ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Metody umělé inteligence pro motivační herní prostředí
Student:	Matěj Sochor
Vedoucí:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2018/19

Pokyny pro vypracování

Cílem práce je prozkoumat možnosti herní umělé inteligence přizpůsobit svou strategii strategiím používaným proti ní v komplexní strategické hře.
1) Seznamte se s metodami umělé inteligence, které se používají v počítačových hrách. Seznamte se s motivační webovou aplikaci Taskino (www.taskino.cz) [1], [2].
2) Vyberte či navrhněte vhodný algoritmus adaptivní umělé inteligence (např. dynamické skriptování, evoluční techniky, reinforcement learning), který umožní, aby počítačový hráč byl schopen přizpůsobit svou strategii strategiím, které jsou proti němu používány.
3) Algoritmus implementujte a ověřte jeho funkčnost.
4) Podrobně analyzujte a vyhodnoťte kvalitu navrženého algoritmu z hlediska jím vytvořené umělé inteligence, ve srovnání s počítačovými hráči, kteří mají neměnné chování.

Seznam odborné literatury

[1] Kvasnička, Michal. Motivační webová aplikace s využitím gamifikačních prvků. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2015.
[2] Kulík, Jakub. Herní prostředí s umělou inteligencí pro motivační webovou aplikaci. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 22. února 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Metody umělé inteligence pro motivační herní prostředí

Matěj Sochor

Vedoucí práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

15. května 2017

Poděkování

Chtěl bych poděkovat vedoucímu bakalářské práce doc. RNDr. Ing. Marcelu Jiřinovi, Ph.D. za jeho čas, trpělivost a nedocenitelné rady, Bc. Jakubu Kulíkovi za pomoc při pochopení jeho práce zásadní pro správné vyhodnocení naměřených výsledků a pozorování a rodině, přátelům a přítelkyni za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

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

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Sochor, Matěj. *Metody umělé inteligence pro motivační herní prostředí*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Bakalářská práce zkoumá možnosti herní umělé inteligence přizpůsobit svou strategii protivníkovi v komplexní strategické hře, která je součástí motivační úkolové aplikace Taskino, a tím zlepšit její hratelnost. Popisuje a hodnotí různé metody adaptivní umělé inteligence, z nichž pro praktické zhodnocení vybírá dynamické skriptování.

Dynamické skriptování je upraveno pro použití ve zkoumané hře a do hry implementováno. Kvalita upraveného algoritmu je vyhodnocena na základě zápasů jím vytvářené adaptivní umělé inteligence a tří neměnných umělých inteligencí, reprezentujících ofenzivní, defenzivní a budovatelskou herní strategii. Z průběhu a výsledků zápasů práce vyvozuje závěry ohledně herních mechanik zkoumané hry a doporučuje jejich vylepšení určená ke zvýšení její hratelnosti.

Klíčová slova Taskino, strategická hra, adaptivní umělá inteligence, strojové učení, dynamické skriptování

Abstract

This bachelor thesis examines possibilities of an artificial intelligence for games to adapt its strategy to its opponent in a complex strategy game, which

forms part of the motivational task management application Taskino. Such an adaptation would improve the gameplay. It describes and evaluates various methods of adaptive artifical intelligence. From these methods, a dynamic scripting method was chosen for the practical part.

Dynamic scripting was adjusted for the game and was implemented in it. Its quality was assessed according to the behaviour of the adaptive artificial intelligence (created by dynamic scripting) and its performance in the matches against three static artifical intelligences representing offensive, defensive and constructive game strategies. The thesis draws conclusions about the game mechanics of the game from the course and results of these matches and recommends enhancements to improve its gameplay.

Keywords Taskino, strategy game, adaptive artificial intelligence, machine learning, dynamic scripting

Obsah

Úvod **1**

1 Cíl práce **3**

2 Teorie **5**
2.1 Herní umělá inteligence 5
2.2 Taskino 8

3 Vybrané metody adaptivní umělé inteligence **13**
3.1 Posilované učení 13
3.2 Dynamické skriptování 14
3.3 Evoluční algoritmy 16
3.4 Monte-Carlo Tree Search 17

4 Návrh a implementace **19**
4.1 Výběr metody 19
4.2 Použité technologie 20
4.3 Úpravy hry 20
4.4 Dynamické skriptování 26

5 Experimenty a vyhodnocení **31**
5.1 Postup experimentů 31
5.2 Výsledky experimentů 32
5.3 Vyhodnocení průběhu her a vah pravidel 35
5.4 Diskuze 38

Závěr **43**

Literatura **45**

A Seznam použitých zkratek **49**

B Obsah přiloženého CD **51**

C Instalační příručka **53**
C.1 Závislosti 53
C.2 Průběh instalace v linuxovém prostředí 53
C.3 Spuštění aplikace 54

D Experimentální příručka **55**

Seznam obrázků

2.1 Stav herního ostrova umělé inteligence po 150 tazích, důsledek exponenciální rychlosti rozvoje populace 9
2.2 Kód řídící růst populace. Násobení současné populace vede k exponenciální rychlosti růstu. 10

3.1 Model interakce agenta s prostředím, který je základem posilovaného učení 14
3.2 Základní princip dynamického skriptování 15
3.3 Obecné fungování evolučních algoritmů. Přeloženo a upraveno autorem, původní zdroj je [1]. 16
3.4 Základní iterativní proces MCTS. Přeloženo a upraveno autorem, původní zdroj je [2]. 18

4.1 Původní způsob zpracování skriptu umělé inteligence 21
4.2 Vylepšený způsob zpracování skriptu umělé inteligence 21
4.3 Ukázka pravidla využívajícího militarizaci a agresivitu 22
4.4 Ukázka původní implementace jedné z možných podmínek pro pravidlový systém. Při každém ověření podmínky je nutné projít všechny budovy, což je při vyšším počtu budov velmi náročné. . . 24
4.5 Ukázka optimalizované implementace jedné z možných podmínek pro pravidlový systém. Hodnota je při prvním dotazu uložena a následně je pouze vracena. 24
4.6 Chybná funkce v původní verzi hry. Přepočítání množství nesených ukradených surovin v případě poškození čety chybně vede k navýšení tohoto množství. 25
4.7 Ukázka vojenských pravidel ze souboru pravidel 28

5.1 Podíl her vyhraných dynamickým skriptováním proti Útočníkovi . 33
5.2 Podíl her vyhraných dynamickým skriptováním proti Obránci . . . 34
5.3 Podíl her vyhraných dynamickým skriptováním proti Staviteli . . . 35
5.4 Implementované pravidlo řešící jeden typ ekonomického zaseknutí . 36

5.5 Ukázka nevyrovnanosti poměru sil a utrpěných ztrát výpisem herního serveru o bitvě. Slabý útočník (ve výpisu druhý) utrpěl podstatně menší surovinové ztráty než silný obránce. 38
5.6 Ukázka neefektivity agrese výpisem herního serveru o bitvě. Velmi silný útočník s množstvím obléhacích jednotek (ve výpisu druhý) způsobil obránci velmi nízké škody. 38
5.7 Ukázka neefektivity agrese výpisem herního serveru o bitvě. Silný útočník (ve výpisu druhý) utrpěl velmi vysoké ztráty při útoku na ostrov bráněný pouze malým množstvím věží. 39

Seznam tabulek

4.1 Změny militarizace v závislosti na akcích umělé inteligence 22
4.2 Změny agresivity v závislosti na akcích umělé inteligence 22

5.1 Podíl her vyhraných dynamickým skriptováním proti Útočníkovi (čtyři experimenty pro obě varianty) 32
5.2 Podíl her vyhraných dynamickým skriptováním proti Obránci (čtyři experimenty pro obě varianty) 33
5.3 Podíl her vyhraných dynamickým skriptováním proti Staviteli (čtyři experimenty pro obě varianty) 33
5.4 Tabulka počtu zaseknutí rozvoje v počáteční fázi hry ve sto hrách proti jednotlivým statickým inteligencím pro učící i neučící se dynamické skriptování 36
5.5 Tabulka zaokrouhleného průměrného rozdílu vah ve všech dvojicích experimentů, rozdělených podle inteligence proti které byly váhy učeny. První hodnota bere v úvahu váhy všech pravidel, druhá pouze vojenských. 37

Úvod

Kvalita herní umělé inteligence je jedním z nejdůležitějších faktorů ovlivňujících hratelnost moderních strategických her, přesto bývá často zanedbávána a hry se snaží zaujmout grafickým zpracováním, zajímavými příběhy a hratelností hry více hráčů.

Většina her tak stále využívá herní umělou inteligenci s neměnným chováním, která je dostatečná pro začínající hráče, ale selhává ve snaze poskytnout dostatečnou výzvu zkušenějším hráčům a hratelnost proti ní rychle klesá. Kromě monotónnosti chování, kterému může lidský hráč snadno přizpůsobit svou strategii, často obsahuje zásadní nedostatky a zranitelnosti, které může člověk zneužívat a snadno proti ní vítězit.

Herní umělá inteligence proto zpravidla po krátké době selhává ve svém úkolu, kterým je poskytnout konzistentní výzvu odpovídající schopnostem hráče. Většina her se tento problém snaží řešit možností zvolit různé úrovně obtížnosti, které však obvykle fungují spíše na principu poskytnutí nespravedlivých výhod umělé inteligenci, než na principu výrazného zlepšení jejího rozhodování. Vyšší úrovně obtížnosti proto stále trpí stejnými neduhy jako ty nižší. Výhody jim poskytnuté navíc mohou být pro hráče velmi frustrující, jako příklad může sloužit chování nejobtížnější umělé inteligence ve hře *Medieval II: Total War*. Ta má k dispozici mnohem více zdrojů než lidský hráč, což vede k časově náročným bitvám s velkým množstvím početných nepřátelských armád, které však stále trpí nekvalitní bitevní inteligencí, a jsou proto snadno poráženy opakováním několika základních taktik. Vyšší obtížnost tak nevede k náročnější hře, nýbrž pouze prodlužuje herní dobu.

O řešení monotónnosti počítačem ovládaných protivníků i problému přizpůsobení jejich obtížnosti schopnostem hráče se snaží výzkum v oblasti adaptivní herní umělé inteligence. Ten již navrhl množství metod, které byly s větším či menším úspěchem vyzkoušeny v různých komerčních hrách. Většina her je však stále nevyužívá, zčásti kvůli neznalosti herních vývojářů, zčásti kvůli obavám z jejich obtížné implementace, náročného testování a často nepředvídatelného charakteru. Právě jejich značný a stále z větší části nevyužitý

potenciál motivoval autora k prozkoumání možností jejich využití v existující strategické hře.

Taskino je webová aplikace pro organizaci plnění úkolů z reálného světa, využívající herní prostředky a principy k motivaci uživatelů. Jedním z těchto prostředků je komplexní strategická hra, vytvořená a rozšířená spolu se zbytkem aplikace Taskino v rámci dvou předcházejících závěrečných prací – diplomové práce Ing. Michala Kvasničky a bakalářské práce Bc. Jakuba Kulíka. Tato strategická hra je zaměřena na příležitostné hráče a třebaže obsahuje protivníky řízené umělou inteligencí, je tato inteligence neměnná a velmi nenáročná. Neposkytuje proto dostatečnou výzvu pro hráče, pro něž je hlavním zdrojem hratelnosti zápasení s protivníkem, a může již po krátké době začít nudit i příležitostné hráče. Oba tyto problémy limitují použitelnost hry jako dlouhodobého motivačního prostředku.

Adaptivní umělá inteligence by je mohla oba vyřešit a tím nejen poskytnout lepší hratelnost členům týmů využívajících Taskino, ale i zlepšit jeho atraktivitu jako motivačního prostředku a pomoci tak vedoucím těchto týmů. Právě možnostmi využití adaptivní umělé inteligence v Taskinu se tato práce zabývá.

První kapitola přesněji vymezuje cíl práce. Následující dvojice kapitol představuje teoretický základ pro praktickou část řešení. Druhá kapitola obsahuje základní informace o herní umělé inteligenci a teoretické znalosti potřebné pro pochopení kapitol následujících a zkoumá strategickou hru v Taskinu se zaměřením na informace důležité pro návrh adaptivní umělé inteligence. Třetí kapitola popisuje konkrétní zvažované metody adaptivní umělé inteligence a udává příklady jejich využití. Takto popsány jsou čtyři metody – posilované učení, dynamické skriptování, evoluční algoritmy a *Monte-Carlo Tree Search*.

Poslední dvě kapitoly popisují praktickou část řešení a její výsledky. Čtvrtá kapitola se zabývá výběrem metody, která bude dále zkoumána, popisem použitých technologií a popisem a vysvětlením úprav a vylepšení, které byly provedeny v Taskinu v rámci práce. Nakonec rozebírá úpravy vybrané metody (dynamického skriptování) nutné pro její použití v Taskinu, a její konkrétní implementaci a nastavení. Závěrečná kapitola seznamuje čtenáře s postupem experimentů určených k vyhodnocení úspěšnosti upraveného dynamického skriptování, detailně popisuje jejich výsledky a nakonec tyto výsledky diskutuje a navrhuje změny Taskina určené k řešení během experimentů odhalených nedostatků a zvýšení jeho hratelnosti.

Kapitola 1

Cíl práce

Cílem práce je prozkoumat možnosti využití metod adaptivní umělé inteligence pro přizpůsobení strategie umělé inteligence strategiím proti ní používaným v motivační hře v aplikaci Taskino. Takové přizpůsobení by zajistilo jak poskytnutí výzvy bojechtivým hráčům, tak i dlouhodobou hratelnost a zlepšilo tak použitelnost této hry jako motivačního prostředku.

Cílem teoretické části práce je seznámit se se základy herní umělé inteligence, s aplikací Taskino a s existujícími metodami adaptivní herní umělé inteligence potenciálně využitelnými v aplikaci Taskino.

Cílem praktické části je vybrat jednu z těchto metod nebo navrhnout metodu vlastní, implementovat ji v aplikaci Taskino a vyhodnotit její kvalitu na základě porovnání jí vytvořené umělé inteligence a počítačových protivníků s neměnným chováním.

Vzhledem ke značné složitosti problému a vysokým výpočetním nárokům experimentů se práce omezuje na případ zápasení dvou hráčů. Při výběru vhodné metody však stále bere v úvahu i případné praktické využití v zápasech více hráčů.

Cílem práce není vytvoření pro lidské hráče hratelné podoby navrhované adaptivní inteligence, ani kompletní implementace do hry za tímto účelem.

Kapitola 2

Teorie

Tato kapitola nejprve obecně zkoumá herní umělou inteligenci a poskytuje základní znalosti o ní potřebné k plnému pochopení následujících částí práce. Ve své druhé části podrobně popisuje strategickou hru, která je součástí motivační aplikace Taskino, se zaměřením na informace důležité pro návrh adaptivní umělé inteligence.

