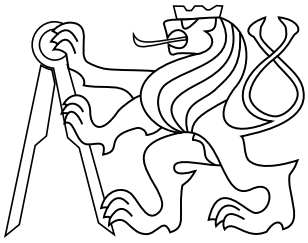
diplomová práce

Dynamické vyvažování obtížnosti her pomocí metod teorie her

Lukáš Beran



Květen 2013

Branislav Bošanský

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra kybernetiky

Poděkování

Chtěl bych poděkovat vedoucímu práce Mgr. Branislavu Bošanskému za cennou pedagogickou a odbornou pomoc při návrhu a řešení zadané problematiky. Dále bych rád poděkoval rodině a přítelkyni za psychickou podporu a motivaci. Děkuji.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....
Podpis autora práce

Abstrakt

Dynamické vyvažování obtížnosti slouží k lepšímu přizpůsobování se programů úrovni uživatelům. V teoretické části shrnuji existující přístupy a jejich aplikaci, definuji zábavnost a navrhuji metriky, jak ji měřit. Dále navrhuji nový přístup založený na hráči prostředí. Tento hráč ovlivňuje náhodu a skrytou informaci ve hře za účelem lepšího požitku z ní. V praktické části popisuji testující prostředí s implementovanými hrami Bludiště, Ludo a Ztracená města, které jsem použil pro evaluaci algoritmů hráče prostředí a pro porovnání s dvěma existujícími algoritmy.

Klíčová slova

Dynamické vyvažování obtížnosti her; adaptivní UI; teorie her;

Abstrakt

Dynamic difficulty adjustement serves for better software adaptivity to user skills. In theoretical part I summed up existing approaches and applications. I define fun and I suggest metrics of fun. I suggest new approach based on environmental player. This player control chance and hidden information in game with purpose of better fun. In practical part I describe testbed with games Maze, Ludo and Lost Cities which I have used for evaluation of environmental player algorithms and for compering them with two existing approaches.

Keywords

Dynamic difficulty game balancing; adaptive AI; game theory;

Obsah

Seznam obrázků

Zkratky

Seznam zkratek a některých pojmů používaných v textu.

DDA	dynamic difficulty adjustement, dynamicky vyvažovaná obtížnost
POMDP	částečně pozorovatelné markovské rozhodovací procesy
NPC	non-player character, postava ovládaná počítačem
FPS	first person shooter, střílečka z pohledu 1. osoby
CBR	case-base reasoning
power-up	bonusový předmět vylepšující hráče

1. Úvod

V současné době v domácnostech stále přibývají inteligentní elektrická zařízení se zabudovaným operačním systémem. Počítače a chytré telefony jsou již téměř samozřejmostí, inteligentní televize a ledničky se pomalu přidávají. Tato zařízení jsou ovládána lidmi s různou dovedností. Koncept dynamicky vyvažované obtížnosti se snaží o přizpůsobování systému uživatelům dle jejich schopností.

Tématem této diplomové práce je dynamicky vyvažovaná obtížnost(DDA) v počítačových hrách. DDA v počítačových hrách má za úkol přizpůsobovat hru hráči, aby z ní měl co nejlepší požitek.

V úvodní kapitole popisuji cíl práce, blíže zadefinuji koncept DDA a jeho dělení do kategorií. Dále ve stručnosti zmíním různé aplikace DDA nejen v komerčním průmyslu, ale také např. v lékařství a ve vzdělávání.

V druhé kapitole se zaměřím na pojem zábava, představím pojem flow a zadefinuji několik možných metrik, pomocí kterých lze zábavu měřit. Ve chvíli, kdy jsme schopni zábavu měřit, můžeme nejen ohodnocovat zábavu po skončení hry, ale můžeme tyto metriky dále použít při dynamické úpravě her.

Třetí kapitola je rešerší existujících DDA přístupů. Mezi existujícími přístupy jsou zástupci známých oblastí umělé inteligence jako jsou neuronové sítě, genetické algoritmy, POMDP, fuzzy logika, teorie her a další.

Na začátku čtvrté kapitoly popíši teoretický základ z teorie her a jejích algoritmů, který poté využiji při návrhu vlastního přístupu DDA nazvaného hráčem prostředí. Představím zde 4 algoritmy založené na novém principu.

Navržené algoritmy je potřeba otestovat. V páté kapitole seznamuji s testovacím prostředím a použitými hrami v něm.

V předposlední šesté kapitole popisuji provedené experimenty a jejich výsledky.

V závěru shrnuji provedenou práci, jestli se podařilo splnit zadání a cíle diplomové práce.

Nakonec přidávám v příloze stručný popis ovládání aplikace a několik dodatečných grafů z experimentů, které se nehodily do šesté kapitoly.

1.1. Cíl diplomové práce

Cílem diplomové práce je prozkoumat možnosti přístupů k dynamické adaptivitě v počítačových hrách, navrhnout přístup nový a ten analyzovat a porovnat s přístupy existujícími.

1.2. Definice

Dynamické vyvažování obtížnosti (dynamic difficulty adjustement, DDA) je obecným konceptem, jak přistupovat k návrhu programů, jež jsou využívány uživateli rozdílných schopností a zkušeností.

Klasickým přístup je předdefinování několika rozdílných nastavení s odlišnými požadavky na dovednosti uživatelů, kteří si mezi těmito nastaveními volí ručně. Naopak dynamický, adaptivní přístup volbu nastavení nechává alespoň z části na samotném programu, který se rozhoduje na základě modelu uživatele. Dále se zaměřím na koncept DDA používaný v počítačových hrách.

Synonymy pro DDA jsou např. dynamic difficulty balancing, dynamic game balancing, auto-dynamic difficulty, adaptive difficulty. Vyvažování obtížnosti je základem velké většiny her využívající DDA. Myšlenka za tím je jednoduchá. Je-li hra pro hráče příliš snadná, hráč se nudí, hru opustí. Naopak je-li hra příliš obtížná, hráč je frustrován, hru opouští. V obou případech hra ztrácí své uživatele.

Hlavní cílem DDA v počítačových hrách je udělat hru více zábavnou pro hráče. Míra zábavnosti hry není závislá pouze na její obtížnosti. Jak lze definovat zábavu a jak ji lze měřit, více popíši v kapitole 2.2.

Každé využití auto-dynamického vyvažování hry lze zařadit do několika kategorií z různých úhlů pohledu. Různé hry vyžadují různé přístupy a je dobré se zamyslet nad konkrétní hrou. Vybrat si z jakých kategorií by mělo být DDA použito. Designér hry by si měl položit několik následujících otázek :

1. Vytvářím vážnou hru, nebo hru jen pro zábavu?
2. Měl by hráč vědět, jestli je DDA použito?
3. Má být obtížnost měněna v průběhu hry, nebo pouze na začátku?
4. Jakým způsobem může být ovlivňována obtížnost hry?

Na základě odpovědí lze hru zařadit do následujících kategorií.

1.2.1. Vážné a nevážné hry

U her ze zábavního průmyslu(entertainment games) je na prvním místě samotná zábava a na tu by se měli vývojáři zaměřit. Především jde o snahu udělat hru dostatečnou výzvou pro hráče. Návrháři vážných her (serious games) mají těžší úkol. Zábava je pouhý prostředek pro splnění jiného cíle. Vezměme v úvahu vzdělávací hry a hry poskytující nějaký druh tréninku. Úkolem těchto her je přenos znalostí mezi hrou a uživatel, a tedy DDA se snaží tento přenos co nejvíce zefektivnit. Důležité je najít vyrovnanou úroveň mezi zábavou a smyslem těchto her, přenosem znalosti. [1]

1.2.2. Explicitní a implicitní

Druhé dělení rozlišuje stav, kdy je hráč obeznámen s dynamickou obtížností a kdy naopak je mu to zatajena. Jestliže hráč dopředu ví, že se obtížnost hry mění dle jeho

konání, jedná se o explicitní DDA. Pokud je snaha utajit dynamickou změnu obtížnosti, jedná se o implicitní DDA.

Explicitní použití by mělo být dobře známé i ze všedního života. Když mezi s sebou v něčem soutěží týmy, kteří nejsou svými schopnostmi vyrovnaní, často se přistoupí k nějakému handicapu pro ty silnější. Ať už se jedná o věnování náskoku při závodu v běhu mladšímu z bratrů, či o posazení zdravého člověka do kolečkového křesla na paralympiádě.

Příkladem ze světa deskových her může být ve hře Go povolení slabšímu hráči na začátku hry zahrát několik svých kamenů navíc, nebo naopak odebrání některých figur ve hře šachy hráči silnějšímu.

