiadavkami. Kvôli agentom využívajúcim neurónovú sieť je nutné nainštalovať aj PyBrain (15). Zdrojový kód frameworku je voľne dostupný na (8).

2 Inštalácia

Python aj s inštalačnými inštrukciami je voľne dostupný na svojich oficiálnych stránkach (16). Balíček NumPy sa dá nainštalovať z príkazového riadku príkazom „pip install numpy“. Simulátor sa inštaluje príkazom „pip install -r requirements.txt“ z priečinku HearthstoneAI. Návod na inštaláciu PyBrain sa nachádza na jeho oficiálnych stránkach (15). Pre zobrazovanie záznamov hier je vhodné mať nainštalovaný aj program Hearthstone Deck Tracker (návod na oficiálnych stránkach (13)).

3 Pridanie umelej inteligencie

Všetkých agentov a balíčky kariet pokrýva priečinok AI.

4 Nový agent

Všetci agenti sú lokalizovaní v priečinku AI/bots. Medzi inými sa tam nachádza aj vzorový agent bot_template.py obsahujúci všetky metódy, ktoré musí implementovať každý umelý agent. Najdôležitejšou metódou je metóda get_next_turn, ktorá je volaná vždy, keď je hráč na ťahu. Každý agent má verziu, ktorá je uložená v premennej „version“. Identifikátor agenta je zložený z mena agenta a jeho verzie.

5 Nový balíček

Každý jeden balíček, s ktorým môžu agenti hrať, sa nachádza v balíčku decks, ktorý je umiestnený v balíčku AI. Jednému balíčku zodpovedá práve jeden textový súbor. Jeho názov musí začínať názvom hrdinu, pre ktorého je určený. Tento názov hrdinu musí byť oddelený od zvyšku podčiarkovníkom. Obsahom súboru je tridsať riadkov, pričom na každom z nich sa nachádza meno jednej karty.

6 Simulácie

Podľa vzoru modulu full_game.py bol vytvorený modul general_game.py určený na odsimulovanie jednej alebo viacerých hier definovaných v balíčku bots (pozri 4). Pre každú odsimulovanú hru vytvára priečinok v priečinku game_results. Tento priečinok obsahuje záznam hry pre štatistické spracovanie vo formáte csv. Okrem toho obsahuje aj súbor s príponou „.hdtreplay“, ktorý je vizuálnym záznamom hry a je ho možné zobraziť programom Hearthstone Deck Tracker (pozri (13)). Každá simulácia hry pridáva riadok s výsledkom do súhrnného súboru s výsledkami simulácií (pre špecifiká tohto súboru pozri 7) .

7 Argumenty

Modul je potrebné spustiť so šiestimi argumentmi určujúcimi parametre simulácií:

- Prvým argumentom je meno prvého z agentov zúčastňujúcich sa simulácií. Toto meno je názov triedy, ktorá je umiestnená v niektorom z modulov v balíčku AI/bots. Program prehľadáva všetky triedy vo všetkých moduloch v danom balíčku, pre prehľadnosť sme každú triedu mali vo vlastnom module.
- Druhý argument určuje meno balíčku, s ktorým bude hrať agent definovaný prvým parametrom. Menom je myslený názov súboru s balíčkom bez prípony „.txt“.Tento súbor musí spĺňať všetky podmienky stanovené v podkapitole 5.
- Tretím argument definuje druhého agenta obdobne ako prvý argument prvého.
- Štvrtý argument definuje balíček pre druhého agenta obdobne ako druhý argument pre prvého.
- Počet odsimulovaných hier určuje piaty argument.
- Posledný argument rozhoduje, do ktorého súboru sa bude zapisovať informačný riadok o výsledku každej odsimulovanej hry. Tento riadok sa skladá z identifikátorov oboch agentov, mien ich balíčkov, identifikátoru víťaza a začínajúceho hráča. Validnými hodnotami tohto argumentu sú:
    - „F“ pre pripísanie výsledkov simulácií na koniec súboru results_summary.csv, ktorý sa nachádza v priečinku game_results,
    - „T“ pre nahradenie obsahu súboru results_summary.csv výsledkami simulácií,

- „O“ pre vytvorenie vlastného súboru pre výsledky simulácií v priečinku game_results (jeho názov sa skladá z identifikátorov zúčastnených agentov a mien použitých balíčkov).