2.1 Herní umělá inteligence

Chápání herní umělé inteligence se značně liší od akademické definice umělé inteligence. Akademická umělá inteligence se zabývá technikami schopnými vykonávat úkoly, které jsou jednoduché pro člověka, ale náročné pro počítač. V řešení mnoha problémů mají dnešní počítače nadlidské schopnosti, například v aritmetice, řazení, vyhledávání a dokonce i v některých deskových hrách. V jiných ovšem ve srovnání s člověkem stále selhávají, například v rozeznávání tváří, chápání přirozeného jazyka, rozhodování mezi různými akcemi v závislosti na situaci a v kreativním myšlení. Právě tyto úkoly jsou doménou akademické umělé inteligence. Cílem herní umělé inteligence oproti tomu je, aby chování herních postav lidsky hlavně vypadalo. K tomu využívá množství různých, často k nepoznání upravených technik, včetně algoritmů převzatých z akademické umělé inteligence. [3, str. 3-4]

2.1.1 Agentní přístup k herní umělé inteligenci

Agent je entita schopná vnímat prostředí, ve kterém se pohybuje, a měnit ho svými akcemi. Agentní přístup definuje chování jednotlivých agentů, kteří se v prostředí nacházejí. Chování celé hry pak vzniká kombinací chování jednotlivců a jejich interakcí. Hra se tedy chová jako multiagentní systém. Opakem agentního přístupu je neagentní přístup, který přímo centralizovaně definuje chování celé hry. [3, str. 11]

2.1.1.1 Multiagentní systém

Multiagentní systémy vznikají kombinací více autonomních agentů. Určeny jsou k řešení problémů, které jsou pro jednoho agenta příliš komplexní, rozsáhlé nebo nepředvídatelné. Základními charakteristikami multiagentních systémů jsou nedostatečnost informací nebo schopností každého jednotlivého agenta pro vyřešení problému, abscence jejich globální kontroly, decentralizace informací a asynchronní běh. [4]

2.1.2 Úrovně herní umělé inteligence

V [3] jsou identifikovány dvě hlavní oblasti využití umělé inteligence pro jednotlivé agenty v herním prostředí – pohyb a rozhodování. Pro pohyb jsou zásadní algoritmy vyhledávání cesty. Pro rozhodování existuje velké množství různých technik, například konečné automaty, rozhodovací stromy nebo pravidlové systémy.

Komplexnější hry ale vyžadují inteligenci vyšší úrovně, než je pohyb jednotlivých agentů v herním prostředí a jejich reagování na současnou situaci. Takovou inteligenci je možné rozdělit na dvě základní úrovně [5]:

Strategická úroveň řídí chování nejvyšší úrovně. Definuje dlouhodobé cíle a rozhoduje, jak těchto cílů dosáhnout. Ve strategických hrách zajišťuje budování základen a obranných pozic, produkci surovin, rekrutaci jednotek apod. Implementací často jsou pravidlové systémy a konečné automaty.

Taktická úroveň se stará o chování skupin postav vykonávajících plán určený strategickou úrovní. Její součástí je například hledání cest, určování formací, ve kterých se jednotky pohybují, a výběr taktiky, kterou budou jednotky využívat. Implementace jsou různé v závislosti na implementované funkčnosti, od algoritmů vyhledávání cesty až po pravidlové systémy a konečné automaty pro řízení taktiky.

Hra může využívat pouze strategickou úroveň, pouze taktickou úroveň, jako mnohá FPS (*First-person shooter*), nebo obě najednou, typicky v komplexních strategických hrách.

2.1.3 Pravidlový systém

Pravidlové systémy jsou jednou z nejznámějších technik umělé inteligence. Již dlouhou dobu jsou používané ve hrách pro rozhodování agentů. Jejich hlavní výhodou oproti rozhodovacím stromům a konečným automatům je možnost rozhodování v situacích, které vývojář nepředpokládal. [3, str. 427]

Pravidlové systémy se obvykle skládají ze tří částí [3, str. 427-428]:

Databáze obsahující informace o stavu hry známé umělé inteligenci.

Souboru pravidel složených z podmínky a akce, která je provedena, pokud je podmínka na základě znalostí z databáze vyhodnocena jako pravdivá.

Arbitra rozhodujícího, které z pravidel provede svou akci, pokud jsou splněny podmínky u více z nich.

Pravidlový systém byl využit například ve známé RTS (*Real-time strategy*) *Age of Empires 2*.

2.1.4 Dělení herní umělé inteligence dle adaptivity

Herní umělé inteligence je možné rozdělit do dvou kategorií dle jejich schopnosti přizpůsobit své rozhodování:

Statické herní inteligence mají rozhodování plně určeno během vývoje hry a v jejím průběhu ho již nemění. Mezi jejich hlavní výhody patří možnost otestovat je před vydáním hry a záruka konzistence jejich kvality po celou dobu hraní. Hlavním problémem je jejich monotónnost, která snižuje hratelnost při opakovaném hraní.

Adaptivní herní inteligence dokáží měnit své rozhodování v průběhu hry a přizpůsobovat se situaci. K tomu je možné využít metody strojového učení z akademické umělé inteligence. Většina vývojářů je však nevyužívá, zčásti kvůli obavám z vyvinutí příliš nebo nedostatečně efektivního chování, zčásti kvůli náročnosti jejich implementace a obtížnému testování [6].

2.1.5 Učení v herní umělé inteligenci

Učení je v herní umělé inteligenci velmi populární téma. Učící se umělá inteligence je teoreticky schopná přizpůsobit se hráči a poskytnout tak konzistentní výzvu, což zajistí dlouhodobou hratelnost hry i v případě, kdy nejsou k dispozici lidští protihráči. Prakticky však ještě učení nesplnilo vysoká očekávání do něj vkládaná. Jeho využití ve hře vyžaduje pečlivé plánování a porozumění nástrahám jednotlivých technik, které se pohybují od jednoduchých úprav číselných parametrů až po komplexní algoritmy, jako jsou například umělé neuronové sítě (výpočetní model, který napodobuje biologické nervové systémy). [3, str. 579]

2.1.5.1 Online a offline učení

Podle doby kdy učení probíhá ho dělíme na [3, str. 579-581]:

Online učení probíhá během hraní proti hráči. Umožňuje tak dynamicky se přizpůsobit jeho hernímu stylu a poskytnout mu výzvu lépe odpovídající jeho schopnostem. Významnými problémy však jsou jeho nepředvídatelnost a obtížnost testování. Umělá inteligence se může naučit naprosto

nepoužitelné chování a replikace tohoto problému za účelem jeho vyřešení může v nejhorším případě vyžadovat přesné zopakování všech akcí provedených hráčem až do doby vzniku problému, což je často zcela nemožné.

Offline učení může probíhat mezi různými úrovněmi hry, častěji však probíhá ještě před jejím vydáním. Představuje většinu učení dnes v herní umělé inteligenci používaného. Spočívá obvykle ve zpracování informací o dříve zahraných hrách a využití získaných informací k úpravě herních strategií nebo parametrů. Umožňuje tak využít i méně předvídatelné algoritmy a provést vyčerpávající testování jejich funkčnosti.

2.1.5.2 Přeučení

Běžným problémem učící se umělé inteligence je přeučení. Pokud je učící se algoritmus často vystavován určitým situacím, může se naučit reagovat odpovídajícím způsobem pouze na ně. Obvykle však naopak chceme, aby byl schopen ze svých zkušeností generalizovat a dokázal se tak vypořádat s širokým okruhem různých situací. [3, str. 582]

2.2 Taskino

Taskino je webová aplikace pro organizaci plnění úkolů z reálného světa, využívající herních prvků k motivaci uživatelů. Jedním z těchto prvků je komplexní strategická hra, vytvořená a posléze rozšířená o počítačové protivníky v rámci dvou předcházejících závěrečných prací – [7] a [8]. Není-li řečeno jinak, označuje dále v této práci název Taskino specificky tuto strategickou hru, ne celou motivační aplikaci.

Následující subsekce ji popisují se zaměřením na prvky relevantní během návrhu vhodné metody adaptivní umělé inteligence.

2.2.1 Základy

Každý hráč ovládá svůj ostrov, může na něm stavět budovy, rekrutovat jednotky, zkoumat nové technologie a vysílat své jednotky na nájezdy na ostatní ostrovy. Hra je tahová, přičemž každý tah představuje v přepočtu na skutečný čas čtyři hodiny. V rámci jednoho tahu může hráč provést libovolný počet stavebních operací, rekrutací i přesunů jednotek. Přepočítání stavu surovin, populace a boje ale probíhají vždy na konci tahu, podle stavu aktuálního v době přepočtu.

Hra využívá komplexní ekonomický systém, obsahující množství produkčních budov, pět druhů surovin, skladovací prostory, populaci, výzkum a modifikátory zaměstnanosti a štěstí obyvatel. Všechny tyto faktory je nutné brát

Obrázek 2.1: Stav herního ostrova umělé inteligence po 150 tazích, důsledek exponenciální rychlosti rozvoje populace

v úvahu pro vytvoření efektivní ekonomiky, schopné rekrutovat dostatek jednotek pro zajištění úspěšné obrany ostrova i nájezdů na ostrovy protivníků, což je netriviální úkol i pro lidského hráče a představuje významnou výzvu při tvorbě kvalitní umělé inteligence.

Pro výstavbu některých typů budov a jednotek je potřeba zkoumat technologie. Cenou za tyto technologie jsou diamanty, které hráč dostává jako odměnu za splněné úkoly z reálného světa. To by mělo hráče motivovat k vyššímu pracovnímu výkonu.

2.2.2 Tempo hry

Vztah populace a produkce je obzvláště důležitý pro návrh a zhodnocení umělé inteligence, protože udává exponenciální tempo růstu síly hráčů. Produkční budovy vyžadují pro svou funkčnost zaměstnance. Není-li dostatek zaměstnanců pro všechny budovy, snižuje se jejich produkce. Nemá tedy valný smysl stavět produkční budovy, když pro ně není dostatek zaměstnanců. Rychlost růstu produkce proto zhruba odpovídá rychlosti růstu populace, která je exponenciální vzhledem k počtu tahů, viz obrázek 2.2. Důsledek je možné vidět v obrázku 2.1.

2.2.3 Boj

Nájezdy na cizí ostrovy mohou sloužit ke zničení nepřátelských armád, loupení surovin nebo k ničení protivníkových budov. Pro omezení rivality mezi

```
if self._happiness >= 80:
    self._population = math.ceil(self._population * 1.05)
elif self._happiness >= 60:
    self._population = math.ceil(self._population * 1.025)
elif self._happiness >= 40:
    self._population = self._population
elif self._happiness >= 20:
    self._population = math.floor(self._population * 0.975)
else:
    self._population = math.floor(self._population * 0.95)
```

Obrázek 2.2: Kód řídící růst populace. Násobení současné populace vede k exponenciální rychlosti růstu.

spolupracovníky způsobené zápasením ve hře může lidský hráč provádět nájezdy pouze na ostrovy počítačových protivníků, což dále zvyšuje vliv kvality umělé inteligence na hratelnost hry.

Přesun jednotek mezi ostrovy je okamžitý. K bitvě dojde, pokud jsou na některém ostrově na konci tahu jednotky nepřátelského hráče.

Ve hře jsou jasně rozlišitelné tři typy jednotek dle jejich využití:

Obranné jednotky jsou určeny hlavně k obraně vlastního ostrova, mají vysokou sílu v obraně a nízkou v útoku.

Útočné jednotky jsou opakem jednotek obranných, mají vysokou sílu v útoku a velmi nízkou v obraně.

Obléhací jednotky umožňují ničit budovy na ostrovech cizích hráčů.

V obraně navíc pomáhají věže, útočící na nepřátelské jednotky ve svém dosahu.

Samotný boj probíhá na bitevním poli odvozeném od napadeného ostrova a jeho budov. Jednotky obránce i útočníka jsou rozděleny do čet, které jsou na začátku bitvy náhodně rozmístěny na ostrově a ve kterých se po něm následně pohybují a bojují s četami protivníka. Náhodné rozmístění vede k unikátnímu průběhu boje v rámci každé bitvy.

2.2.4 Využití umělé inteligence

Herní umělá inteligence je ve hře využita na strategické i taktické úrovni.

Na taktické úrovni je použit multiagentní systém, řídící simulaci bitvy. Adaptivní umělá inteligence na taktické úrovni ovšem postrádá smysl, protože je tento systém společný jak počítačovému, tak lidskému hráči, a lidský hráč s ním nepřichází do kontaktu.

Na strategické úrovni jsou počítačoví protivníci řízeni jednoduchým pravidlovým systémem, určeným ke zpestření hry a zvýšení její hratelnosti. Jednoduchost a statická povaha tohoto pravidlového systému ovšem znamenají, že nedokáže dlouhodobě poskytnout dostatečnou hratelnost a tudíž s časem rychle klesá využitelnost hry jako motivačního prostředku. Tento problém je možné řešit úpravou pravidlového systému na adaptivní umělou inteligenci, která bude schopná přizpůsobit svou strategii strategiím, které jsou proti ní používané.

S ohledem na zjištěné informace je strategie umělé inteligence pro potřeby práce definována mírou produkce jednotlivých surovin, množstvím a typem stavěných vojenských i nevojenských budov, množstvím rekrutovaných jednotek a jejich zaměřením na útok, nebo obranu a vysokou, nebo nízkou mírou útočnosti. Přizpůsobení těchto herních rozhodnutí je hlavním úkolem navrhované adaptivní umělé inteligence.

Úkolem adaptivní inteligence naopak není složité manévrování s armádami, jehož možnosti jsou v Taskinu z několika důvodů velmi omezené. Za prvé, rozhodování umělých inteligencí je spouštěno v každém tahu ještě před rozhodováním lidských hráčů, nemůže tedy reagovat na jejich akce. Za druhé, okamžité přesuny jednotek mezi ostrovy znemožňují načasované útoky a neumožňují ani zachránit menší armádu jejím vysláním mimo ostrov a následným stažením po útoku. Tento nedostatek možností manévrování by bylo možné alespoň částečně řešit například umožněním vlastnictví více ostrovů jediným hráčem, které však nebylo v předchozích pracích funkčně implementováno, případně pomocí inteligence vyšší úrovně, koordinující jednotlivé počítačové hráče. Cílem této práce je však prozkoumat možnosti adaptivního chování jednotlivých hráčů a vyšší úroveň inteligence je tedy mimo její rámec.

Kapitola 3

Vybrané metody adaptivní umělé inteligence

Tato kapitola popisuje zvažované metody adaptivní umělé inteligence a uvádí příklady jejich použití. Metody jsou vybrány z metod popsaných odbornými články, u nichž byla experimentálně prokázána možnost využití na strategické úrovni v komplexních strategických hrách, vyjímaje deskové hry, které představují obor problémů diametrálně odlišných od Taskina.