Někdy zařazení hry nemusí být zcela jednoznačné. Mnoho hráčů z laické veřejnosti si je vědomo podvádění v závodních hrách jako je Mario Kart[2], či Need For Speed[3], přestože se o tom nedozvědí v pravidlech hry. V případě, že se vývojáři rozhodnou pro implicitní DDA, měli by jej navrhnout tak, aby jej hráč neodhalil. Jestliže o něm každý ví, asi už není zcela korektní hru zařadit do implicitní kategorie.

Nejasná kategorizace může být i v opačném případě. Mějme jako příklad moderní deskovou hru Vysoké napětí[4]. Ve hře existují pravidla závislá na aktuálním pořadí hráčů a snaží se pomáhat prohrávajícím hráčům a naopak ubližovat těm ve vedení. U Vysokého napětí hráči nakupují suroviny v opačném pořadí než si vedou. Stejné suroviny mají rozdílnou cenu. Poslední hráč vykoupí nejlevnější suroviny a naopak na toho prvního zůstanou ty nejdražší. Za stejnou věc zaplatí různě, a tedy se zvýší šance posledního hráče dostat se do vedení. Přestože toto pravidlo je všem zúčastněným známo, tak asi málokdo o něm přemýšlí jako o dynamické vyvažování obtížnosti hry.

1.2.3. Dynamická a statická

Obtížnost hry lze přizpůsobovat před začátkem hry, nebo v jejím průběhu. Dle toho se rozděluje DDA na statickou(offline), či dynamickou(online). Typický příkladem offline adaptivity bude zpracování hráčových dat a úprava hry během jejího načítání. Z tohoto důvodu je offline adaptivita zaměřena především na generování herního světa, herních scénářů a úkolů[1]. Příkladem mohou být hry Fallout 3 a Fallout: New Vegas[5], kde se při vstupu na nové území generují nepřátelé, a to dle levelu hráčova avatara.

Offline adaptivitu lze také využít při vytváření logických hádanek a her. Na základě rychlosti řešení/nedokončení předchozí hádanky, lze vygenerovat hádanku novou lépe odpovídající hráčovým schopnostem. Tímto problémem se zabývá článek Automatic Generation of Game Elements via Evolution[6], kde testovanými hrami byla hra se šachovými figurami a hra procházení barevným bludištěm.

Online adaptivita bude naopak zaměřena na adaptivitu umělé inteligence NPC a úpravu pravidel hry. Vede-li si hráč příliš dobře v závodní hře, ostatním hráčů se zvýší maximální rychlost. Ztratí-li hráč příliš životů, v rohu v další místnosti se objeví lékárnička doplňující zdraví. Adaptivita může být samozřejmě i druhým směrem. Sebere-li hráč soupeři důležité figurky v deskové hře šachy, soupeři se zvýší "inteligence", začne

uvažovat na více kol dopředu.

Online adaptivitu můžeme zasadit přímo do principů hry. Ve hře Pocket Billiards hrají proti sobě dva hráči na kulečníkovém stole. Na stole je několik koulí dvou barev. Každý hráč má přiřazenou jednu z barev a jeho úkolem je dostrkat své koule do děr dříve než to udělá soupeř. Adaptivita je zde dána samotným principem hry. Dostane-li se jeden hráč do vedení a odstraní z plochy znatelně více koulí než jeho soupeř, pak je pro něj stále obtížnější dostat další koule do děr. Zároveň se zvyšuje pravděpodobnost, že bude muset provést takový tah, který může odstranit z plochy soupeřovu koule, a tedy mu tím může pomoci dorovnat skóre.[2]

1.2.4. Adaptivita herních komponent

Každá hra lze rozdělit do několika komponent. Konkrétně se jedná o následující komponenty [1] :

1. Svět a objekty v něm.
2. Herní mechaniky.
3. NPC a jejich umělá inteligence.
4. Příběh.
5. Herní scénáře a úkoly.

Všechny jednotlivé komponenty mohou být adaptivní a přizpůsobovat se konkrétnímu hráči.

Svět a objekty v něm : V [6] vytvářejí adaptivně logické úlohy, Hamlet [7] pro Half-Life rozmisťuje inteligentně náboje a lékárničky po úrovních. V Left 4 Dead adaptivně generují prostředí s různou komplexitou.

Herní mechaniky : Ve hře Max Payne se s výkonem hráče mění zaměřovací asistent a síla protivníků. [8]

NPC a jejich umělou inteligenci : V Pro Evolution Soccer 2008 se soupeř přizpůsobují strategii hráče. [9]

Příběh : Ve Valve považují výběr typu nepřátel dle aktuálního stavu hráče ve hře Left 4 Dead jako formu vyprávění příběhu. [10] Dále se touto problematikou zabývali např. Barber a Kudenko [11].

Herní scénáře a úkoly : Tato oblast patří zatím k těm méně probádaným. Lze si ale představit v dnes populárních MMORPG úpravu úkolů na míru jednotlivých hráčů a dle aktuální situace hry. Pokud např. hráč navštěvuje stále stejné území, může NPC postava zadat jako úkol pobití určitého množství monster nacházející se v dané oblasti a tím pro něj rutinu proměnit v o trochu více zábavnou.

1.3. Aplikace

Algoritmy vyvažování obtížnosti lze využít v širokém spektru aplikací. Mohou být vhodné všude tam, kde je vyžadována určitá dovednost, schopnost. V takovém případě

může být obtížné aplikaci, program navrhnout tak, aby byl dobře využitelný velkým spektrem lidí různých schopností.

DDA můžeme nalézt nejen v zábavním průmyslu, ale i u vážných her. V následujících 4 podkapitolách popisuji konkrétní užití dynamické obtížnosti v komerční i v akademické sféře.

1.3.1. Zábavní průmysl

Hráče počítačových her lze rozdělit dle jejich schopností od příležitostných až po velmi zkušené hráče. Většina her obsahuje statickou volbu obtížnosti na začátku hry. V některých případech to nemusí být dostačující, a proto se tvůrci komerčních her snaží více, či méně úspěšně implementovat adaptivní obtížnost.

Na www stránce [3] lze nalézt desítky příkladů všech různých žánrů. Do následujícího seznamu 5 her jsem vybral ty známější příklady.

Left 4 Dead

V zombie hře Left 4 Dead pojmenovali adaptivní systém The AI Director. Na základě aktuálního hráčova zdraví, munice a relativní pozice v rámci dané úrovně hry The AI Director generuje ve hře zbraně, munici, lékárničky na pomoc hráči a naopak generuje lehčí, či těžší nepřátele. Např. blíží-li se hráč ke konci herní úrovně a má plné zdraví i munici, hra vygeneruje těžkého soupeře „Tank“. [10]

Max Payne 3

Hra Max Payne 3 obsahuje celkem 5 statických obtížností(Lehká, Střední, Těžká, Velmi těžká, Ze staré školy), které se v průběhu hry adaptivně přizpůsobují hráči. Čím nižší obtížnost si hráč na začátku zvolí, tím více se hra může měnit ve prospěch hráče.

Jestliže hráč opakovaně umírá, dostane se mu pomoci ve formě bonusových léků, které umožní lehčí projití daného úseku hry. Při smrti na lehkou a střední obtížnost se hráčův avatar obnoví minimálně s jedním plným zásobníkem pro každou zbraň vyjma granátometů. Plus za každé tři úmrtí ve stejném úseku dostane jeden lék navíc až do maximálního limitu devíti léků. Na těžkou obtížnost je dynamická obtížnost více limitovaná. Jestliže hráč zemře 5 krát po sobě, dostane jeden lék. Pokud zemře podesáté, dostane druhý lék. Další léky mu hra již nepřidává. [12]

The Elder Scrolls IV: Oblivion

Dalším příkladem mohou být hry Oblivion a Fallout 3 od Bethesda Softworks. V Oblivionu nepřátelé levelují s hráčem. Stráže ve městě mají level o 2-5 vyšší než vy, banditi mají level o 2-5 nižší atd. Tímto je docíleno, že se můžete vydat kamkoli ve hře aniž byste narazili na příliš obtížné nepřátele.

Mimo síly nepřátel se adaptivně upravuje druh nepřátel, jejich vybavení, nabízení zboží v obchodech apod. Občas může docházet k nelogickým situacím, kdy obyčejní

potulní bandité mají na sobě nejmodernější brnění, nebo kdy máte za úkol donést vlčí kožešinu ve světe, kde už se tak slabí nepřátelé nepohybují. [13]

Mario Kart Wii

Závodní simulátory jsou dobře známé využíváním adaptivní obtížnosti her a mezi nejznámější zástupce patří arkádové závody Mario Kart. Ve hře se adaptivně mění rychlost protivníků a také bonusové power-upy, které můžete sbírat. Hra podporuje natolik prohrávající hráče, že ať už je aktuální stav hry jakýkoli, může vyhrát kdokoliv.