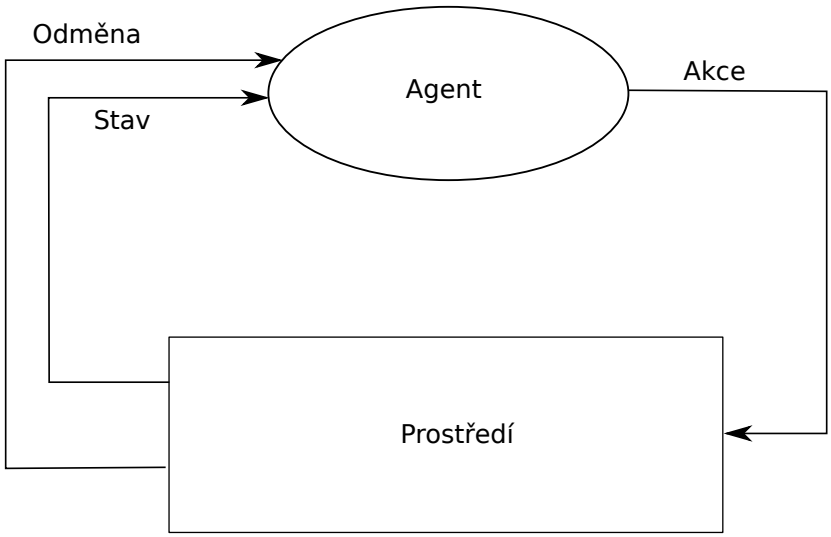
Z uvažovaných metod jsou vyřazeny ty, které pro své fungování vyžadují záznamy dřívějších her. Tyto záznamy totiž pro Taskino neexistují a jejich vytvoření je pro potřeby této práce příliš časově náročné.

3.1 Posilované učení

Posilované učení je soubor technik strojového učení založených na přímé interakci agenta s prostředím za účelem dosažení cíle. Tuto interakci reprezentuje pomocí stavů prostředí, akcí, které může agent ve stavech vykonat, a odměn za vykonané akce, určených dle plnění agentova cíle. Interakci ilustruje obrázek 3.1. [9, str. 15]

Posilované učení typicky určuje pro každou dvojici stav-akce co nejpřesnější odhad odměny získané ve všech následujících krocích, tedy dlouhodobou výhodnost provedené akce. Agent následně ve většině případů volí akci, pro kterou je tento odhad pro daný stav maximální. Techniky posilovaného učení se zpravidla liší hlavně metodou jeho stanovení. [9, str. 6-7]

V [10] je posilované učení aplikováno v jednoduchém bojovém scénáři v populární strategické hře *StarCraft:Broodwar*, článek [11] popisuje jeho úspěšné použití v komplikované bitevní situaci ve hře *Battleground* a konečně článek [12] ukazuje jeho efektivní využití pro volbu předpřipravené vysokoúrovňové strategie nejlépe odpovídající herní situaci ve strategické hře *Civilization IV*.



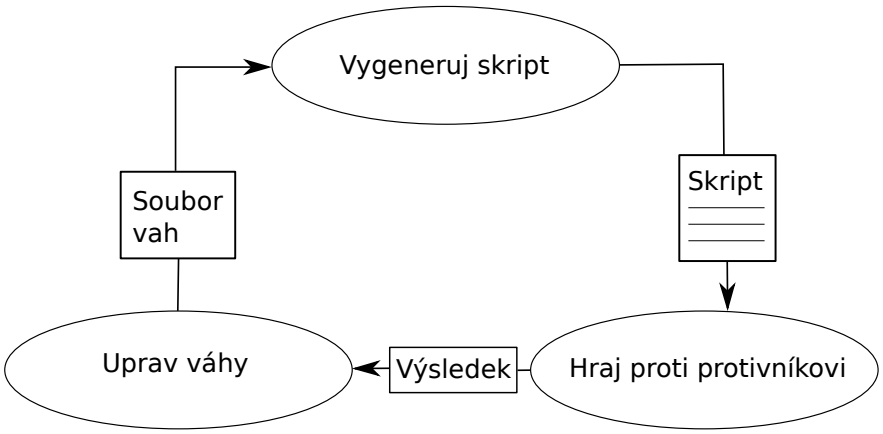
Obrázek 3.1: Model interakce agenta s prostředím, který je základem posilovaného učení

I přes některé úspěšné aplikace a vysoký zájem však posilované učení zatím nebylo pro vývoj her šířeji využito. „*Posilované učení je nejvhodnější pro offline učení. Funguje dobře na problémy s množstvím různých interagujících komponent, jako je optimalizace chování skupiny postav nebo hledání sekvencí na pořadí závislých akcí.*“ [3, str. 642, vlastní překlad] Zvláště efektivní je při vyhodnocování výhodnosti herní situace v deskových hrách. Není však vhodné pro problémy s mnoha různými stavy a problémy, kde se úspěšné strategie postupem času mění. [3, str. 641-643]

3.2 Dynamické skriptování

Dynamické skriptování je metoda *online* strojového učení pro herní umělou inteligenci založená na posilovaném učení. Funguje na principu výběru pravidel pro pravidlový systém z většího souboru pravidel, přičemž pravděpodobnost výběru pravidla je určena jeho váhou, což je hodnota označující jeho kvalitu zjištěnou v průběhu učení. [13]

Učení probíhá mezi zápasy. Na začátku zápasu je ze souboru pravidel vybrán pro umělou inteligenci skript pevně určené velikosti. Na jeho konci je úspěšnost skriptu vyhodnocena. Pokud byl úspěšný, jsou váhy použitých pra-



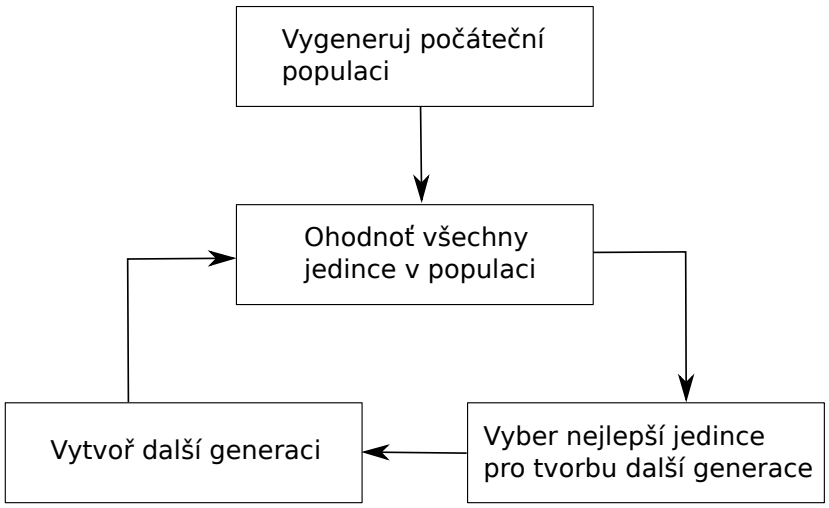
Obrázek 3.2: Základní princip dynamického skriptování

videl zvýšeny, pokud byl neúspěšný, jsou sníženy. Princip fungování dynamického skriptování ilustruje obrázek 3.2, kde soubor vah je pole vah všech pravidel. [13]

Součet vah pravidel je udržován konstantní pomocí kompenzace změny vah použitých pravidel rovnoměrným rozdělením opačné hodnoty mezi zbývající pravidla, což dále urychluje adaptaci a usnadňuje zotavení z neefektivních naučených vah. [13]

Mezi výhody dynamického skriptování popsané v [13] patří:

- využití pravidlových systémů, které umožňuje pochopení a úpravu herní inteligence i neprogramátorům,
- snadné zakomponování znalostí vývojáře o herních mechanikách a strategiích,
- ve srovnání s posilovaným učením velmi rychlé učení,
- snadné přizpůsobení složitosti hry schopnostem hráče, ať už pomocí kvality pravidel, nebo pomocí technik fungujících na principu úpravy změn vah pro příliš náročné počítačové protivníky.



Obrázek 3.3: Obecné fungování evolučních algoritmů. Přeloženo a upraveno autorem, původní zdroj je [1].

Článek [13] popisuje úspěšné využití dynamického skriptování v CRPG (*Computer role-playing game*) hře *Neverwinter Nights*, v [14] je úspěšně aplikováno na taktické úrovni ve strategické hře *World in Conflict* a nakonec diplomová práce [15] používá dynamické skriptování i na strategické úrovni pro RTS *Wargus*, kopii známé RTS *Warcraft II*.

3.3 Evoluční algoritmy

„*Evoluční algoritmy jsou stochastické prohledávací a optimalizační heuristiky odvozené z klasické teorie evoluce, ve většině případů implementované na počítačích. Jejich základní myšlenkou je, že pokud se rozmnožují pouze jedinci splňující určitá výběrová kritéria a ostatní členové populace zemřou, bude populace konvergovat k jedincům, kteří výběrová kritéria splňují nejlépe.*“ [1, str. 4, vlastní překlad]

Obecný proces evolučních algoritmů dále popisuje obrázek 3.3. Jejich hlavním účelem je hledat kvalitní řešení výpočetních problémů, pro které buďto neexistují exaktní algoritmy, nebo jsou příliš výpočetně náročné. Populaci z výše citované definice tak tvoří různá řešení řešeného problému. [1]

Existuje mnoho různých variant evolučních algoritmů, patří mezi ně například:

Genetické algoritmy kladou velký důraz na kódování řešení množinou genů, obvykle pomocí binárních řetězců. Představují tak nejvěrnější přenesení procesů evoluce v přírodě do počítačových systémů. Nová populace je tvořena z jedinců vybraných ze staré populace kombinovaných a upravených pomocí operátorů křížení a mutace. Křížení vytváří nové jedince kombinací více starých jedinců, zatímco mutace reprezentuje náhodné kopírovací chyby při kopírování DNA. [1, str. 6-8]

Evoluční strategie jsou specializovány na optimalizaci reálných čísel, která mění přímo, na rozdíl od genetických algoritmů, které by je měnily v jejich kódované reprezentaci. Operátory evoluční strategie tak umožňují lépe přizpůsobit prohledávání prostoru řešení konkrétnímu problému. [1, str. 9-10]

Genetické programování je nejúspěšnější z evolučních algoritmů navržených pro automatické generování programů. Kvalita jedince je v nich vyhodnocována pomocí úspěšnosti běhu navrženého programu. [1, str. 10-11]

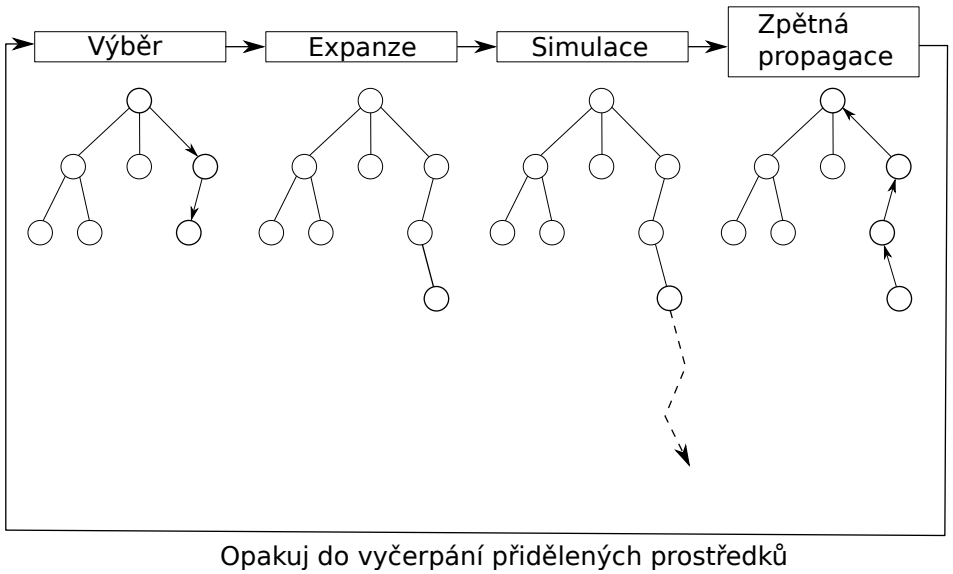
Neuroevoluce je kombinací evolučních algoritmů a umělých neuronových sítí. Populace neuronových sítí je optimalizována evolučními algoritmy pro nalezení neuronové sítě, která dokáže dostatečně kvalitně řešit zadaný problém. Vyhodnocení kvality jedince, podobně jako u genetického programování, typicky spočívá ve zhodnocení výsledků běhu neuronové sítě jím reprezentované. [16]

Článek [17] ukazuje možnost úspěšného *online* použití neuroevoluce ve hře *Wargus*. V [15] je popsáno *offline* využití evolučního algoritmu pro generování nových strategií a taktik tamtéž. V [18] je genetický algoritmus použit pro *online* učení ve hře *invAIders*, navržené specificky za tímto účelem.

3.4 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) je metoda hledání optimálních rozhodnutí pomocí vytváření vyhledávacího stromu s využitím náhodných simulací. Při každém rozhodnutí je vytvářen strom, jehož kořenem je stav, ve kterém se agent právě nachází. Každý uzel představuje stav prostředí, do kterého se agent svými akcemi může dostat, každá hrana představuje akci, kterou může provést. [2]

Po vytvoření kořene je strom až do vyčerpání přiděleného výpočetního času, paměti, nebo určeného počtu iterací iterativně rozšiřován o nové listy. Přidaný list je ohodnocen pomocí náhodné simulace dalšího vývoje hry ze



Obrázek 3.4: Základní iterativní proces MCTS. Přeloženo a upraveno autorem, původní zdroj je [2].

stavu, který reprezentuje. Tato simulace může probíhat do konce hry, případně do jinak určeného konečného stavu. Ohodnocení listu je pak zpětně propagováno do jeho předků. Iterativní rozšiřování stromu znázorňuje obrázek 3.4. [2]

Po poslední iteraci algoritmu je zvolena vhodná akce podle akumulovaných ohodnocení. Podmínkou fungování MCTS tedy je, aby skutečná kvalita akce mohla být odhadnuta pomocí simulace jejích následků. [2]

Nejpopulárnější oblastí využití MCTS jsou hry, nejznámější z nich v deskové hře *Go*. Aplikace MCTS se však rozhodně neomezují na deskové hry. Metoda již byla úspěšně vyzkoušena v komplexních strategických hrách, například na taktické úrovni ve výše zmiňované hře *Wargus* [2]. MCTS byl dokonce použit i jako součást základní umělé inteligence v komerčně vydané strategické hře *Total War: Rome II*, kde rozhoduje o distribuci zdrojů na strategické úrovni hry [19].

Kapitola 4

Návrh a implementace

První sekce této kapitoly se zabývá zhodnocením použitelnosti uvažovaných metod adaptivní herní inteligence v Taskinu na základě informací popsaných v předchozí kapitole a výběrem jedné z nich. Další sekce se zabývají popisem použitých technologií, úpravami Taskina vhodnými pro jeho hratelnost nebo potřebnými pro úspěšnou implementaci a vyhodnocení použitelnosti vybrané metody a nakonec jejím přizpůsobením pro Taskino a její konkrétní implementací.

4.1 Výběr metody

MCTS vyžaduje simulace dopadů uvažovaných akcí. V Taskinu by tyto simulace však byly velmi časově a paměťově náročné i v případě uvažovaného případu dvou hráčů, nemluvě o využití v praxi, kde by tato náročnost dále rychle rostla spolu s počtem hrajících hráčů. Navíc by tyto simulace vzhledem ke složitosti hry a náhodnosti do ní vnášené bojovým systémem nebyly schopny odhadnout skutečnou kvalitu akce s dostatečnou přesností. Dosažitelnou přesnost ještě více snižuje fakt, že hráč nemá přístup k velké části informací o svých protivnících. Z těchto důvodů není MCTS pro Taskino vhodné.

Evoluční algoritmy by dle zkoumané literatury bylo možné efektivně využít pro *offline* učení, například pro evoluci skriptů, které by zvýšily hratelnost zvýšením variability protivníků řízených umělou inteligencí, cílem práce je ale *online* učení. Nalezené případy *online* učení pomocí evolučních algoritmů vyžadují množství generací, které je příliš vysoké pro jejich praktické využití v Taskinu, zvláště s ohledem na omezení jeho rychlosti čtyřmi hodinami na tah.

Posilované učení je pro *online* adaptaci vhodnější, možnosti jeho aplikace v Taskinu však narušuje velmi vysoký počet možných stavů a akcí již v uvažovaném případě dvou hráčů, natož pak v případě počtu vyššího, a fakt, že se úspěšná strategie může snadno změnit rozhodnutími lidských protihráčů.

Poslední ze zvažovaných metod je dynamické skriptování, které umožňuje *online* učení, má ve srovnání s ostatními metodami nízké časové i paměťové nároky, je schopné rychlé adaptace a umožňuje využít v Taskinu již existující pravidlový systém. Vzhledem k jeho obvyklému využití pro učení na konci zápasů je však potřeba ho pro použití v Taskinu upravit. Ze srovnání s ostatními uvažovanými metodami adaptivní herní umělé inteligence přesto vychází nejlépe.

4.2 Použité technologie

Práce navazuje na [7] a [8], použité technologie jsou proto stejné. Je jich velké množství, pro tuto práci jsou však důležité hlavně dvě:

Python je interpretovaný objektově orientovaný vysokoúrovňový programovací jazyk. Zabudované vysokoúrovňové datové struktury a dynamické typování dělají Python velmi atraktivním nejen pro rapidní vývoj aplikací, ale i jako skriptovací jazyk. Jeho jednoduchá a snadno naučitelná syntaxe je zaměřená na vysokou čitelnost, čímž usnadňuje údržbu programů. [20]

V současné době se používají dvě hlavní verze, Python 2 vydaný v roce 2000 a Python 3 vydaný v roce 2008. Verze jsou vzájemně nekompatibilní a mnoho projektů proto stále využívá Python 2, třebaže jeho vývoj a podpora skončí již v roce 2020. [21]

Vzhledem k tomu, že [8] využívá Python 2.7, je v něm vyvíjena a testována i implementační část této práce.

MongoDB je open-source dokumentově orientovaná databáze poskytující vysoký výkon, dostupnost a automatickou škálovatelnost. Záznamy jsou ukládány v podobě dokumentů, datových struktur složených z párů polí a hodnot. Dokumenty jsou podobné JSON objektům, díky čemuž jsou snadno zpracovatelné mnohými programovacími jazyky, včetně jazyka Python. [22]

4.3 Úpravy hry

Následující subsekce popisují v rámci této práce provedené úpravy, opravy a vylepšení Taskina, některé nutné pro úspěšnou implementaci dynamického skriptování, jiné pouze vhodné podle zjištěných informací o této hře.

4.3.1 Cyklické zpracování souboru pravidel

Původní pravidlový systém zpracovává skript vytvořený přeložením pravidel do jazyka Python pouze jednou, viz obrázek 4.1, a pravidla jsou vyhodnocována a případně vykonávána v pořadí, v jakém se ve skriptu nacházejí. Tento

```
try:
    exec self._ruleset
except EndException:
    pass
```

Obrázek 4.1: Původní způsob zpracování skriptu umělé inteligence

```
continue_executing = True
while continue_executing:
    continue_executing = False
    exec self._ruleset
```

Obrázek 4.2: Vylepšený způsob zpracování skriptu umělé inteligence

přístup je vhodný pro původně zamýšlenou výrazně oslabenou umělou inteligenci, nehodí se však pro umělou inteligenci zaměřenou na poskytnutí výzvy soutěživým hráčům, protože příliš výrazně omezuje počet provedených operací na tah a nedovoluje provedení pravidel v jiném pořadí, než v jakém jsou uvedena ve skriptu, což výrazně snižuje kvalitu výsledné umělé inteligence. Tento problém je ve vylepšeném pravidlovém systému vyřešen zajištěním opakovaného provádění skriptu, dokud je vykonáno alespoň jedno z pravidel, viz obrázek 4.2.

4.3.2 Rozšíření pravidlového systému

Původní pravidlový systém obsahuje pouze nízké množství pravidel, a tedy i podmínek a akcí, ze kterých jsou tato pravidla složena. Pro efektivní implementaci dynamického skriptování je potřeba, aby byla pravidla dostatečně různorodá. Nový pravidlový systém proto nabídku možných podmínek a akcí rozšiřuje z pouhých 18 na více než 50, čímž umožňuje vytváření velkého množství různorodých a efektivních pravidel. Některé z těchto podmínek a akcí nejsou v konečném návrhu pravidel pro herní inteligence využity, protože pravidla s nimi vytvořená jsou pro účely prozkoušení navržené metody příliš nebo nedostatečně kvalitní.

4.3.3 Militarizace a agresivita

Dalším z vylepšení je systém dvou proměnných, umožňujících rychlé, ale velmi jednoduché přizpůsobení chování i bez využití algoritmů strojového učení. Obě proměnné se mění vždy na konci bitvy, na základě úspěšnosti armády umělé inteligence, která je posuzována podle vzájemně způsobených ztrát.

```
(defrule
    (militarization-dice-roll)
    (try-building-unit-by-aggresivity)
=>
    {0}
)
```

Obrázek 4.3: Ukázka pravidla využívajícího militarizaci a agresivitu

Tabulka 4.1: Změny militarizace v závislosti na akcích umělé inteligence

	Úspěšně	Neúspěšně
Útočí	+1	−1
Brání se	−1	+1

Tabulka 4.2: Změny agresivity v závislosti na akcích umělé inteligence

	Úspěšně	Neúspěšně
Útočí	+1	−1
Brání se	+1	−1

První z nich je militarizace, reprezentující výhodnost útočných jednotek v útoku a potřebu obranných jednotek v obraně. Využívána je v pravidlech zabývajících se stavbou vojenských budov a rekrutací jednotek, obvykle jako procentuální pravděpodobnost provedení těchto akcí. Například v pravidle v obrázku 4.3 by hodnota militarizace 60 vedla k šedesátiprocentní pravděpodobnosti, že se pravidlo pokusí rekrutovat jednotky. Jejím hlavním úkolem tedy je upravovat míru rekrutace jednotek. Pohybuje se mezi 40 a 60 včetně a její počáteční hodnota je 40. Její změny v závislosti na výsledku bitvy jsou ilustrovány tabulkou 4.1.

Druhou je agresivita, upravující zaměření na útok nebo obranu. Využívána je obvykle v pravidlech rozhodujících se o typu rekrutovaných vojenských jednotek, například viz obrázek 4.3. Hodnota agresivity 60 by při zpracování ukázaného pravidla znamenala, že pravidlo s šedesátiprocentní pravděpodobností rekrutuje útočné jednotky a s čtyřicetiprocentní pravděpodobností jednotky obranné. Agresivita se pohybuje mezi 40 a 60 včetně a její počáteční hodnota je 50. Její změny v závislosti na výsledku bitvy jsou ilustrovány tabulkou 4.2.

4.3.4 Statické inteligence

Po změně pojetí umělé inteligence ve hře z nenáročného rozptýlení na výzvu poskytujícího soupeře se původní skript řídící počítačové protivníky ukazuje být nedostačujícím. Změny provedené v pravidlovém systému nyní dovolují

tvorbu podstatně kvalitnějších statických soupeřů. Podle průzkumu hry byly navrženy tři základní strategie:

Stavitel se zabývá převážně budováním ekonomiky a budov, v menší míře však rekrutuje i jednotky a občas i zaútočí na ostatní hráče. Je nejméně předvídatelný, protože má potenciál být agresivní i defenzivní.

Obránce preferuje budování na svém ostrově chráněné mohutnou obrannou armádou a obrannými věžemi. Příležitostně ovšem rekrutuje i útočné jednotky a výjimečně útočí na ty, kdo ho napadají.

Útočník se snaží získat výhodu častými útoky na své soupeře, podceňuje přitom ovšem svou vlastní obranu a ekonomiku.

Každá z těchto strategií byla zpracována v podobě skriptu pro pravidlový systém. Ačkoli tyto umělé inteligence jsou již schopny mnohem lépe využívat potenciálu hry, jsou stále omezeny kvalitou svých ekonomických pravidel. Důvodem k tomu je hlavně omezení doby experimentů, aby bylo možné v rozumném čase testovat efektivitu dynamického skriptování. Experimenty se statickými inteligencemi totiž ukazují, že i přes níže popsané optimalizace stoupá doba výpočtu úměrně počtu postavených budov a množství jednotek účastnících se bitev. Ekonomicky omezené umělé inteligence také lépe reprezentují méně aktivní styl hraní, který se od hráčů Taskina očekává s ohledem na to, že jde o pracovní nástroj.

4.3.5 Úpravy herních mechanik

Průzkum herních mechanik během rozšiřování pravidlového systému vedl ke dvěma úpravám.

První z nich je změna ceny základních produkčních budov. Dřevorubec, produkující dřevo, i lovec, produkující kůži, vyžadují k výstavbě surovinu, k jejíž produkci jsou určeni. To umožňuje snadné zaseknutí ekonomického rozvoje v začátcích hry jak pro umělou inteligenci, tak pro začínajícího lidského hráče. Ceny proto byly upraveny tak, aby dřevorubec vyžadoval pouze kůži a lovec pouze dřevo.

Druhá se týká vojenských budov, jejichž stavění je omezeno tím, že zaměstnávají populaci. Pro zvýšení vojenského potenciálu hráčů byla tato potřeba odstraněna.

4.3.6 Optimalizace výpočtů

Původní pravidlový systém obsahuje neefektivní vyhodnocení některých podmínek. To není problémem při značně omezeném množství vyhodnocovaných pravidel při jediném průchodu skriptem. Při cyklickém zpracování, vyhodnocujícím v rámci každého tahu podstatně větší množství pravidel, však již tyto podmínky výrazně prodlužují dobu vyhodnocení umělé inteligence.

```
def is_resource_blocked(self, amount=0):
    storage = 0
    buildings = self._island.get_buildings()
    for building in buildings:
        if ACTIONS.STORAGE in buildings_detail[building['type']]['actions']:
            storage +=
↪   buildings_detail[building['type']]['actions'][ACTIONS.STORAGE]

    resources = sum(self._island.get_resources().itervalues())
    if resources + amount < storage:
        return False
    return True
```

Obrázek 4.4: Ukázka původní implementace jedné z možných podmínek pro pravidlový systém. Při každém ověření podmínky je nutné projít všechny budovy, což je při vyšším počtu budov velmi náročné.

```
# Ve třídě Island
def get_storage_space(self):
    if self._storage_space is None:
        self._storage_space = 0
        for building in self._buildings:
        if ACTIONS.STORAGE in buildings_detail[building['type']]['actions']:
            self._storage_space +=
↪   buildings_detail[building['type']]['actions'][ACTIONS.STORAGE]

    return self._storage_space

# Ve třídě DecisionEngine
def is_resource_blocked(self, amount=0):
    storage = self._island.get_storage_space()

    resources = self._island.get_resource_sum()
    if resources + amount < storage:
        return False
    return True
```

Obrázek 4.5: Ukázka optimalizované implementace jedné z možných podmínek pro pravidlový systém. Hodnota je při prvním dotazu uložena a následně je pouze vracena.

```
# Funkce volaná v případě poškození čety
def recalculate_resources(self):
    # Hodnota představuje volnou kapacitu čety pro nesení surovin.
    haul = self.get_haul()
    if haul >= 0:
        return

    total = self.get_stolen_resources_amount()
    for res, amount in self._resources.iteritems():
        percentage = float(amount) / total

        # Problematický řádek, haul je zde totiž vždy záporný.
        # Tento řádek proto přisoudí surovině větší množství než měla před
↪   přepočtem.
        res_count = round((total - haul) * percentage)

        self._resources[res] = res_count
```

Obrázek 4.6: Chybná funkce v původní verzi hry. Přepočítání množství nesených ukradených surovin v případě poškození čety chybně vede k navýšení tohoto množství.

Neefektivita je způsobena opakováním náročných výpočtů při vyhodnocení každého z pravidel. Pro ukázku byla vybrána podmínka *resource-blocked*, vyhodnocovaná funkcí *is_resource_blocked()*, viz obrázek 4.4.

Součástí práce je optimalizace těchto výpočtů pomocí uložení vypočítané hodnoty a následného vracení uložené hodnoty, přepočítávané pouze při akcích, které vedou k její změně, a jejich přesunutí do vhodných datových objektů. Pro ilustraci optimalizace viz obrázek 4.5.

4.3.7 Oprava chyb

Průzkum pravidlového systému a následné testování vytvořených umělých inteligencí odhalily několik chyb nenalezených předcházejícími autory Taskina, které byly v rámci práce opraveny.

Nejzávažnější z nich je chyba v simulaci bitvy, způsobující zvyšování množství četou nesených nakradených surovin v případě, kdy je četa poškozena, ale ne zcela zničena. Chybná funkce je ukázána a okomentována v obrázku 4.6.

Druhá chyba způsobovala, že pokud byla četa schopna ukradnout všechny suroviny, které se nacházely v plundrovaném skladu, přidala tyto suroviny ke svým nakradeným surovinám, již je ale neodečetla ze skladu. To znovu vedlo k přehnaně vysokým příjmům z rabování, neboť čety kradly z těchto skladů dokud všechny nenesly plnou nosnost ukradených surovin.