Což lze brát jako velkou výhodu, kdy žádný z hráčů nemá důvod ke vzdávání hry. Nevýhodou je právě známost a odhalení tohoto systému, a tudíž je lehce zneužitelná. Např. konkrétně ve variantě Mario Kart Wii je vedoucí hráč na začátku posledního kola bombardován modrým krunýřem, či jinou devastující zbraní a je záhy poslán na poslední místo.

Optimální strategie pro hraní hry v tomto případě zahrnuje princip vyvažování. Optimální strategií je projet do posledního kola na druhé pozici. [2]

Pro Evolution Soccer 2008

Úspěšný fotbalový simulátor Pro Evolution Soccer se ve své verzi s číslem 2008 chlubil adaptivním systémem nazvaný Teamvision. Teamvision se učí od hráče jeho styl hry a snaží se upravovat taktiku svého týmu, aby co nejlépe reagovala na tu soupeřovu. Použití jedné strategie může fungovat jednou, dvakrát, ale později již naprosto stejná strategie nevede k vítězství. [9]

1.3.2. Cvičení

Herní zařízení jako jsou Microsoft Kinect[14] a Nintendo Wii[15] dávají prostor pohybovým hrám. Stejně jako v jiných příkladech i zde platí, že existují lidé s diametrálně odlišnou fyzickou kondicí. Kondice se v ideálním případě při opakované hře stále zlepšuje, a proto je vhodné k tomu přizpůsobovat obtížnost hry.

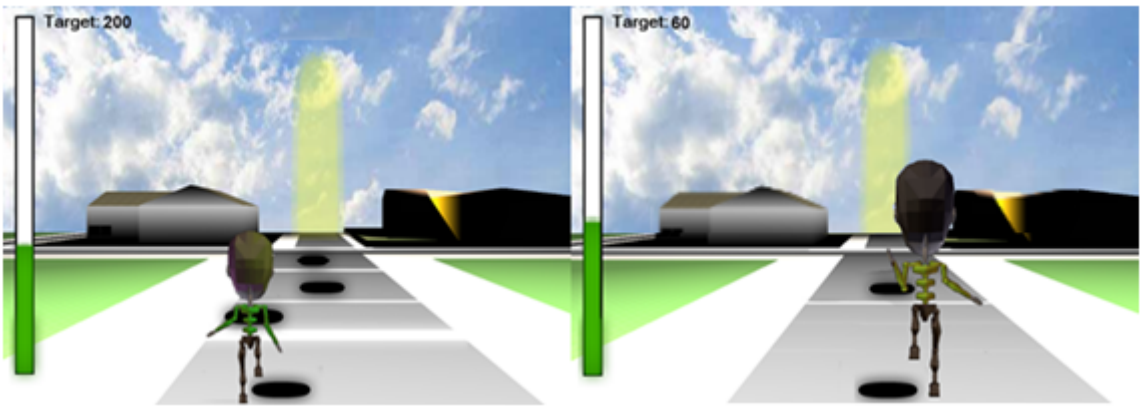
Příkladem takové aplikace může být jednoduchá chodící hra hratelná v internetovém prohlížeči, jež má za úkol motivovat starší lidi k pohybu.

V druhém případě nebylo využito žádné ze zmíněných zařízení. Autoři článku [16] se zaměřili na jogging v páru.

Podpora pohybu starších lidí

Skupina z Technologického institutu se zaměřila na vývoj hry, jež má starší lidi motivovat k pohybu a jejíž nedílnou součástí je vyvažování obtížnosti. [17]

Hra je vytvářena v HTML5 pro běžné použití ve webových prohlížečích a využívá Microsoft Kinect ovládání.



Obrázek 1. Screenshoty HTML5 hry. Vlevo hraje nadaný hráč, vpravo s menší dovedností. [17]

Základním cílem hry je udělat předem dané množství kroků v každé hře. Kroky jsou zaznamenávány pomocí Kinectu. Hráč musí jít v rytmu a zároveň se musí vyhýbat dírám v zemi. V případě špatných, či propásnutých kroků, hráčův avatar zpomalí.

Při hře více hráčů se všichni zúčastnění snaží jít ve stejném tempu. Obtížnost je upravována přidáváním, či odebíráním překážek pro jednotlivé hráče.

Jogging na dálku

Ne každého baví běhat po parku samostatně a zároveň může být těžké najít někoho, kdo by si s vámi zaběhal ve stejnou dobu. Řešením může být jogging na dálku (jogging over distance), kdy dva lidé běží ve stejnou dobu, ale každý běží jinde, třeba i v jiném státě. Oba cvičící se dorozumívají přes telefon se sluchátky na hlavě. Povídají si, navzájem se podporují.

Různí lidé mají různou fyzickou kondici a může být problém se navzájem přizpůsobit v běhu tak, aby oba dva jedinci byli přibližně stejně namáháni. Neměla by nastat situace, kdy jeden udýchaně nemůže skoro mluvit a druhému naopak cvičení nic nedává.

V článku Balancing Exertion Experiences [16] popisují svůj přístup k dané problematice. Každý z joggujících partnerů má při sobě chytrý telefon a měřič srdeční frekvence. Na telefonu mají nastavenou svojí ideální, cílovou srdeční frekvenci každý dle své fyzické kondice, případně dle doporučení doktora. Jestliže oba partneři mají srdeční frekvenci relativně stejnou vůči své cílové, pak je vše v pořádku. Partneři mohou běžet několik minut s cílovou srdeční frekvencí, poté se vyhecují, že na chvíli zrychlí a běží např. na 120% své cílové frekvence. Pokud nastane situace, kdy první partner běží na 80%, druhý na 110%, jak vhodně donutit prvního zrychlit a druhého zpomalit?

Autoři článku přišli se zajímavým řešením. Pomocí sluchátek mohou simulovat vjem, kdy se partneři slyší vedle sebe, kdy jeden slyší druhého před sebou, případně za sebou. Ve výše uvedeném příkladu by partner běžící na 110% slyšel spoluběžce za ním, což by ho donutilo zpomalit. Opačně partner běžící na 80% by slyšel toho druhého před sebou a byl by donucen zrychlit, aby se ve výsledku slyšeli co nejlépe, vedle sebe.

1.3.3. Zdravotnictví

Aplikace s adaptivní obtížností mohou pomáhat i nemocným lidem. Lidé po vážných úrazech se mnohdy učí, jak se vrátit zpět do normálního života. Při rehabilitaci lze mnohdy využít i počítačových her, které více motivují ke cvičení. Jestliže je taková hra příliš obtížná, pacient o ní brzy ztratí zájem. To platí i v opačném případě, kdy hra nutí pacienta provádět věci, které již bez problémů zvládá. Z těchto důvodů je velice vhodné obtížnost adaptivně měnit vůči konkrétním pacientům. Příkladem této aplikace je následující podkapitola Rehabilitace po utrpění mozkové mrtvice.

Druhá podkapitola v této sekci popisuje program asistující lidem trpícím demencí při jednoduchých úkolech.

Rehabilitace po utrpění mozkové mrtvice

Po cévní mozkové příhodě může dojít k částečnému až k v úplnému ochrnutí některých končetin. Pomocí různých cvičení lze tento dopad zvrátit. Mimo jiné mohou dobře posloužit jednoduché počítačové hry ovládané haptickým zařízením s adaptivním odporem a senzory. Obtížnost hry spočívá především v odporu ovladače a vzdálenosti, kterou musí osoba překonat. Jestliže se obtížnost zvolí špatně, hráč se může brzy nudit, být frustrován. V tom případě hru vypne a může být odrazen od další rehabilitace. Úkolem dynamického vyvažování hry je opět přizpůsobit hru různým lidem s různou rychlostí rehabilitace. [18]

Pomoc lidem trpícím demencí

Lidé trpící nemocemi jako je Alzheimerova choroba potřebují pomoci i při běžných úkolech jako je mytí rukou. Tento proces lze rozdělit do několika podúkolů. Puštění vody, namydlení rukou, opláchnutí rukou, vypnutí vody, usušení rukou ručníkem. Někteří pacienti některé z kroků zapomínají, a poté jim je musí aplikace slovně připomenout. Cílem bylo vytvořit aplikaci, která se bude přizpůsobovat dovednostem aktuálního uživatele. Aplikace by neměla připomínat kroky mytí rukou, které pacient zvládne bez nápovědy provést sám. Připomínání všech kroků při každém mytí rukou by mohlo vést k jeho frustraci. [19]

1.3.4. Výukové programy

Existuje velké množství vzdělávacích programů a her. Jak takovou hru udělat, aby efektivně vzdělávala úplného začátečníka, ale i již pokročilého uživatele? I zde je prostor pro adaptivní přizpůsobování se programu uživateli. Představme si simulátor výuky v autoškole, kde by se dle schopnostech začínajícího řidiče měnilo prostředí. Na začátku by žák projížděl vesnicemi s minimálním provozem. Jak by se žák zlepšoval, přibýval by provoz, dopravní značky, semafory, vjel by do města apod. Jestliže by jel příliš rychle, v dalším úseku by se objevil retardér atd.