4.4 Dynamické skriptování

Tato sekce popisuje přizpůsobení dynamického skriptování nutné pro jeho využití v Taskinu a jeho implementaci a nastavení parametrů použité během experimentů popsaných v následující kapitole. Sekce nepopisuje základní fungování dynamického skriptování, protože to již bylo vysvětleno v sekci 3.2.

4.4.1 Přizpůsobení pro Taskino

Příklady využití dynamického skriptování ve zkoumané literatuře se učí vždy na konci úseku hry, který ho využívá, například po zápasu mezi týmy v [13] nebo celé hře ve strategické hře, jako v [15]. Toto použití by však v praxi v Taskinu nebylo možné ze dvou důvodů. Prvním z nich je, že hra v Taskinu nemá žádné ukončovací podmínky, a tento úsek tak může trvat libovolně dlouho. Druhým je velmi dlouhá doba hraní. Každý tah trvá čtyři hodiny, pouhých 180 tahů již zhruba měsíc. Aby byla úprava strategie prospěšná pro hratelnost, je tedy nutné, aby k ní docházelo již po poměrně nízkém počtu tahů.

Z těchto důvodů tato práce navrhuje upravenou verzi algoritmu, navrženou pro adaptaci přímo v průběhu hry. Upravený algoritmus dělí probíhající hru na úseky učení. Na začátku každého úseku je vybrán nový skript pro umělou inteligenci, na jeho konci je úspěšnost skriptu zhodnocena a provedena změna vah. Jako úsek učení pro Taskino bylo zvoleno patnáct tahů, doba umožňující vzhledem k vysokému tempu hry projevení efektu vojenských i ekonomických pravidel a zároveň dovolující rychlou adaptaci.

Změnu vah použitých pravidel po této úpravě není možné počítat pouze z hodnocení současného stavu hry, protože by nebrala v úvahu stav, ve kterém se hráč nacházel na začátku současného úseku učení. Nebyla by tak schopna objektivně hodnotit efektivitu použitých pravidel. Počítá se proto z rozdílu hodnocení současného stavu hry a hodnocení stavu hry na konci předchozího úseku učení.

Návrh algoritmu tedy předpokládá, že pravidla, která jsou často krátkodobě efektivní, budou efektivní i dlouhodobě. Tento předpoklad je možné podpořit dvěma argumenty:

1. Vzhledem ke zjištěným možnostem adaptace algoritmus nehodnotí složité plány, ale hlavně míru stavby jednotlivých budov, míru rekrutace různých typů jednotek a útočnost, jejichž efekt se projeví podstatně rychleji.

2. Taskino je velmi rychlé. Vzhledem k exponenciální rychlosti růstu, vysvětlené v subsekci 2.2.2, umožňuje hra během patnácti tahů zvýšit populaci násobením $1{,}05^{15}$, tedy ji více než zdvojnásobit. Stavba budov je okamžitá, stejně jako rekrutace jednotek a jejich přesuny mezi ostrovy. Rekrutované jednotky je tak nejen možné, ale i potřebné využít velmi

rychle, než získá protivník soustředící se na ekonomiku příliš velkou ekonomickou výhodu.

4.4.2 Implementace

Dynamické skriptování bylo implementováno v modulu řídícím herní umělou inteligenci v Taskinu, vytvořeném v [8]. Zdrojové kódy Taskina s implementovaným dynamickým skriptováním, včetně v této práci vytvořených souborů pravidel pro jednotlivé statické inteligence a pro samotné dynamické skriptování, je možno nalézt na elektronickém médiu přiloženém k práci.

Taskino již obsahuje pravidlový systém potřebný pro implementaci dynamického skriptování, vylepšený v rámci této práce provedenými úpravami popsanými v předchozí sekci. Metody pro výběr skriptu a změnu vah byly implementovány dle pseudokódu uvedeného v [13], s dále popsanými úpravami.

4.4.2.1 Soubor pravidel

Pravidla v souboru pravidel jsou zásadní pro efektivitu dynamického skriptování a kvalitu umělé inteligence jím vytvářené. Nejprve je nutné zvolit jejich komplexitu. Na jednu stranu mohou být pravidla velmi složitá, obsahující celé strategie od postupů pro vybudování produkce až po její využití k vojenským účelům. Na druhou stranu mohou být i poměrně jednoduchá, například stavba produkční budovy při nedostatku suroviny kterou produkuje. Pro tuto práci byla zvolena poměrně jednoduchá pravidla, za účelem zajištění vysoké variability generovaných skriptů.

Aby dynamické skriptování využívalo pravidla podobné kvality jako statické inteligence, proti kterým je v experimentech testováno, byl soubor pravidel vytvořen z pravidel z nich převzatých a pravidel obdobné kvality. Obsahuje pravidla pro:

- vyšší i nižší produkci surovin,
- časté i méně časté budování ubytování i vojenských budov,
- častou i méně častou rekrutaci jednotek obranných, útočných, náhodného typu, typu ovlivněného agresivitou i jednotek obléhacích,
- různé určení cíle útoku a velikosti útočné armády, která je k němu potřeba.

Díky této rozmanitosti se může umělá inteligence chovat mnoha různými způsoby. Soubor pravidel neobsahuje žádná pravidla pro výzkum technologií, z důvodů popsaných v sekci 5.1. Obrázek 4.7 obsahuje ukázku čtyř pravidel, umožňujících častou i méně častou rekrutaci útočných nebo obranných jednotek.

Některá z pravidel vedoucích ke střední produkci surovin se v souboru nacházejí vícekrát. Hlavním důvodem je snaha podpořit preferenci vojenského

```
(defrule
    (militarization-dice-roll)
    (try-building-offensive-unit)
=>
    {0}
)
;
(defrule
    (militarization-dice-roll)
    (try-building-defensive-unit)
=>
    {0}
)
;
(defrule
    (militarization-dice-roll)
    (probability-dice-roll 20)
    (try-building-offensive-unit)
=>
    {0}
)
;
(defrule
    (militarization-dice-roll)
    (probability-dice-roll 20)
    (try-building-defensive-unit)
=>
    {0}
)
```

Obrázek 4.7: Ukázka vojenských pravidel ze souboru pravidel

přizpůsobení nad pouhou optimalizací ekonomiky. Vícenásobná přítomnost navíc umožňuje použít pravidlo vícekrát v jediném průchodu, případně pravidla umístit ve skriptu v různém pořadí.

Soubor obsahuje celkem 57 pravidel, z toho 52 unikátních, z nichž je jich pro každý generovaný skript voleno 22.

4.4.2.2 Určení výše změny vah

Jednou z nejdůležitějších součástí algoritmu je určení výše změny vah. To v této práci závisí na ohodnocovací funkci, maximální odměně, maximálním potrestání, počáteční váze, minimální váze, maximální váze a rychlosti učení. Funkce změny vah by měla odpovídat kvalitě vybraného skriptu, je tedy přinejmenším nutné, aby byla kladná pro efektivní skript a záporná pro neefektivní. Je jednou z částí algoritmu nejvýrazněji přizpůsobených pro použití v Taskinu.

Ohodnocovací funkce určuje kvalitu současného stavu hry z pohledu hráče upravujícího své váhy. V případě učení na konci zápasu, jako v [13], by byla ohodnocovací funkce přímo použita k určení změny vah. Při učení po určitém počtu tahů je však potřeba počítat i se stavem, ve kterém hodnocený skript hru převzal. Pro změnu vah je proto použit rozdíl současné hodnoty ohodnocovací funkce a hodnoty vypočítané při předchozí změně vah. Jako ohodnocovací funkce byl zvolen vztah

$$F = \frac{S_h}{\bar{S}},$$

kde S_h označuje skóre adaptivního hráče a $\bar{S}$ průměrné skóre všech hráčů. Skóre je hodnota, kterou Taskino využívá pro hodnocení úspěchu hráče, počítaná vztahem

$$S_h = 0{,}3 \cdot U + 0{,}2 \cdot B + 0{,}2 \cdot J + 0{,}3 \cdot T,$$

kde U je množství uskladněných surovin, B je množství surovin utracených za všechny stojící budovy, J je množství surovin utracených za žijící jednotky a T je množství diamantů utracených za výzkum technologií. Každý hráč má přístup ke skóre všech ostatních hráčů, ohodnocovací funkce tedy využívá pouze všeobecně dostupné informace, které zároveň uceleně informují o stavu všech hráčů.

Skutečná změna vah pravidel hráče se pak počítá pomocí vztahu

$$Z = R \cdot (F - F_0),$$

kde F je současná hodnota ohodnocovací funkce, F_0 je hodnota ohodnocovací funkce z konce předchozího úseku učení a R je rychlost učení.

Původním záměrem při implementaci rychlosti učení byly její změny v průběhu učení, umožňující přizpůsobení zpomalit v případě, kdy adaptivní hráč vítězí a zabránit tak přeučení, nebo naopak zrychlit v případě, kdy adaptivní hráč prohrává a podpořit tak zotavení z naučeného neefektivního souboru vah. Tento záměr nakonec nebyl v rámci práce doveden do funkční podoby. Rychlost učení proto slouží pouze pro úpravu rychlosti přizpůsobeni při návrhu algoritmu bez nutnosti měnit počáteční, minimální a maximální váhu. Podle pozorování předběžných experimentů je nastavena na 10, což vzhledem k níže popsanému rozsahu vah a pozorovaným změnám ohodnocovací funkce umožňuje kompletní změnu naučených vah již během několika desítek učících příležitostí. Příliš nízká hodnota tohoto parametru by vedla k pomalému přizpůsobení, příliš vysoká hodnota ke snadnému poškození již naučených efektivních vah.

Počáteční váha je hodnota, na kterou jsou váhy nastaveny na začátku procesu učení. Minimální a maximální váha omezují rozsah, ve kterém se během učení mohou váhy pravidel pohybovat. Počáteční váha je nastavena na 12,5, minimální a maximální váha na 5 a 35. Cílem tohoto poměrně malého rozsahu (ve srovnání například s rozsahem použitým v [13]) je podpořit rozmanitost

generovaných skriptů. Nenulová minimální váha také umožňuje rychlejší zotavení v případě, kdy se změní strategie protivníků.

Posledními hodnotami upravujícími změnu vah jsou maximální odměna a maximální potrestání. Tyto hodnoty omezují změnu vah v obou směrech a jejich cílem je zabránit příliš vysokým změnám vah v případě náhlých vysokých změn skóre. Jejich hodnoty jsou nastaveny na 2 pro odměnu a -2 pro potrestání.

4.4.2.3 Rozdělení souboru pravidel na podmnožiny dle funkce

Soubor pravidel obsahuje pravidla mnoha různých typů, od produkce různých surovin až po vojenská pravidla. Pro vytvoření kvalitní umělé inteligence však je potřeba zkombinovat pravidla všech typů. Například, pokud umělá inteligence zcela vynechá některý typ surovin, nebude schopna podporovat rekrutaci armády nezávisle na tom, kolik a jak kvalitních pravidel pro ni zvolí. Dynamické skriptování za této situace nemůže fungovat, protože i skript z většiny obsahující pravidla vhodná pro současnou situaci může snadno dosáhnout negativních výsledků a snížit tak jejich váhy, třebaže pro úspěch učení je nutný opak.

V rámci této práce implementovaná úprava tento problém řeší rozdělením souboru pravidel na podmnožiny dle jejich funkce. Pravidla jsou tedy rozdělena například na pravidla zajišťující produkci určité suroviny, pravidla zajišťující rekrutaci jednotek nebo pravidla zajišťující stavbu ubytování. Z každé z těchto podmnožin je vybírán pevně daný počet pravidel při každém generování skriptu. Kombinací zvolených pravidel ze všech podmnožin pak vzniká celkový skript. Změny vah jsou prováděny zvlášť na každé z podmnožin, učeny jsou však všechny ve stejné chvíli a s pomocí stejné funkce změny vah.

Kapitola 5

Experimenty a vyhodnocení

Tato kapitola popisuje experimentální vyhodnocení kvality navržené adaptivní umělé inteligence. Její první sekce popisuje postup experimentů k tomuto vyhodnocení použitých. Druhá sekce popisuje úspěšnost adaptivní inteligence v průběhu experimentů. Třetí sekce blíže zkoumá, jakým způsobem se adaptivní inteligence během těchto experimentů chovala. Závěrečná sekce diskutuje zjištění z předcházejících sekcí, vyvozuje z experimentů závěry týkající se herních mechanik Taskina a na základě těchto závěrů doporučuje změny určené ke zvýšení jeho hratelnosti.

5.1 Postup experimentů

Každý z provedených experimentů zkoumá přizpůsobení navržené adaptivní inteligence proti jedné ze statických inteligencí popsaných v subsekci 4.3.4. Tyto inteligence – Stavitel, Obránce a Útočník – jsou v této kapitole pro přehlednost zvýrazněny velkým počátečním písmenem.

Experiment je tvořený sérií po sobě hraných her, během které se hráč řízený adaptivní inteligencí postupně přizpůsobuje chování hráče řízeného inteligencí statickou. Protože Taskino nemá žádné ukončovací podmínky, bylo by možné, aby experiment tvořila jediná hra. Ta by ale musela být velice dlouhá, aby měl adaptivní hráč dostatečný čas k přizpůsobení a toto přizpůsobení mělo čas se projevit. To není vhodné z následujících důvodů:

1. Při exponenciálním růstu populace dokáží umělé inteligence vyčerpat prostor k výstavbě na svém ostrově již během poměrně krátké doby. Vyčerpání stavebního prostoru odstraňuje velkou část komplexity hry, která se následně pro lidské hráče stává prakticky nehratelnou, protože zbývá pouze rekrutace jednotek a jejich vysílání na protivníka. Je proto zbytečné zkoumat chování adaptivního hráče v takové situaci.

2. Už po krátké herní době může jeden z hráčů získat výhodu příliš vysokou na to, aby ji jeho protivník mohl zvrátit. Zkoumat přizpůsobení adap-

Tabulka 5.1: Podíl her vyhraných dynamickým skriptováním proti Útočníkovi (čtyři experimenty pro obě varianty)

	Minimum	Průměr	Maximum
Bez učení	0,32	0,42	0,48
S učením	0,48	0,64	0,76

tivního hráče v takové situaci je zbytečné, protože buďto je tím slabším a přizpůsobením si nijak nepomůže, nebo je tím silnějším a přizpůsobení není důležité pro jeho úspěch.

Délka jednotlivých her byla stanovena na 150 tahů, dobu dostatečně dlouhou pro zajímavé zápasy a zároveň dostatečně krátkou na to, aby k případnému vyčerpání prostoru k výstavbě docházelo až ke konci hry.