Dále přiblížím dvojici výukových her. První se zaměřuje na na výuky hry na elektrickou kytaru. Druhá má za úkol rozšiřovat slovní zásoby předškolních dětí.

Výuka hry na elektrickou kytaru

Hra Rocksmith má zábavnou formou uživatele naučit hrát na elektronickou kytaru. Oproti Guitar Hero, Rock Band neovládáte hru speciálním plastovým ovladačem, naopak využíváte opravdovou elektrickou kytaru, kterou připojíte přes speciální kabel do USB. Lze využít kytaru zakoupenou se hrou nebo jakoukoli jinou.

V této výukové hře máte za úkol zahrát na kytaru správné akordy ve správnou chvíli. „Vše začíná jednoduchým brnkáním na jednu notu a pokračuje přes slajdy a příklepy k akordům a dalším složitějším technikám."[20] Tímto lze popsat statickou část obtížnosti, ale autoři se zaměřili i na dynamické vyvažování obtížnosti a sami to vyzdvihují ve svém propagačním videu.[21] Jestliže během hraní uděláte několik chyb po sobě, hra se zjednoduší. Např. místo každého tónu budete hrát pouze každý třetí.

Rozšiřování slovní zásoby

Peter Peerdeman se ve své diplomové práci Intelligent Tutoring in Educational Games[22] zabýval využitím DDA u výukových her. Hra Mijn naam is Haas(Moje jméno je Zajíc) je zaměřena na mladší hráče, jež si mají rozšířit svojí slovní zásobu. Hlavní postavou ve hře je zajíc, který se stává průvodcem po hře. Hráč ovládá hru kreslením různých objektů do světa zajíce. Hra se mu přizpůsobuje a dle nakreslených objektů vybírá další úkoly. Např. nakreslí-li několik mraků, začne pršet a dalším úkolem je nakreslit deštník, který by ochránil zajíce Haase před zmoknutím.

Hra při zadávání úkolů vhodně vybírá slovíčka dle úrovně hráče. Využívá se databáze 6000 slov, kde každé slovo je ohodnoceno číslem mezi 0 – 100 určující jejich obtížnost. Ohodnocení slova vyjadřuje kolik procent učitelů si myslí, že toto slovo je důležité znát pětiletými dětmi v Holandsku. Lze předpokládat, že slova s hodnotou 90-100 žáci již dobře znají a naopak slova ohodnocená 0 – 30 nejsou důležitá k naučení. Zbytek slov lze rozdělit do šesti úrovní obtížnosti. Obtížnost 1 obsahující slova s hodnotou 80 – 90 až po obtížnost 6 se slovy s hodnotou 30 – 40.

Zadání úkolu vždy obsahuje většinu slov dítěti dobře známých, zbylé tvoří prostor pro učení. Každý úkol má přiřazeno několik různě obtížných synonym a program vybírá nejvhodnější slovo dle úrovně hráče, které několikrát zopakuje v různých větách pro lepší pochopení jeho významu.

Na základě předchozích příkladů z různých odvětví lze vidět, že DDA má široké spektrum aplikací, a proto má smysl se jím zabývat.

2. Definice zábavy

Pojem zábava patří mezi velice subjektivní pojmy. Pro každého je zábavného něco jiného a svět her není výjimkou. Mezi hlavní zastupitele zkoumající tento pozitivní požitek patří Mihaly Csikszentmihalyi, který ideální stav maximální zábavy, maximálního ponoření do některé z činností nazval flow. Blíže tento stav bude popsán v následující podsekci.

Přestože se jedná o pojem subjektivní, pro použití DDA je nutné najít některé složky zábavy, které jsou změřitelné, kvantifikovatelné. Musíme být schopni zábavu měřit. Jestliže ji dokážeme změřit, můžeme to využít pro stavbu DDA algoritmů a i pro měření kvalit různých algoritmů mezi sebou. Možné metriky budou popsány dále.

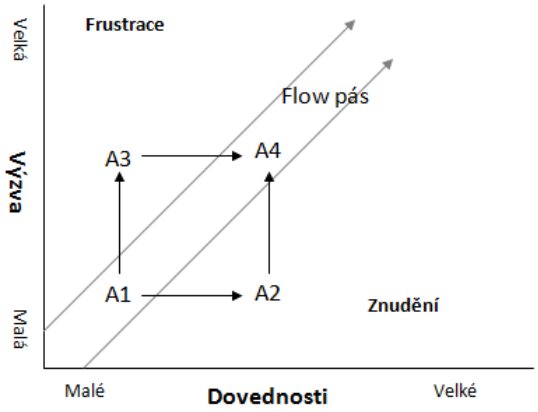
2.1. Flow

Flow (tok) je stav mysli, kdy je osoba v průběhu provozování činnosti naprosto soustředěná. Osoba ve stavu flow dosahuje většinou, na své poměry, nadprůměrných výkonů, ale přitom ji to nepůsobí výraznou námahu. Tento stav je většinou spojen s pocity energie, radosti, harmonie a seberealizace.[23] S pojmem Flow prvně přišel zástupce pozitivní psychologie Mihaly Csikszentmihalyi ve své práci Flow: The psychology of optimal experience, česky vydané pod názvem O štěstí a smyslu života.

Požadovaný stav lze charakterizovat z pohledu dovedností člověka a náročnosti prováděné činnosti. Dosáhneme ho, jestliže provádíme úkol náročností odpovídající našim schopnostem. Odtud pochází jeden z hlavních principů DDA, adaptivita obtížnosti hry dle schopností uživatele.

Jestliže se člověk seznamuje s novou činností, začíná v levém dolním rohu flow grafu 2. V tu chvíli nemá žádné dovednosti a pravděpodobně se začíná učit provádět činnost od jednoduchých částí po složitější. V knize [24] uvádí Csikszentmihalyi jako příklad takové činnosti hraní tenisu a vysvětluje to na obr. 2. Alex začíná hrát tenis, a tedy se nachází ve fázi označené A_1. Alex v této chvíli neumí vůbec hrát tenis a začíná s tréninkem trefování se do míčku. Není to příliš obtížné, ale Alex si to užívá, protože náročnost tohoto úkolu přesně odpovídá jeho schopnostem.

V této chvíli se Alex pohybuje v tzv flow pásu. Na stejném místě v diagramu nemůže zůstat Alex věčně. Jak danou činnost procvičuje, stává se v ní čím dál lepší, přestává ho to bavit, dostává se do nudy znázorněné v grafu A_2. V opačném případě se může stát, že potká zkušeného hráče. Hra proti němu je mnohem náročnějším úkolem a převyšuje Alexovy schopnosti. Alex se v takovém případě dostává do stavu úzkosti a stresu A_3.



Obrázek 2. Příklad pohybu jedince ve flow grafu.

V obou zmíněných příkladech se Alex nachází mimo flow pás a bude se snažit do něj opět dostat. V horším případě hru vzdá a další stav A_4 již v grafu nebude. V případě, že je ve stavu A_2, je dalším logickým krokem začít obtížnější úkol, vytyčit si nový cíl odpovídající jeho schopnostem. Ve stavu stresu A_3 má Alex jedinou možnost, zlepšit své dovednosti, aby se opět vrátil do flow pásu. Teoreticky může ubrat na výzvě, náročnosti úkolu, ale jak je jednou člověk vystaven takové výzvě, je pro něj těžké ji ignorovat a vzdát se jí. [24]

2.1.1. Flow ve hrách

Nás bude více zajímat napojení flow na vývoj počítačových her. Jenova Chen ve své diplomové práci Flow in Games[25] vybral několik flow komponent, které označil za hlavní při návrhu hry. Dle Chena musí hra obsahovat následující tři elementy, aby hráč dosáhl stavu flow.

1. Předpokladem je, že hra sama o sobě je pro hráče odměňující. Hráč sám o sobě chce hru hrát.
2. Hra nabízí správnou náročnost úkolů vzhledem k hráčovým schopnostem, což mu umožňuje více se do hry ponořit.
3. Hráč potřebuje cítit kontrolu nad prováděnou činností.

Jsou-li splněny všechny tři body, hráč může ztratit pojem o čase a zcela se do hry ponořit.

Stavem flow ve hrách se dále zabýval Lennart Nacket, který ve svém článku[26] shrnuje spojení flow s počítačovými hrami od různých autorů. Sweetserův a Wyethův herní flow obsahuje osm složek vycházejících z 9 složek flow od M. Csikszentmihalyi ([24], [27], [23]).

1. Jasné cíle
2. Zpětná vazba
3. Výzva
4. Hráčovi dovednosti

5. Koncentrace
6. Pocit kontroly
7. Ponoření se do činnosti
8. Sociální interakce

Jasné cíle : Hráč by měl být vždy schopen kognitivně zpracovávat herní mise, úrovně, úkoly. Aktuální úkoly by měly být vždy jednoznačné a nematoucí.

Zpětná vazba : Hra by měla vždy uživatele informovat o výsledku provedených akcí. Hráč by měl být seznámen, jak se blíží, či oddaluje od splnění cíle hry.

Výzva : U příliš jednoduchých her se nemohou uživatelé ponořit do hry. Hra musí být výzvou. Podstatné je rozlišovat náročnost ve formě špatně navrženého uživatelského rozhraní a ovládání a výzvu jako část herního designu. Špatné ovládání není nikdy žádoucí.

Hráčovi dovednosti : Hra by měla být navržena tak, aby hráči umožňovala efektivní získávání herních dovedností. Hra by měla brát v úvahu i možné dovednosti získané hráčem z jiných her.

Koncentrace : Hráč musí být do hry zcela ponořen, věnovat ji veškerou svojí pozornost.

Pocit kontroly : Hráč má mít pocit, že ovládá dění ve hře. Tento bod může být kritickým u návrhu DDA. Zjistí-li hráč, že jeho úspěch ve hře není zcela závislý na jeho výkonu, ztrácí pocit kontroly.

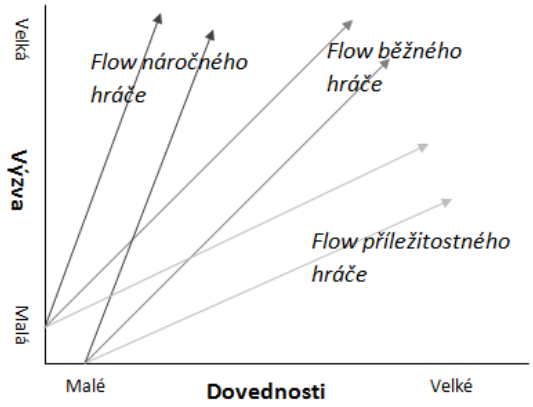
Ponoření se do činnosti : Podobné koncentraci. Hra má pohlcovat a udržovat hráče v maximální pozornosti, ale tak, aby mu to bylo stále příjemné.

Sociální interakce : Tento bod je přidán oproti prvotnímu flow. K dokonalému zážitku potřebuje člověk další lidské spoluhráče a protihráče.

2.1.2. Zóna flow pro různé hráče

Vraťme se znovu ke flow diagramu 2. Dle příkladu s Alexem a jeho učením se tenisu by se mohlo zdát, že pro každého je ideální udržovat zcela vyrovnanou hodnotu schopností a náročnosti úkolů a udržovat uživatele v úzkém flow pásu.

Bohužel takový flow diagram nebere v úvahu individualitu hráče. Existují zkušení hráči, kteří mají rádi větší výzvy než jsou v tu chvíli schopni zvládnout. Naopak existují příležitostní hráči, kteří netouží po velkých výzvách a nejraději se budou pohybovat lehce pod flow zónou. Těchto specifik si všímá např. práce [8]. Ideální průběh hry pro příležitostné/začínající hráče, běžné a zkušené hráče znázorňuje graf na obr. 3. Mnohé práce tato specifika opomíjejí a často je jejich cílem upravit obtížnost hry, aby byl vyrovnaný počet hráčových výher a proher a už neberou v potaz, že někteří hráči přestanou hrát, když budou z poloviny pokusů prohrávat.



Obrázek 3. Flow zóna a specifika pro příležitostné a zkušené hráče.

2.2. Metriky zábavnosti

Hlavním cílem počítačových her je pobavení hráče. Z tohoto důvodu je žádoucí umět zábavu přímo, či nepřímo změřit. Pro techniku DDA je existence takové metriky zcela zásadní. Bez ní by hra nevěděla, jakým způsobem se má přizpůsobovat hráči a neuměla by ohodnotit, jestli to dělá dobře.

Metriky můžeme rozdělit do několika kategorií. Některé metriky jsou specifické pro konkrétní hru, jiné jsou obecně použitelné pro většinu existujících her. Dále jsou metriky, jež vycházejí pouze ze stavu hry, softwaru. Protikladem mohou být speciální metriky měřící hodnoty z vnějšího světa. Příkladem může sledování srdečního tepu [16]. Dále lze využít webkameru, senzory na herních i neherních zařízení. Kromě tepu lze využívat např. pevnost stisku joysticku, měnící se odpor kůže, teplotu těla[1]. V této práci se pro naše účely zaměřím na metriky univerzálně použitelné a získané ze stavu hry.

Zábava ve hrách se často zjednodušuje do podoby náročnosti hry vzhledem k dovednostem hráče. Z tohoto důvodu velké množství přístupů pracuje s metrikami, které zábavu měří nepřímo, měří obtížnost hry. Často používanou metrikou je hodnota win-rate, poměr vítězstvích hráče ku jeho prohrám. Hodnota 0,5 značí polovinu výher a polovinu proher. Mnohé přístupy hodnotu 0,5 berou jako ideální. V tomto případě se hra snaží obtížnosti přiblížit co nejpřesněji dovednostem hráče. Jak bylo naznačeno v předchozí sekci, ne každý hráč ocení polovinu proher. Zde je prostor pro kombinaci statické a dynamické obtížnosti. Při statické obtížnosti si hráč vybere jednu z předem daných obtížností, např. začátečník, pokročilý, expert, kterým budou odpovídat hodnoty win-rate 25, 50, 75

Další metriky využívají heuristiku, která numericky ohodnotí stav hry pro každého hráče. Numerickou hodnotu nazveme ziskem. Velikost zisku nepřímo vyjadřuje pravděpodobnost výhry hráče. Samotná heuristika je herně specifická, ale prakticky u každé hry lze nějakou vymyslet, a proto jsou metriky, které ji využívají, obecně použitelné.

Na základě této heuristiky definujeme status funkci dle algoritmu Dynamická úro-

veň[28] a rozšíříme jí ze dvouhráčových her na jednohráčové a vícehráčové hry. U jednohráčové hry se status hráče přímo rovná jeho zisku. V ostatních případech najdeme hráče s největším ziskem. Status tohoto hráče se rovná rozdílu jeho zisku a zisku druhého hráče v pořadí. Pro ostatní hráče se status spočítá jako jejich zisk mínus zisk hráče s největším ziskem. U her s dvěma hráči bude status jednoho hráče číslo opačné ke statusu druhého hráče.

Je-li status kladný, hráč má větší šanci na výhru. V případě záporného statusu je větší pravděpodobnost hráčovy prohry. U dvou a vícehráčových her má vždy jeden hráč status kladný, ostatní mají status záporný.

Příkladem jednoduché heuristiky ve hře dáma bude počet zbývajících kamenů hráče. U hry Člověče, nezlob se může být jednoduchou heuristikou suma vzdáleností figur jednoho hráče od cíle.

V článku [28] na základě zmíněné heuristiky přicházejí k třem metrikám. Počet změn ve vedení, napínavost během hry a konečný náskok. Všechny 4 zmíněné metriky (3 předchozí + win-rate) lze dobře využít v teorii her pro ohodnocení jednotlivých algoritmů a porovnání jich mezi sebou.

Metriky doplníme o dalších pět nových. Počet výměn hráčů ve vedení nemusí být dostatečný. Mohlo by se stát, že každý z hráčů by byl během hry 2 krát ve vedení, ale jeden z nich by byl ve vedení 90% času a zbylí hráči zbývajících 10%. Z tohoto důvodu zavedeme metriku s jednoslovným názvem vedení. Další metrikou je svoboda. Hráč může ztratit zájem o hru, jestliže nemá možnosti volby a existuje-li jenom malé množství možných tahů.

Poslední 3 metriky jsou spjaty s mým DDA přístupem popsaným dále ve 4. kapitole a jsou použitelné pouze u her, které jsou nedeterministické nebo které nejsou s úplnou informací. V případě neúplné informace potřebujeme, aby hra obsahovala informace, které nejsou známy ani jednomu z hráčů. Příkladem můžou být zakryté karty v balíčku, ale ne karty v soupeřově ruce. My se v rámci našeho přístupu budeme snažit náhodné jevy a neznámou informaci v rámci pravidel hry ovlivňovat, a proto se vyskytla potřeba dalších tří metrik.