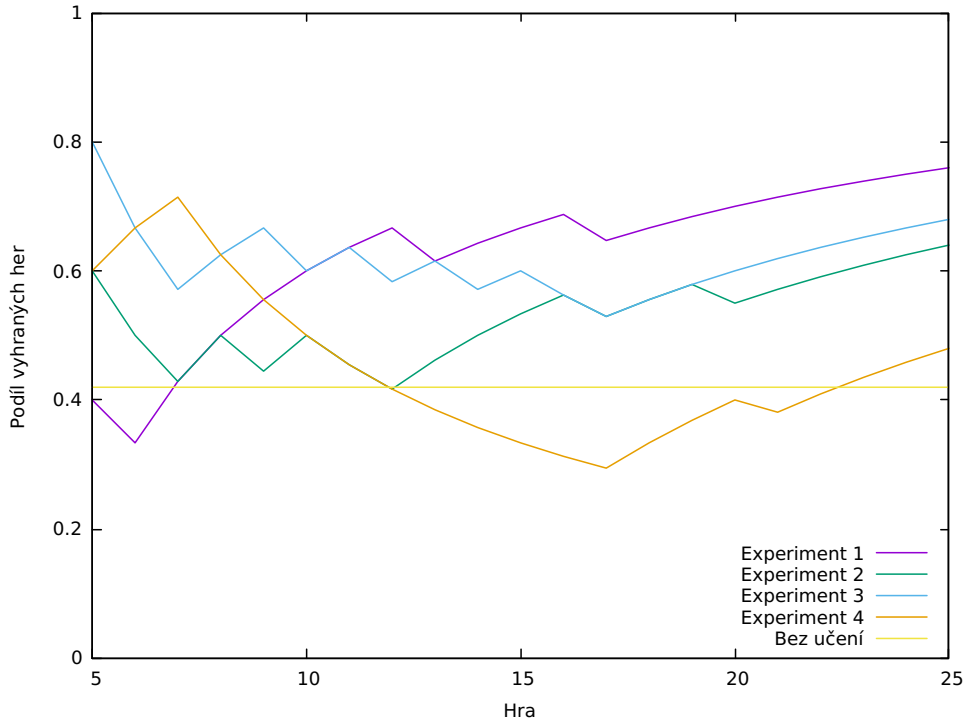
Aby bylo možné v průběhu každé hry vyzkoumat všechny technologie a hráči tak mohli využívat všechny prostředky dostupné ve hře, bylo v každém tahu přiděleno každému hráči deset diamantů. To umožňuje vyzkoumat všechny technologie již před stým tahem. Pořadí výzkumu je stejné pro všechny inteligence, statické i adaptivní, protože nijak výrazně neodlišuje jejich strategii a stejné pořadí umožňuje vyhodnotit možnosti výzkumu ještě před cyklickým zpracováním souboru pravidel, které by pouze zbytečně zpomalovaly.

Pro experiment bylo zvoleno 25 her, počet dostatečně vysoký pro zkoumání efektu dynamického skriptování (poskytuje 250 příležitostí k učení, což by pro něj měl být více než dostatečný počet) a zároveň zkoumající přiměřené období, které by vzhledem ke čtyřem hodinám na tah v produkční verzi Taskina trvalo méně než dva roky. Délka experimentů byla také omezena kvůli jejich vysoké výpočetní a časové náročnosti, aby bylo možné pozorovat jich vyšší množství a vyvozovat pak z jejich průběhu závěry.

Proti každé ze statických inteligencí byly provedeny čtyři experimenty, jejichž výsledky a průběh jsou v následujících sekcích popisovány a zkoumány. Aby bylo možné vyčíslit efektivitu přizpůsobení adaptivního hráče, byly proti každé ze statických inteligencí ještě provedeny čtyři stejně dlouhé experimenty zjišťující úspěšnost neučícího se dynamického skriptování. To funguje stejně jako jeho učící se varianta, ale nemění váhy pravidel, které tak zůstávají v průběhu všech her na svých počátečních hodnotách.

5.2 Výsledky experimentů

Podle podílů vyhraných her v tabulkách 5.1, 5.2 a 5.3 učení zvyšuje úspěšnost adaptivní inteligence proti všem třem statickým inteligencím. Nejméně výrazné je zlepšení proti Staviteli, nejvíce proti Obránci. Proti všem třem statickým inteligencím vyhrává adaptivní inteligence průměrně více než polo-



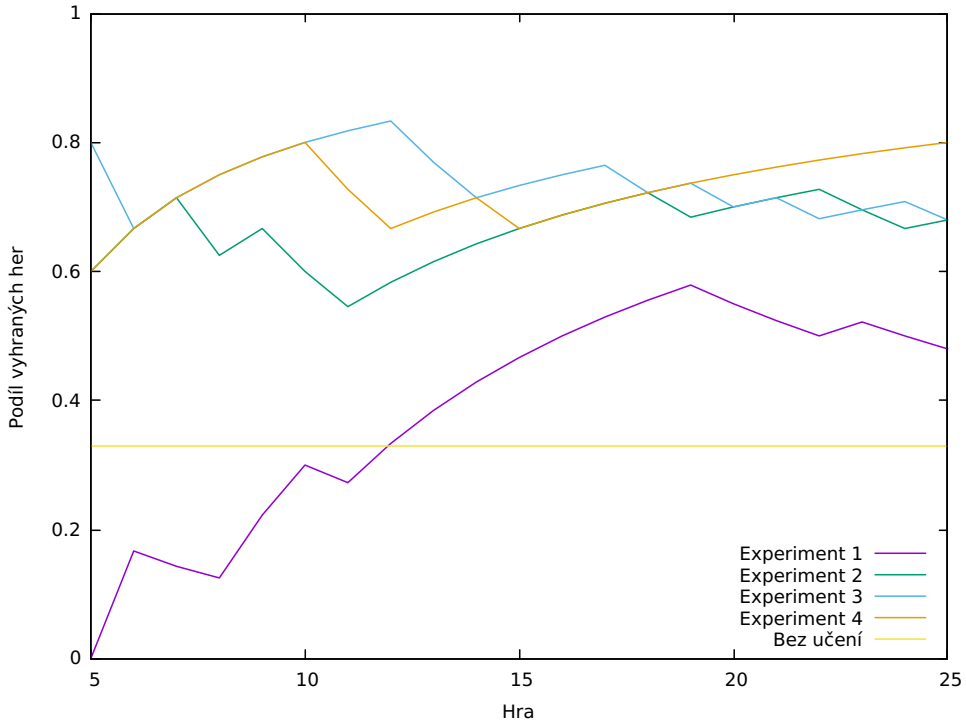
Obrázek 5.1: Podíl her vyhraných dynamickým skriptováním proti Útočníkovi

Tabulka 5.2: Podíl her vyhraných dynamickým skriptováním proti Obránci (čtyři experimenty pro obě varianty)

	Minimum	Průměr	Maximum
Bez učení	0,24	0,33	0,44
S učením	0,48	0,66	0,80

Tabulka 5.3: Podíl her vyhraných dynamickým skriptováním proti Staviteli (čtyři experimenty pro obě varianty)

	Minimum	Průměr	Maximum
Bez učení	0,24	0,44	0,60
S učením	0,44	0,57	0,76



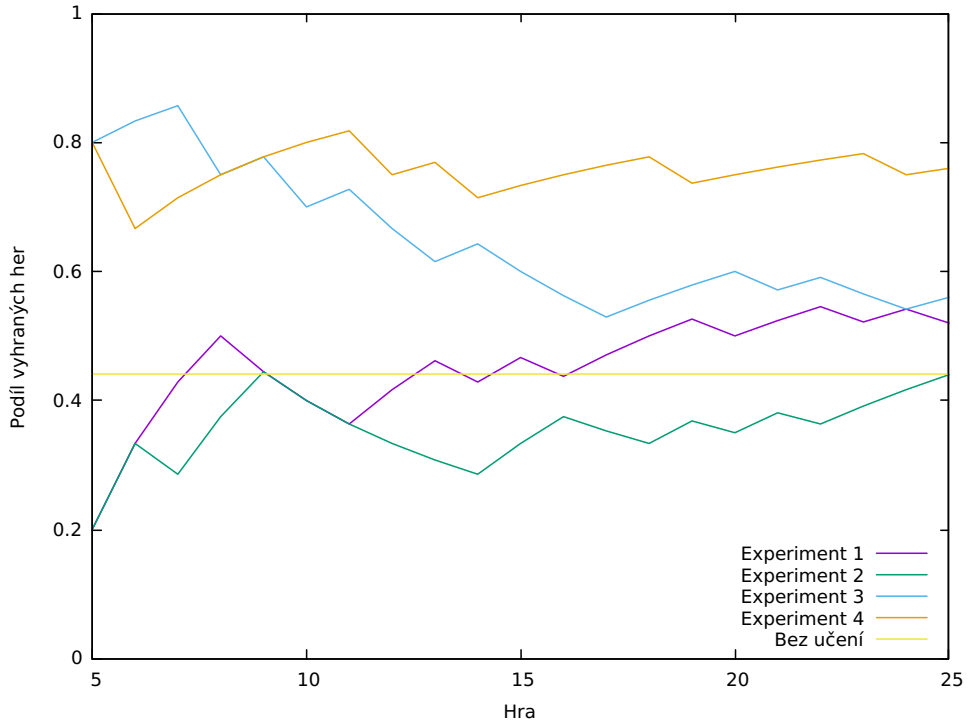
Obrázek 5.2: Podíl her vyhraných dynamickým skriptováním proti Obránci

vinu hraných her. Upravené dynamické skriptování je tedy schopné generovat adaptivní umělou inteligenci kvalitnější, než jsou všechny tři navržené statické inteligence.

Poměrně vysoké podíly her vyhraných neučícím se dynamickým skriptováním ukazují, že inteligence jím vytvářená je schopna kvalitní hry, aniž by proběhlo jakékoli učení.

Postup přizpůsobení proti Útočníkovi ilustruje obrázek 5.1. Dva z jeho experimentů stabilně výrazně překonávají úspěšnost neučícího se dynamického skriptování již po prvních deseti hrách. Přizpůsobení v experimentu 2 je pomalejší, připojuje se k nim ale již kolem patnácté hry. Nejméně úspěšný byl experiment 4, ve kterém se adaptivní inteligence během několika počátečních her naučila velmi neefektivní kombinaci ekonomických vah, z čehož se následně zotavovala. Po jednadvacáté hře již adaptivní inteligence ve všech čtyřech experimentech stabilně vítězí, dochází tedy k efektivní strategii.

Přizpůsobení proti Obránci je podle obrázku 5.2 nejrychlejší. Ve třech experimentech proti němu adaptivní inteligence výrazně překonává neučící se dynamické skriptování v průběhu celého grafem znázorněného období a podíl vyhraných her v nich (kromě období několika her během experimentu 2) vždy překračuje 0,6. Pouze během experimentu 1 trvalo přizpůsobení delší



Obrázek 5.3: Podíl her vyhraných dynamickým skriptováním proti Staviteli

dobu, protože se adaptivní inteligenci nedařilo během prvních her vytvořit dostatečně kvalitní ekonomiku.

Přizpůsobení proti Staviteli je naopak nejpomalejší, jak je vidět z obrázku 5.3. Experiment 2 ani nepřekračuje úspěšnost neučícího se dynamického skriptování a během experimentu 3 je dokonce možné pozorovat postupný úpadek, který se dynamickému skriptování podařilo zvrátit až po deseti hrách. Ve třech ze čtyř experimentů ale přesto adaptivní inteligence vyhrává více než polovinu her, dokáže tudíž přehrávat i na výstavbu zaměřeného Stavitele.

5.3 Vyhodnocení průběhu her a vah pravidel

Předchozí sekce ukazuje, že navržený algoritmus dokázal zlepšit úspěšnost adaptivní inteligence proti všem třem statickým inteligencím. Neříká však nic o tom, jakým způsobem toho dosáhl.

Tím se zabývá tato sekce, která nejprve popisuje závěry z pozorování provedených experimentů a následně zkoumá váhy pravidel, ke kterým dynamické skriptování dochází.

Tabulka 5.4: Tabulka počtu zaseknutí rozvoje v počáteční fázi hry ve sto hrách proti jednotlivým statickým inteligencím pro učící i neučící se dynamické skriptování

	Neučící se	Učící se
Útočník	21	12
Obránce	14	6
Stavitel	16	13

```
(defrule
    (resource-rule-fails rwood)
    (not-built-anything-in 5)
    (try-refunding-wood-consuming-building)
=>
    {0}
)
```

Obrázek 5.4: Implementované pravidlo řešící jeden typ ekonomického zaseknutí

5.3.1 Průběh her

Podle pozorování průběhu her dokáže učení výrazně zrychlit ekonomický růst adaptivní inteligence, což tvoří hlavní část zlepšení její úspěšnosti. Ve vojenské oblasti učení výrazně zvyšuje její agresivitu proti všem třem statickým inteligencím. Kromě velmi agresivních strategií ale adaptivní inteligence často používá i strategii zaměřenou čistě na stavbu budov, při které rekrutuje pouze minimum jednotek a proti nájezdům se chrání budováním obranných věží.

Pozorováním byla zjištěna zásadní nevýhoda zvolené nízké komplexity pravidel v prostředí strategické hry s velmi složitou ekonomikou – algoritmus umožňuje generování skriptů, které mohou, zvláště v počátečních fázích hry, způsobit úplné zaseknutí rozvoje. To je stav, kdy adaptivní inteligence není schopna nic dělat, aniž by bourala již postavené budovy. Jeho příčinami jsou nevyrovnaná produkce a konzumace různých surovin a přílišné utrácení surovin na vojenské účely ve chvíli, kdy je nutné jejich využití pro posílení produkce.

Z tabulky 5.4 je možné vyčíst, že učící se algoritmus dokáže tento problém zmírnit, nedokáže ho však zcela eliminovat. Možným řešením, vyjímaje změny v ekonomickém systému hry a zvýšení komplexity pravidel, je například implementace pravidel umožňujících zotavení se z těchto stavů. Pro ověření této možnosti se jedno takové pravidlo již nacházelo v souboru pravidel během experimentů, viz obrázek 5.4. Pozorováním bylo zjištěno, že opravdu dokáže část zaseknutí vyřešit.

Tabulka 5.5: Tabulka zaokrouhleného průměrného rozdílu vah ve všech dvojicích experimentů, rozdělených podle inteligence proti které byly váhy učeny. První hodnota bere v úvahu váhy všech pravidel, druhá pouze vojenských.

	Útočník	Obránce	Stavitel
Útočník	363/170	376/182	416/195
Obránce	376/182	334/155	336/146
Stavitel	416/195	336/146	334/141

5.3.2 Průzkum souborů vah

Pro průzkum byly prioritně voleny soubory vah z konce experimentů. Pokud však byla úspěšnost před koncem experimentu nízká, byl zvolen soubor vah ze hry co nejblíže konci, okolo které byla adaptivní inteligence úspěšná. Tím byly získány čtyři soubory vah proti každé ze tří statických inteligencí, celkem je tedy zkoumáno dvanáct souborů vah.