Nové metriky spravedlnost a náhodnost porovnávají statusy hráčů před a po odkrytí skryté informace alespoň jednomu z hráčů (např. líznutí karty), nebo před a po projevu náhodného jevu (např. hod kostkou). Poslední metrikou je uvěřitelnost. Hra by nebyla uvěřitelná, jestliže by jednomu hráči padlo pětkrát za sebou 6, nebo pokud by si vytáhl všechny karty stejné barvy.

Ve zbytku sekce popíši výpočet všech nastíněných metrik.

2.2.1. Poměr vítězství

První metrika poměr vítězství vychází přímo z hodnoty win-rate. Hru považujeme za více zábavnou, jestliže se hráči o vítězství dělí v průběhu několika her rovnoměrně. Poměr vítězství se spočítá jako směrodatná odchylka hodnot win-rate jednotlivých hráčů.

2.2.2. Změna vedení

Hráč, který má kladný status, je ve vedení. Jestliže se dostane do vedení jiný hráč, zaznamená se to. Metrika měří počet výměn hráčů ve vedení během hry od začátku do konce. Je zde předpoklad, že hra je více zábavná, jestliže se hráči častěji ve vedení střídají, a tedy není pořadí hráčů stejné na začátku, během a na konci hry.

U hry s jedním hráčem se status hráče rovná přímo jeho zisku. Metrika změna vedení u hry s jedním hráčem se rovná počtu změn znaménka statusu tohoto hráče.

2.2.3. Náskok

Metrika náskok je závislá pouze na aktuálním kole a. Udává náskok prvního hráče nad ostatními. Hodnota náskoku se přímo rovná statusu prvního hráče. Status prvního hráče v kole a označíme s_a^*. Potom výpočet metriky lze jednoduše zapsat jako absolutní hodnotu statusu :

$$naskok(a) = |s_a^*|$$

Absolutní hodnotu můžeme vynechat u her více hráčů, kde status vedoucího hráče není nikdy záporný.

Hru budeme považovat za zábavnější, jestliže hodnota náskoku bude co nejmenší.

2.2.4. Napínavost hry

Napínavost je dána průměrným náskokem prvního hráče před druhým během hry od prvního do aktuálního kola a. Náskok v i-tém tahu je dán statusem hráče ve vedení, označíme jej s_i^*. Vzorec pro výpočet napínavosti vypadá následovně :

$$napinavost(a) = \frac{1}{a}\sum_{i=1}^{a}|s_i^*|$$

Stejně jako u náskoku můžeme absolutní hodnotu vynechat u vícehráčových her. Opět čím nižší hodnota náskoku, tím lépe.

2.2.5. Vedení

Hra nebývá příliš zábavná, jestliže po většinu času je stejný hráč ve vedení. Hra je pro všechny hráče více zábavná, jestliže se každý hráč držel přibližně stejnou dobu na prvním místě.

Metrika vedení bude udávat směrodatnou odchylku časů ve vedení jednotlivých hráčů.

Výpočet střední hodnoty vedení :

$$\mu_v = \frac{1}{|P|}\sum_{p\in P} dobaVedeni(p)$$

Výpočet metriky vedení :

$$vedeni(a) = \sqrt{\frac{1}{|P|}\sum_{p \in P} |\mu_v - dobaVedeni(p)|^2}$$

U her s jedním hráčem považujeme za dobu ve vedení počet kol, kdy hráč má kladný status. Metrika vedení se zde spočte jako směrodatná odchylka počtu kol s kladným a záporným statusem.

Opět platí, že čím menší hodnota vedení, tím považujeme hru za zábavnější.

2.2.6. Svoboda

Svoboda se jednoduše spočte zprůměrováním počtu možných tahů všech hráčů od začátku hry do aktuálního kola hry.

$$svoboda(a) = \frac{1}{a}\sum_{i=1}^{a} pocetMoznychTahu_i$$

Čím větší hodnota svobody, tím lépe.

2.2.7. Spravedlnost

U nedeterministických her se nezřídka stane, že náhoda rozhodne vítěze hry. Např. u karetní hry se může stát, že i když jeden hráč hraje ideální tahy, tak přesto hru prohraje. Definujeme si spravedlnost na základě statusu všech hráčů před prvkem náhody a po něm. Rozdíl statusů před a po pro každého hráče značí, kolik a jak jsme jednotlivým hráčům pomohli, či ublížili. Hra bude nejvíce spravedlivá, jestliže všechny hráče poškodíme/pomůžeme jim stejně.

Nejdříve si spočteme pro každého hráče, jak jsme mu během hry doposud pomáhali/poškodili ($s(p)$). Proměnná $\Delta status_i(p)$ značí rozdíl statusů hráče p před a po náhodné události v kole i.

$$\forall p \in P : s(p) = \frac{1}{a}\sum_{i=1}^{a} \Delta status_i(p)$$

Následně metriku spravedlnosti vypočteme jako směrodatnou odchylku pro $s(p)$.

$$\mu_s = \frac{1}{|P|}\sum_{p \in P} s(p)$$

$$spravedlnost(a) = \sqrt{\frac{1}{|P|}\sum_{p \in P} |\mu_s - s(p)|^2}$$

Spravedlivější hry budeme považovat za zábavnější. Znovu platí, že čím menší hodnota metriky spravedlnosti, tím lépe.

2.2.8. Uvěřitelnost

Metrika uvěřitelnosti bude patřit mezi ty nejdůležitější. Dle flow musí mít hráči pocit kontroly. Jakmile získají podezření, že hra se nechová dle jejich očekávání, pocit flow se vytratí. Uvěřitelnost je silně závislá na typu hry. U každé hry je neuvěřitelné něco jiného.

Ve hře obsahující hrací šestihranné kostky každý předpokládá, že s největší pravděpodobností každému z hráčů padne během hry každé číslo od 1 do 6 přibližně ve stejném počtu. U karetních her hráč předpokládá, že od každé barvy dostane během hry stejné množství karet. V případě, že bude dostávat pouze samé srdcové karty, přestane věřit, že hra se chová náhodně, a to i v případě, že se takto náhodný jev stal.

Přestože rozdílně vnímáme uvěřitelnost různých her, můžeme nalézt společný základní kámen pro metriku uvěřitelnosti. Definujeme si pojem uvěřitelnost pole četností.

Během hry si budeme zaznamenávat četnosti výsledků náhodných jevů. Hra bude nejvíce uvěřitelná, jestliže si budou četnosti všech jevů odpovídat. Základem uvěřitelnosti pole četností bude rozptyl četností jednoho náhodného jevu var_1.

Rozptyl četností hodnot je zřídka roven nule, a to i když je jev zcela náhodný. Po třech hodech kostkou nikdy nebude rozptyl roven nule. Od var_1 odečteme minimální hodnotu rozptylu var_{min} pro daný součet všech četností jednotlivých jevů.

Nejen hry s nulovým rozdílem $var_1 - var_{min}$ jsou uvěřitelné. Z tohoto důvodu si definujeme hodnotu prahu T, při které je hra ještě stále zcela uvěřitelná. Velikost prahu je závislá na konkrétní hře. Můžeme ji určit heuristicky.

Výsledný vzorec pro uvěřitelnost pole četností :

$$uveritelnostPC(a) = \begin{cases} 0 & \text{pokud } var_1 - var_{min} \leq T \\ (var_1 - var_{min}) - T & \text{pokud } var_1 - var_{min} > T \end{cases}$$

2.2.9. Náhodnost

Poslední metrikou je náhodnost/determinismus hry. U deterministických her výsledek závisí pouze na schopnostech jednotlivých hráčů. U her nedeterministických má vliv na výsledek náhoda. Budeme měřit velikost vlivu náhody. Metrika náhodnost se bude rovnat průměrnému rozdílu statusů hráčů před a po náhodném prvku ve hře.

$$nahodnost(a) = \frac{1}{a|P|} \sum_{i=1}^{a} \sum_{p \in P} |\Delta status_i(p)|$$

Narozdíl od ostatních metrik zde nelze říct, že hry s menší hodnotou náhodnosti jsou zábavnější. Vnímání vlivu této metriky bude subjektivní.