Nejčastěji volená vojenská pravidla jsou společná přizpůsobení všem statickým inteligencím:

1. Pravidlo nařizující útoky bez ohledu na počet dostupných útočných jednotek.

2. Pravidla rekrutující útočné jednotky.

3. Pravidlo nařizující budování obranných věží.

Kromě těchto nejčastěji volených pravidel se preferovaná vojenská pravidla výrazně liší experiment od experimentu, i mezi experimenty proti stejné statické inteligenci, a proto z nich není možné vyvozovat další závěry. Vybočuje pouze přizpůsobení proti Útočníkovi. Proti němu adaptivní inteligence volí pravidla budující více ubytování pro vojenské jednotky, což mu umožňuje stavět větší armády, a často rekrutuje obléhací jednotky. Nakonec je potřeba poznamenat, že třebaže mají pravidla rekrutující útočné jednotky vysoké váhy nejčastěji, nejde o jejich naprosté vítězství. V souborech vah mají často podobné nebo i vyšší váhy jiná pravidla zabývající se rekrutací.

Z ekonomických pravidel obvykle volí adaptivní inteligence ta, která jí umožňují budovat více produkčních budov. Nejdůležitější z nich jsou pravidla ovlivňující rychlost růstu populace, kterou se adaptivní inteligence snaží maximalizovat proti Staviteli, proti Obránci a v menší míře i proti Útočníkovi.

Závěry získané zkoumáním souborů vah tedy podporují zjištění z pozorování průběhu her. Adaptivní inteligence opravdu volí pravidla pro rychlý růst ekonomiky a z vojenského hlediska preferuje agresivitu a stavbu věží proti všem třem statickým inteligencím. Jedinou výraznější odlišností je časté použití obléhacích jednotek a stavba více ubytování pro vojenské jednotky proti

```
{u'u_catapult': 0, u'u_spearman': 267, u'u_knight': 23, u'u_swordsman': 557,
 ↪  u'u_axeman': 63, u'u_siegeram': 0, u'u_teutonicknight': 56}
{u'u_catapult': 0, u'u_spearman': 0, u'u_knight': 20, u'u_swordsman': 0,
 ↪  u'u_axeman': 87, u'u_siegeram': 0, u'u_teutonicknight': 0}
Attacker losses: 1236.0
Defender losses: 3024.0
```

Obrázek 5.5: Ukázka nevyrovnanosti poměru sil a utrpěných ztrát výpisem herního serveru o bitvě. Slabý útočník (ve výpisu druhý) utrpěl podstatně menší surovinové ztráty než silný obránce.

```
{u'u_catapult': 0, u'u_spearman': 0, u'u_knight': 0, u'u_swordsman': 0,
 ↪  u'u_axeman': 0, u'u_siegeram': 0, u'u_teutonicknight': 0}
{u'u_catapult': 34, u'u_spearman': 0, u'u_knight': 69, u'u_swordsman': 0,
 ↪  u'u_axeman': 765, u'u_siegeram': 27, u'u_teutonicknight': 0}
Attacker losses: -34.0
Defender losses: 194.0
```

Obrázek 5.6: Ukázka neefektivity agrese výpisem herního serveru o bitvě. Velmi silný útočník s množstvím obléhacích jednotek (ve výpisu druhý) způsobil obránci velmi nízké škody.

Útočníkovi. Základní strategie je tedy prakticky stejná proti všem třem statickým inteligencím.

Tabulka 5.5, ilustrující rozdíly mezi soubory vah experimentů, dále podporuje zjištění, že nedochází k výraznému přizpůsobení specifickému strategii protivníka. Rozdíly mezi soubory vah naučenými proti stejné statické inteligenci totiž nejsou výrazně nižší, než rozdíly mezi soubory naučenými proti inteligencím rozdílným.

5.4 Diskuze

Výsledky provedených experimentů ukazují, že navržený algoritmus dokáže zlepšovat úspěšnost adaptivní inteligence v zápasech proti inteligencím statickým, která je tak schopná proti těmto inteligencím častěji vítězit, než prohrávat. Pozorování průběhu her a průzkum naučených souborů vah však vedou k závěru, že většina tohoto zlepšení spočívá v urychlení ekonomického růstu. Z hlediska vojenské strategie učení vede nezávisle na protivníkovi k preferenci útočných jednotek, útočení nezávislému na množství dostupných útočných jednotek a časté stavbě věží. Nedochází navíc k výraznému přizpůsobení konkrétní strategii protivníka. Pozorované chování může mít množství různých příčin, od návrhu algoritmu, přes postup experimentů a pravidel v nich pou-

```
{u'u_catapult': 0, u'u_spearman': 0, u'u_knight': 0, u'u_swordsman': 0,
 ↪  u'u_axeman': 0, u'u_siegeram': 0, u'u_teutonicknight': 0}
{u'u_catapult': 0, u'u_spearman': 0, u'u_knight': 31, u'u_swordsman': 0,
 ↪  u'u_axeman': 57, u'u_siegeram': 16, u'u_teutonicknight': 0}
Attacker losses: 1243.0
Defender losses: 113.0
```

Obrázek 5.7: Ukázka neefektivity agrese výpisem herního serveru o bitvě. Silný útočník (ve výpisu druhý) utrpěl velmi vysoké ztráty při útoku na ostrov bráněný pouze malým množstvím věží.

žitých, až po herní mechaniky. Popisem těchto příčin se zabývají následující odstavce.

Navržený algoritmus hodnotí pouze poměrně krátkodobé dopady zvolených pravidel. Je proto možné, že volí pouze krátkodobě výhodná pravidla a ta jsou proti všem statickým inteligencím obdobná a preferují ekonomický růst. Pro hodnocení dlouhodobých dopadů by algoritmus bylo možné upravit pomocí propagace změn vah z následujících úseků učení do úseků předchozích. Tato varianta algoritmu ale může nespravedlivě trestat/odměňovat použitá pravidla na základě efektivity skriptů zvolených v následujících úsecích. Oba přístupy tak mají své výhody i nevýhody.

Druhou možnou příčinou je způsob výpočtu ohodnocovací funkce. Ta bere podobným dílem v úvahu všechny aspekty hry. Její změna tak, aby preferovala vojenství před ostatními složkami, by mohla podpořit přizpůsobení adaptivní inteligence ve vojenské rovině.

Třetí z možných příčin se týká souboru pravidel dynamického skriptování a navržených statických inteligencí. Třebaže byl soubor pravidel vytvořen z pravidel převzatých ze statických inteligencí a z pravidel obdobné kvality, dovoluje podle pozorovaného průběhu experimentů rychlejší ekonomický růst, než má kterýkoli z protivníků. Je proto možné, že adaptivní inteligence dochází k rychlému růstu ekonomiky, protože jde o nejsnadnější cestu k jejímu vítězství. Jakmile pak v průběhu hry získá výraznou výhodu v ekonomické produkci, vyplatí se jí více útočit, než například hromadit obranné jednotky na svém ostrově, protože i pokud útok způsobí nižší ztráty než sám utrpí, adaptivní inteligence dokáže tyto ztráty nahradit snadněji, než její protivník.

Další ze zvažovaných příčin jsou herní mechaniky, směřující adaptivního hráče k pozorovanému chování. Při zhodnocení této možnosti je nutné brát v úvahu několik zásadních informací o Taskinu zjištěných v průběhu experimentů:

Disproporce ztrát a poměru sil

Kvůli implementaci bitev multiagentním systémem, kde se jednotky rozdělené do čet postupně pohybují po ostrově a zápasí s četami protivníka

v soubojích mezi dvojicemi čet, jsou výsledné ztráty v bitvách obvykle disproporcionální poměru sil. Podle pozorovaných výsledků bitev jsou tím zvýhodněny hlavně útočící armády, obzvláště v případě, kdy se na ostrově nachází velké množství útočných jednotek obránce. Ty se zřejmě boji nevyhnou, ani když je přítomen dostatek obranných jednotek. Jako příklad lze uvést útok slabé útočné armády na dobře bráněný ostrov, ilustrovaný výpisem bitvy v obrázku 5.5. Logika a zkušenost s jinými strategickými hrami velí předpokládat, že výsledné ztráty budou hovořit výrazně ve prospěch obránce. Jak je však z výpisu patrné, obránce utrpěl o mnoho vyšší surovinové ztráty než útočník.

Neefektivita agrese

Předchozí bod by mohl zanechat dojem, že nejefektivnější možnou strategií je strategie čistě agresivní, kdy hráč nepříteli působí disproporcionální vojenské ztráty a ještě může vykrádat jeho skladiště a ničit jeho budovy pomocí obléhacích jednotek. Ani to však není pravda. Jakkoli je výhodné útočit na nepřátelské armády, útočit na nepřátelský ostrov se slabou nebo neexistující posádkou je naopak velmi nevýhodné.

Rabováním může útočník ukrást pouze polovinu uskladněných surovin a nosnost jednotek je poměrně nízká, není jím proto možné protivníka zlikvidovat. Přesto by však samozřejmě bylo výhodné, nebýt velmi vysoké efektivity věží, které ho dokáží zcela vyřadit ze hry. Efektivitu věží ilustruje výpis bitvy v obrázku 5.7, v níž útočící armáda obsahující obléhací jednotky utrpí více než padesátiprocentní ztráty při útoku na ostrov bráněný výhradně malým množstvím věží. Průzkum fungování bojového systému provedený po tomto zjištění ukázal, že důvodem je vysoká útočná síla věží a neschopnost útočících jednotek zhodnotit rizika pohybu a stání v jejich dostřelu. Vliv věží dále zvyšuje způsob rozmístění budov stavěných umělou inteligencí převzatý z předchozí práce, který preferuje výstavbu v jejich dostřelu.

Posledním důležitým zjištěním ohledně agrese je velmi nízká efektivita obléhacích jednotek. Ilustruje ji výpis bitvy v obrázku 5.6, kde obléhací jednotky způsobily surovinové škody odpovídající méně než $\frac{1}{28}$ jejich pořizovací ceny. Nízká efektivita je způsobena velmi nízkým poškozením, které budovám jejich útoky způsobují. Obléhací jednotky jsou navíc velmi často ničeny i v případě, kdy se na ostrově nachází pouze malé množství obranných jednotek, které na ně útočí prioritně. Snadno likvidovány jsou i věžemi. S ohledem na jejich vysokou pořizovací cenu jsou tak obléhací jednotky prakticky nepoužitelné. Výjimkou je jejich pozorované využití proti Útočníkovi, který však nestaví věže a rekrutuje pouze minimální množství obranných jednotek. To jim umožňuje nerušeně působit škody na budovách, jakkoli malé, a jejich pořizovací cena sama o sobě zvyšuje skóre adaptivního hráče.

Tempo ekonomického rozvoje

Jak bylo vysvětleno v subsekci 2.2.2, tempo ekonomického rozvoje v Taskinu je exponenciální se základem 1,05 (za předpokladu nejvyšší úrovně štěstí, které však není obtížné dosáhnout). To, v kombinaci s výše popsanou vysokou efektivitou věží, nevýhodností rabování a neefektivitou obléhacích jednotek, vede k zásadnímu vlivu ekonomických schopností hráče a velmi nízkému vlivu jeho vojenské strategie na výsledek hry.

Pomocí zjištěných informací je možné pozorované chování vysvětlit, včetně jeho drobných odlišností proti Útočníkovi. Disproporce ztrát a poměru sil vysvětluje, proč adaptivní inteligence považuje agresivní chování za výhodnější, než chování defenzivní, i proč často útočí bez ohledu na množství dostupných útočných jednotek. Neefektivita agrese zdůvodňuje, proč jde pouze o preferenci a ne o naprosté zaměření na ní. Zároveň vysvětluje vysoké váhy stavby věží. Zaměření adaptivní inteligence na rychlost ekonomického rozvoje odpovídá Tempu ekonomického rozvoje. Odlišnosti proti Útočníkovi jsou časté použití obléhacích jednotek, budování většího množství ubytování pro vojenské jednotky a o něco menší zaměření na rychlost ekonomického rozvoje. Obléhací jednotky jsou přímo vysvětleny v Neefektivitě agrese. Budování většího množství ubytování pro vojenské jednotky je možné vysvětlit tím, že Útočník nestaví věže, a tudíž se proti němu vyplatí útoky silnějšími armádami. Menší zaměření na ekonomický rozvoj pak znovu vysvětluje absence věží, která umožňuje porážet Útočníka vojensky.

Je tedy možné, že adaptivní inteligence opravdu dochází k nejefektivnější strategii vzhledem ke strategiím protivníků, která je ale prakticky stejná i proti strategiím tak odlišným, jako jsou Útočník a Obránce.

Zásadní důležitost rychlosti ekonomického rozvoje zároveň objasňuje, proč je přizpůsobení proti Staviteli nejpomalejší a nejméně výrazné – Útočník i Obránce obsahují pravidla pro pomalejší růst populace, aby bylo více surovin dostupných pro vojenské jednotky. Stavitel naopak využívá pravidla zajišťující nejrychlejší růst populace.

Možným vysvětlením části pozorovaného chování je také zjištěný nedostatek navrženého algoritmu. Algoritmus hodnotí najednou efekt pravidel, která se starají o různé části herní inteligence. Pro přehlednost vysvětlení je možné rozdělit pravidla na vojenství a ekonomiku. Pokud vliv jedné z těchto oblastí výrazně převažuje nad druhou, mohou být váhy méně vlivné oblasti upravovány na základě úspěšnosti vlivnější oblasti. V provedených experimentech byl zjištěn zásadní vliv rychlosti ekonomického rozvoje na úspěšnost adaptivního hráče. Jakmile začne adaptivní hráč díky urychlení ekonomického růstu v úsecích učení častěji vítězit než prohrávat, je možné, že algoritmus volí vojenská pravidla, která jsou použita v největším množství úseků. To je dalším z možných vysvětlení volby útoků bez ohledu na množství dostupných útočných jednotek, které jsou logicky prováděny nejčastěji. Zároveň tento nedostatek umožňuje vysvětlit nejméně častou, ale přesto výraznou oblibu budování

obranných věží proti Obránci, který adaptivní inteligenci napadá pouze výjimečně.

Zjištěné informace o herních mechanikách jsou důležité i proto, že dále omezují efektivní strategie v Taskinu na zásadní důležitost ekonomického rozvoje a několik obecně platných vojenských zásad. Pro řešení této situace a zvýšení hratelnosti Taskina je na základě v této práci zjištěných informací možné doporučit následující vylepšení:

1. Výrazně snížit účinnost věží. Pro to je potřeba snížit jejich útočnou sílu a odolnost. Také je nutné upravit chování útočících jednotek, aby byly lépe schopny vyhodnotit riziko, které pohyb a stání v dostřelu věží představuje.

2. Zvýšit účinnost obléhacích jednotek tak, aby se jejich rekrutace vyplatila. Pro to je potřeba zvýšit jejich útočnou sílu a snížit jejich zranitelnost zvýšením jejich odolnosti a zajištěním jejich eskortování útočícími jednotkami, aby nebyly snadno ničeny obránci.

3. Zpomalit růst populace, snížit množství surovin produkovaných produkčními budovami nebo zvýšit ceny budov.

4. Upravit bojový systém tak, aby ztráty více odpovídaly poměru sil. To by bylo možné udělat například povolením spolupráce více čet při zápasu proti protivníkově četě.

5. Zajistit, aby se útočné jednotky držely během obranných bitev zpátky, pokud není jejich zapojení výhodné.

6. Zvýšit hodnotu útočných jednotek v obraně. V současném nastavení jsou extrémně snadno pozabíjeny i řádově slabším protiútokem.

7. Umožnit větší kontrolu hráče nad bojovým systémem, jehož potenciál pro zvýšení hratelnosti není v současné době využit. Zajímavá by byla například možnost volby z více možných taktik a pozorování záznamů vybojovaných bitev. Obdobně by bylo vhodné umožnit hráči řídit rozmístění bránících se jednotek na svém ostrově.

8. Rozmístit ostrovy do mapy tak, aby přesun mezi nimi nebyl okamžitý a trval různé doby, což by zvýšilo možnosti manévrování s jednotkami. Umělé inteligence by také byly schopny reagovat na přicházející útoky protivníků.

9. Přidat možnost výměny surovin za suroviny jiného typu. Jakkoli by taková výměna musela být nevýhodná, umožnila by snadné řešení surovinových zaseknutí.

10. Použít diamanty i k něčemu jinému, než k výzkumu. V případě použití pouze k výzkumu jejich motivační hodnota velmi rychle klesá.

Závěr

První kapitola práce specifikovala cíl práce. Tím bylo prozkoumat možnosti využití metod adaptivní umělé inteligence pro přizpůsobení strategie umělé inteligence strategiím proti ní používaným v motivační hře v aplikaci Taskino. Cílem teoretické části práce bylo seznámit se se základy herní umělé inteligence, s aplikací Taskino a s existujícími metodami adaptivní herní umělé inteligence potenciálně využitelnými v aplikaci Taskino. Cílem praktické části bylo vybrat jednu z těchto metod nebo navrhnout metodu vlastní, implementovat ji v aplikaci Taskino a vyhodnotit její kvalitu na základě porovnání jí vytvořené umělé inteligence a počítačových protivníků s neměnným chováním.

Druhá a třetí kapitola splnily vytyčený cíl teoretické části práce. Druhá kapitola popsala základy herní umělé inteligence a aplikaci Taskino, se zvláštním zaměřením na informace důležité při návrhu metody adaptivní herní umělé inteligence. Třetí kapitola popsala vybrané metody adaptivní herní umělé inteligence, zvažované pro použití v Taskinu.

Pro splnění cíle praktické části práce bylo ve čtvrté kapitole z uvažovaných metod vybráno dynamické skriptování. Následující část kapitoly popsala úpravy, opravy a vylepšení hry provedené pro zvýšení její hratelnosti a umožnění úspěšně implementace a ověření funkčnosti dynamického skriptování. Zbytek kapitoly popsal návrh upraveného dynamického skriptování a jeho implementaci použitou během experimentů. Závěrečná kapitola experimentálně ověřila jeho funkčnost a podrobně vyhodnotila kvalitu jím vytvořené umělé inteligence pomocí zápasů se třemi neměnnými inteligencemi, reprezentujícími ofenzivní, defenzivní a budovatelskou herní strategii. Nakonec diskutovala zjištěné výsledky a na základě pozorování průběhu zápasů doporučila změny v Taskinu určené ke zvýšení jeho hratelnosti.

Cíl práce i jeho dílčí části tedy byly prací plně splněny.

Výsledky experimentů ukázaly, že upravené dynamické skriptování dokáže zvýšit úspěšnost adaptivní umělé inteligence, která tak dokáže porážet všechny tři počítačové protivníky s neměnným chováním. Dosahuje toho ale hlavně zrychlením ekonomického růstu. Vojenské přizpůsobení spočívá hlavně v časté

volbě rekrutace útočných jednotek, útočení nezávislého na počtu dostupných jednotek a budování obranných věží. Přizpůsobení specifické strategii protivníka není výrazné a volená strategie je tak proti všem třem neměnným inteligencím téměř stejná. V diskuzi byly popsány možné příčiny tohoto chování. Zjištěno mimo jiné bylo, že herní mechaniky zkoumané hry vedou místo rozmanitosti strategií k zásadnímu vlivu ekonomických schopností hráče na výsledek hry a několika obecně platným vojenským zásadám. Úpravy hry navrhované v diskuzi jsou určeny mimo jiné k řešení tohoto problému.

Do budoucna by bylo vhodné v Taskinu implementovat navrhované změny a znovu vyhodnotit úspěšnost navrženého algoritmu v upraveném herním prostředí, které by už obsahovalo více vzájemně protikladných strategií. Upravené dynamické skriptování je možné dále vylepšovat a experimentovat s jeho různými variantami, například v diskuzi zmiňovanou propagací pozdějších výsledků do hodnocení dřívějších akcí, nebo v popisu implementace zmiňovanou změnou rychlosti učení v závislosti na současné úspěšnosti adaptivního hráče.

Literatura

[1] Streichert, F.: Introduction to Evolutionary Algorithms. *Frankfurt Math Finance Workshop*, ročník 4, 2002.

[2] Browne, C. et al.: A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games*, ročník 4, č. 1, březen 2012: s. 1–43, ISSN 1943-068X, doi:10.1109/TCIAIG.2012.2186810.

[3] Millington, I.; Funge, J.: *Artificial intelligence for games*. Burlington: Morgan Kaufmann, druhé vydání, 2009, ISBN 978-0-12-374731-0.

[4] Sycara, K.: Multiagent systems. *AI magazine*, ročník 19, č. 2, 1998: str. 79, ISSN 0738-4602.

[5] Kehoe, D.: Designing Artificial Intelligence for Games (part 3). [online], leden 2015, [cit. 2017-04-14]. Dostupné z: `https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-3`

[6] Graft, K.: When artificial intelligence in video games becomes...artificially intelligent. [online], září 2015, [cit. 2017-04-22]. Dostupné z: `http://www.gamasutra.com/view/news/253974/When_artificial_intelligence_in_video_games_becomesartificially_intelligent.php`

[7] Kvasnička, M.: *Motivační webová aplikace s využitím gamifikačních prvků*. Diplomová práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2015.

[8] Kulík, J.: *Herní prostředí s umělou inteligencí pro motivační webovou aplikaci*. Bakalářská práce, České vysoké učení technické v Praze, Fakulta informačních technologií, Praha, 2016.

[9] Sutton, R.; Barto, A.: Reinforcement Learning: An Introduction, Second edition draft. [online], září 2016, [cit. 2017-04-21]. Dostupné z: http://incompleteideas.net/sutton/book/bookdraft2016sep.pdf

[10] Wender, S.; Watson, I.: Applying reinforcement learning to small scale combat in the real-time strategy game StarCraft:Broodwar. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, září 2012, ISSN 2325-4270, s. 402–408, doi:10.1109/CIG.2012.6374183, [cit. 2017-04-14].

[11] Madeira, C.; Corruble, V.: Combining Reinforcement Learning with a Multi-level Abstraction Method to Design a Powerful Game AI. In *2011 Brazilian Symposium on Games and Digital Entertainment*, listopad 2011, ISSN 2159-6654, s. 132–140, doi:10.1109/SBGAMES.2011.21, [cit. 2017-04-14].

[12] Amato, C.; Shani, G.: High-level reinforcement learning in strategy games. In *AAMAS '10 Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, květen 2010, ISBN 978-0-9826571-1-9, s. 75–82, [cit. 2017-04-14].

[13] Spronck, P. et al.: Adaptive Game AI with Dynamic Scripting. *Machine learning [online]*, březen 2006, [cit. 2017-03-25]. Dostupné z: http://link.springer.com/article/10.1007%2Fs10994-006-6205-6

[14] Lindh, D.: *Adaptive combat AI in strategy games - An approach based on Dynamic Scripting*. Diplomová práce, Lund University, Faculty of Science, Department of Computer Science, Lund, 2008. Dostupné z: http://fileadmin.cs.lth.se/ai/xj/DavidLindh/report.pdf

[15] Ponsen, M.: *Improving adaptive game AI with evolutionary learning*. Diplomová práce, Delft University of Technology, Faculty of Media & Knowledge Engineering, Delft, 2004. Dostupné z: https://pdfs.semanticscholar.org/9882/a2807aec55830fcfae4a64621f3c685e987f.pdf

[16] Risi, S.; Togelius, J.: Neuroevolution in Games: State of the Art and Open Challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, ročník 9, č. 1, březen 2017: s. 25–41, ISSN 1943-068X.

[17] Traish, J.; Tulip, J.: Towards adaptive online RTS AI with NEAT. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, září 2012, ISSN 2325-4270, s. 430–437, doi:10.1109/CIG.2012.6374187.

[18] Martin, M.: Using a Genetic Algorithm to Create Adaptive Enemy AI. [online], srpen 2011, [cit. 2017-04-16]. Dostupné z: http://www.gamasutra.com/blogs/MichaelMartin/20110830/90109/Using_a_Genetic_Algorithm_to_Create_Adaptive_Enemy_AI.php

[19] Champandard, A.: Monte-Carlo Tree Search in TOTAL WAR: ROME II's Campaign AI. [online], srpen 2014, [cit. 2017-04-17]. Dostupné z: http://aigamedev.com/open/coverage/mcts-rome-ii/

[20] Python Software Foundation: What is Python? Executive Summary. [online], [cit. 2017-04-21]. Dostupné z: https://www.python.org/doc/essays/blurb/

[21] Tagliaferri, L.: Python 2 vs Python 3: Practical Considerations. [online], srpen 2016, [cit. 2017-04-21]. Dostupné z: https://www.digitalocean.com/community/tutorials/python-2-vs-python-3-practical-considerations-2

[22] MongoDB, Inc: Introduction to MongoDB. [online], [cit. 2017-04-21]. Dostupné z: https://docs.mongodb.com/manual/introduction/

Příloha **A**

Seznam použitých zkratek

CRPG Computer role-playing game

RTS Real-time strategy

FPS First-person shooter

MCTS Monte-Carlo Tree Search

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... experimentální skripty a upravené zdrojové kódy Taskina
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Sochor_Matej_2017.pdf` ..... text práce ve formátu PDF

Příloha C

Instalační příručka

Tato příručka byla převzata z [7]. Změny provedené v této práci na postupu instalace nic nemění. Jediný rozdíl je v adresářové struktuře na přiloženém médiu – pro v příručce popsané příkazy je nejprve nutné vstoupit do složky taskino.

C.1 Závislosti

- Vývojové prostředí
  - Python 2.6 nebo 2.7
  - Python virtualenv
  - MongoDB
  - Java 7 nebo vyšší
  - NodeJS
- Produkční prostředí
  - Python 2.6 nebo 2.7
  - Python virtualenv
  - MongoDB

C.2 Průběh instalace v linuxovém prostředí

Vývojové prostředí:

```
$ [sudo] apt-get install python-dev nodejs-legacy npm
$ [sudo] npm install -g less bower grunt-cli

$ bower install
$ npm install
```

```
$ ./install.py
```

Produkční prostředí:

```
$ [sudo] apt-get install python-dev

$ ./install.py
```

C.3 Spuštění aplikace

Hlavní konfigurační soubor se nachází v souboru taskino/taskino_server/config.py, pomocí tohoto souboru lze nastavit IP adresu a port serveru, na kterém se má aplikace spustit, přístupové údaje k databázi, přihlašovací údaje k SMTP serveru pro posílání emailů nebo například zda se má spustit produkční nebo vývojové prostředí.

```
$ ./rundb.py
$ ./runserver.py

$ grunt [compile] # spusteni automatizace ukolu
                  # [compile] jednorazova kompilace klientske casti aplikace
```

Příloha D

Experimentální příručka

V rámci práce bylo vytvořeno několik skriptů určených pro pohodlné spouštění experimentů s upraveným dynamickým skriptováním. Skripty se spolu s kódy aplikace nachází na přiloženém médiu. Jejich funkčnost byla testována na operačním systému *Ubuntu*, využití pro jiné operační systémy může vyžadovat jejich úpravu. Před použitím skriptů spouštějících jednotlivé experimenty je nutné spustit aplikační server a (pokud jde o první spuštění serveru) registrovat právě jednoho uživatele v grafickém rozhraní aplikace Taskino.

runtask.bash je určený ke snadnému spuštění aplikačního serveru. Pro jeho správnou funkčnost je nutné, aby se nacházel na stejné pozici vzhledem ke zdrojovým kódům aplikace jako na přiloženém médiu.

runbasiclearning.bash spouští experiment s učícím se dynamickým skriptováním. Má tři parametry – počet her, počet tahů na hru a název souboru do kterého mají být ukládány výsledky experimentu. Příklad použití, který spustí experiment tvořený 25 hrami, každou trvající 150 tahů, a výsledky uloží do souboru *defender_1*:

```
./runbasiclearning.bash 25 150 defender_1
```

runnonlearning.bash spouští experiment s neučícím se dynamickým skriptováním. Má stejné parametry a používá se stejným způsobem jako skript předchozí.

runlearning.bash využívají oba skripty spouštějící experimenty. Ty nejprve provedou nastavení parametrů učení podle svého zaměření a následně spustí tento skript, který zajišťuje samotný běh her. Jde tedy pouze o podpůrný skript a pro spouštění experimentů je doporučeno používat skripty předcházející.

Proti které ze statických inteligencí experiment probíhá je možné změnit v souboru taskino/taskino_server/ai/WorldAIController.py na řádce:

```
# Choose static AI type here
decisionEngine = DecisionEngine(player, island, {'my': stats, 'enemy':
↪  human_stats}, executionEngine, self._score, 'attacker')
```

V posledním parametru je možné zadat jeden ze tří typů – *attacker*, *defender* a *builder*.

Textovým výstupem skriptů spouštějících experimenty jsou v průběhu her informace o tom, v kolikátém tahu se hra právě nachází, na konci každé hry pak základní výpis o jejích výsledcích, naučeném souboru vah a o parametrech použitých během experimentu. Další informace o průběhu jednotlivých her, například výpisy bitev a průběžná skóre hráčů, je možné pozorovat ve výpisu aplikačního serveru.