3. Existující přístupy

Dynamicky vyvažovaná obtížnost se zdá aktuálním tématem a možná v budoucnu zcela nahradí obtížnost statickou. Většina nalezených a citovaných článků byla napsána po roce 2000, velká část z nich vznikla až po roce 2010. Není tedy překvapením, že existující přístupy pro DDA pokrývají velkou část oblastí v AI obecně. Nalezneme zde case-base reasoning, fuzzy logiku, evoluční algoritmy, neuronové sítě, mravenčí kolonie i příklady z teorie her.

Algoritmy můžeme zařadit do kategorií zmíněných v první kapitole. Klasifikaci jsem provedl na základě vědeckých článků, kde danou metodu použili. Ne ve všech případech bylo zařazení metod jednoznačné, a proto tabulka 1 reflektuje i můj subjektivní názor. Tabulka může na první pohled vypadat podivně, stejný přístup je často označen za explicitní i implicitní nebo statický i dynamický.

Explicitnost a implicitnost nevychází přímo z použité metody, ale záleží na designérovi, pro kterou z těchto kategorií se rozhodne. Z tohoto důvodu lze zařadit všechny zmíněné metody do implicitní i explicitní kategorie. Nikdy nemusí být hráč obeznámen s použitím DDA. Do explicitní kategorie jsem zařadil pouze evoluční algoritmus, POSM a dynamickou úroveň. Tyto přístupy pracují s hodnotou obtížnosti, která je snadno zobrazitelná uživateli s jejím jasným významem.

Mimo evoluční algoritmus lze všechny metody zařadit mezi online algoritmy. Přístup producent-konzument je i offline algoritmem. Byl využit u střílečky z pohledu první osoby. Staticky je zde přizpůsobován svět, když do něho hráč vstoupí. Naopak dynamicky se upravuje přesnost a účinnost střelby nepřátel během boje. POSM patří k velice obecným přístupům. Vyžaduje na designérovi návrh konečného množství obtížností a algoritmus z nich posléze vybírá nejvhodnější obtížnost v danou dobu. Z tohoto důvodu

Tabulka 1. Klasifikace metod do různých tříd. Mnohdy je zařazení nejasné, a proto je tabulka z části tvořena subjektivním pohledem.

Metoda DDA	explicitní	implicitní	statická	dynamická	NPC	svět	úkoly
Producent – konzument		✓	✓	✓	✓	✓	
Case-base reasoning		✓		✓	✓		
Fuzzy pravidla		✓		✓	✓		
Evoluční algoritmus	✓	✓	✓			✓	
Mravenčí feromony		✓		✓			✓
POSM	✓	✓	✓	✓	✓	✓	✓
Dynamická úroveň	✓	✓		✓	✓		
MCTS		✓		✓	✓		

je zcela na návrháři, jestli obtížnost bude měněna jen před začátkem hry na základě předchozích her, nebo bude měněna i v průběhu.

Poslední trojice kategorií představuje, čím je dána obtížnost hry. Ve většině případů upravovali umělou inteligenci protihráčů. Jak už bylo zmíněno, producent-konzument upravoval kromě NPC i svět, u POSM je volba obtížnosti na návrháři. Evoluční algoritmus byl využit pro generování úrovní do logické hry. POMDP spolu se simulací mravenců byly využity u vážných her. V obou případech se měnil druh úkolu dle jeho obtížnosti.

3.1. Producent – konzument

V mnohých hrách můžeme pozorovat vztah producent – konzument mezi světem a hráčem. Jestliže hráč získá ze světa moc prostředků, hra přestává být výzvou a naopak. Má-li hráč málo prostředku (např. munice, zdraví), může být frustrován kvůli vysoké obtížnosti. Robin Hunicke popsala systém The Hamlet[7] integrovaný do Half-Life SDK[29], který vyvažuje obtížnost hry právě pomocí výměny zdrojů mezi světem a hráčem. Half-Life patří mezi klasické zástupce first-person shooter (FPS, „střílečky“).

3.1.1. Použitá metrika

Hunicke používá metriku pravděpodobnost smrti hráče. Ze série měření určí pravděpodobnostní distribuci poškození udělené hráči protivníkem během boje. Předpokládá Gaussovskou distribuci :

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{\frac{-(x-\mu)^2/2}{\sigma^2}} \tag{1}$$

Pomocí určitého integrálu F(d) můžeme spočítat pravděpodobnost utrpění poškození menší, nebo rovnu d, kterou lze využít pro určení pravděpodobnosti přežití, jestliže má hráč aktuální zdraví rovné hodnotě d.

$$F(x) = \int_d^\infty p(x)\,\mathrm{d}x \tag{2}$$

Dosazením za p(x) získáváme rovnici 3.

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_d^\infty \mathrm{e}^{\frac{-(x-\mu)^2/2}{\sigma^2}}\,\mathrm{d}x \tag{3}$$

Tento integrál lze aproximovat funkcí erf z knihovny C++. V následujícím vzorci h odpovídá aktuálnímu zdraví hráče, μ, σ pro střední hodnotu a standardní odchylku poškození od aktuálního oponenta v nějakém čase t v budoucnu.

$$F(d_t) = 1 - \frac{1}{2}(1 + erf(\frac{h - \mu t}{\sigma\sqrt{2t}})) \tag{4}$$

Během souboje se zaznamenává poškození d, které každý z protivníků udělí hráči.

Na základě těchto hodnot a vzorců výše lze přibližně spočítat pravděpodobnost smrti hráče.

3.1.2. Vyvažující strategie

Systém Hamlet mění obtížnost na základě poptávky a nabídky. Na straně nabídky může systém zasáhnout umisťováním předmětů v herním prostředí (lékárničky, munice, zbraně). Dále může přizpůsobovat účinnost a přesnost hráčových zbraních, projev brnění apod.

Na straně poptávky manipulovat s nepřáteli (změnou jejich třídy, množství, počtu jejich životů, určením místa jejich objevení se na mapě). Stejně jako u hráče lze přizpůsobit sílu a přesnost jejich zbraní.

Autoři se snaží držet hráče v tzv. „komfortní zóně“, kdy se hráč cítí relativně v bezpečí. Jestliže se v průběhu boje zvedne pravděpodobnost úmrtí nad 40%, Hamlet začne zasahovat do hry výše uvedenými způsoby.

Cílem této politiky je udržet zdraví hráče na střední hodnotě 60 se standardní odchylkou 15 bodů. Hamlet je navržen tak, aby pomáhal hráčům, kteří mají problémy, ale na druhou stranu, aby je neprotahoval za každou cenu skrz herní úrovně.

3.1.3. Výsledky

DDA systém byl ověřen při experimentu, kterého se účastnilo 20 osob různé úrovně. Každá osoba absolvovala dvě hry v délce alespoň 15 minut. V jedné hře bylo zapnuté dynamické vyvažování obtížnosti, v druhé nikoli.

Zaznamenával se počet smrtí během prvních 15 minut hry. Nakonec každý z účastníku odpověděl, která z her ho bavila více.

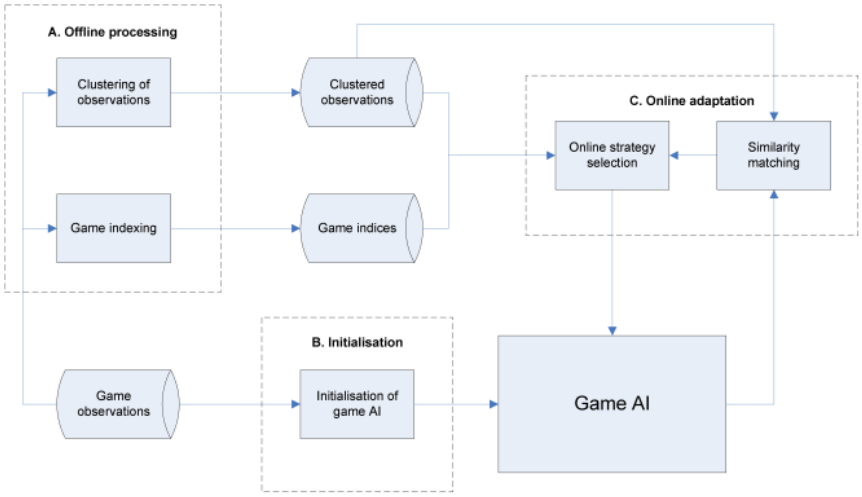
Střední hodnota a směrodatná odchylka počtu smrtí u vyvažované hry byly rovny 4 a 3,0, u nevyvažované 6,4 a 2,1.

U začátečníků se neprojevila souvislost mezi použitím/nepoužítím DDA a pocitem zábavy, pokročilí hráči více upřednostňovali použití DDA.

3.2. Case-base reasoning

Další možností pro implementaci DDA je Case-base reasoning(CBR). CBR metoda používá databázi stavů s ohodnocením jednotlivých hráčů v daném stavu a se strategiemi, které hráči v tu chvíli používali. Při rozhodování hráč vždy nahlédne do databáze, vybere několik stavů podobných současnému a vybere z nich ten nejvhodnější. Pokud se snažíme o co nejlepšího hráče, vybere se z podobných stavů stav, kde dosahuje hráč nejlepšího výsledku. Při aplikaci DDA bude nejvhodnějším stavem stav, kdy jsou síly vyrovnané.

Tento přístup byl využit u strategické hry Spring. [30] Proces celé adaptivní AI znázorňuje diagram Obr. 4. Začne se sběrem dat (game observation). Proběhne simulace



Obrázek 4. Proces adaptivní AI založené na CBR [30]

stovek her s různými hráči a vždy se v průběhu hry v určitých intervalech zaznamená zjednodušený popis stavu hry, použité strategie všech hráčů.

Následuje offline zpracování (A. Offline processing), které může být časově náročné. Jednotlivé stavy se ohodnotí číslem (Game indexing). Dle předem známé heuristiky se určí, který z hráčů vede a „o kolik“. Kvůli velké paměťové náročnosti a posléze náročnosti vyhledávání v databázi se data komprimují pomocí shlukování (Clustering of observations). Situace hry navzájem si podobné se nahradí situací jednou.

Při inicializaci (B. Initialisation) se nastaví první strategie AI, která se ukázala nejlepší v daném scénáři. Poslední části schématu je online adaptace (C. Online adaptation), která nesmí být náročná na výpočetní výkon, jelikož se provádí v reálném čase. V určitých intervalech během hry se změní strategie hráče (Online strategy selection) na strategii, která se ukázala nejvhodnější v podobné situaci(Similarity matching).

3.2.1. Sběr a úprava dat

Při sběru dat pro adaptivní AI ve hře Spring získali 448 567 pozorování z celkem 975 her na třech různých herních mapách. Výsledná data zabírala 1192 MB nekomprimovaně. Spouštěli se vždy hry s dvěma protihráči. Pokaždé inicializováni různými strategiemi. V tomto kontextu se strategií míní vektor celkem 27 parametrů představující důraz na různé strategické chování na vysoké úrovni. Např. parametr aircraft_rate ovlivňuje, jak moc často by měl hráč vytvářet vzdušné jednotky. Jakým způsobem se toho dosáhne už nesouvisí s těmito parametry. Dále je nutné popsat a uložit popis dané situace, jež se později použije pro vyhledávání podobných pozorování. Pozorování je popsáno 6 parametry. Fáze hry, síla materiálu, bezpečí velitele, ovládané území, ekonomická síla a počet vojenských jednotek.

3.2.2. Ohodnocení her

Každé z pozorováních se ohodnotí pomocí fitness funkce. Získané číslo určuje, kdo v dané chvíli vítězí. Fitness hodnota blížící se nule značí vyrovnané síly obou soupeřů. Vypočítá se např. z fáze hry, množství surovin a jednotek jednotlivých hráčů.

3.2.3. Shlukování pozorování

Shlukování je velice časově náročné, a proto se provádí offline. Cílem je nahradit více pozorováních jedním reprezentantem, který může být jedním pozorováním z původních dat, nebo pozorování vzniklé zprůměrováním podobných dat. Záleží na zvoleném algoritmu. Zde postačil dobře známý a jednoduchý algoritmus k-means[31].

3.2.4. Inicializace hry

V této fázi procesu se určí první strategie dynamické AI. Nejdříve se určí strategie, kterou využívá soupeř. Z ní se udělá abstrakce, jednotlivé hodnoty 27 proměnných se nahradí hodnotou z výčtu „málo“, „středně“, „hodně“. Poté se naleznou v databázi pozorování s hráči podobnými soupeři a vybere se inicializační strategie, která si vedla nejlépe proti takovému hráči. Z tohoto důvodu nikdy není vybrána neefektivní strategie jako první.

3.2.5. Online adaptace

V průběhu hry se čas od času spustí adaptace herní strategie dle aktuálního stavu hry. Tato adaptace se skládá ze dvou částí, z porovnávání stavů a výběru strategie.

Podobnost dvou pozorování je rovna váženě sumě podobnosti šestice parametrů v jednotlivých pozorováních.

$$podobnost(poz1, poz2) = ((1 + f) * (0,5 * u)) + mp + sc + cp + ep \tag{5}$$

$$\begin{aligned} f &= rozdilFazeHry(poz1, poz2) \\ u &= rozdilPoctuJednotek(poz1, poz2) \\ mp &= rozdilMaterialniSily(poz1, poz2) \\ sc &= rozdilBezpecnostiVelitele(poz1, poz2) \\ cp &= rozdilObsazenychPozic(poz1, poz2) \\ ep &= rozdilEkonomickeSily(poz1, poz2) \end{aligned}$$

Samotná selekce nejvhodnější strategie se skládá ze tří kroků. Nejdříve se z databáze shluků vybere N nejbližších sousedů k aktuálnímu stavu hry dle výše uvedeného vzorce.

Posléze se vybere menší podmnožina M stavů na základě herního indexu(fitness) dle zvoleného kritéria.

Zde se může projevit kombinace statické volby obtížnosti s dynamickou. Hráč si může před začátkem hry z volit jednu z pevně daných obtížností, např. lehká, normální, těžká. Vybere-li si běžnou obtížnost, bude algoritmus vybírat M stavů, jež mají fitness nejblíže k 0, která značí vyrovnané šance obou hráčů na výhru. Naopak hraje-li těžkou hru, algoritmus může vybírat pozorování s fitness odpovídající větší šance na výhru soupeřem.

Zbývá vybrat jedinou novou cílovou strategii. Z M stavů se vybere takový stav, kde strategie soupeře nejvíce odpovídá aktuální strategii soupeře v současné hře.

3.2.6. Provedené experimenty

Autoři článku provedli dva typy experimentů. V prvním se adaptivní hráč snažil chovat co nejlépe, jeho cílem bylo porazit nepřítele. V druhém případě měl adaptivní hráč za úkol udržovat co nejdéle vyrovnaný stav hry. Oponentem adaptivnímu hráči byl v jednom případě umělý hráč z originální hry(AAI) a v druhém případě hráč s náhodně nagenerovanou strategií. Pro srovnání pustili stejné experimenty s adaptivním hráčem, který měl vypnutou adaptivitu, jeho strategie se v průběhu hry neměnila.

Proti AAI hráč vyhrál ve 39% her, když neměl zapnutou adaptivitu, a ve 77% se zapnutou adaptivitou. Proti náhodnému soupeři vyhrál bez adaptivity v 47% případů, s adaptivitou v 64% her.

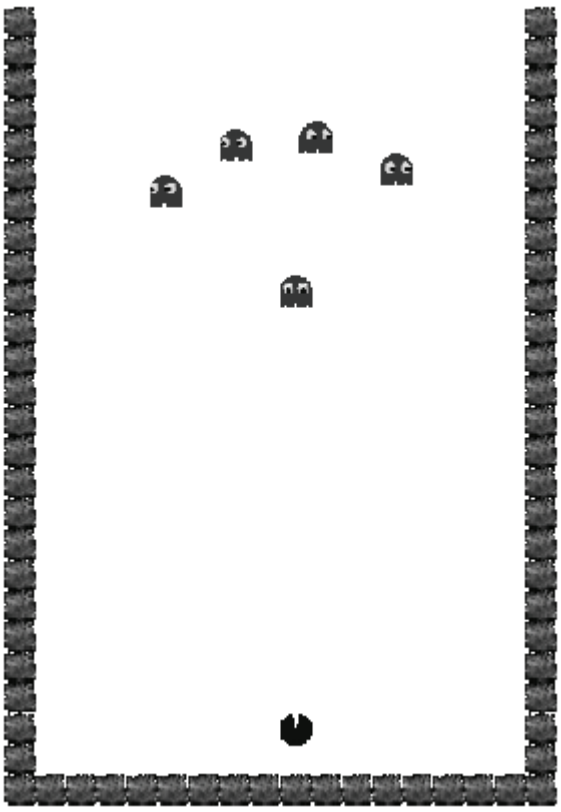
Při druhém experimentu adaptivní hráč udržoval vyrovnaný stav hry průměrně 37 minut proti AAI a 36 minut proti náhodnému protihráči. Bez DDA byl vyrovnaný stav 27 a 28 minut proti AAI a náhodnému hráči.

Výsledky obou experimentů byly pozitivní. DDA v tomto případě mělo smysl.

3.3. Fuzzy pravidla

Tento princip využívá fuzzy logiku. Fuzzy logika se od klasické logiky liší především funkcí příslušnosti. U ostrých množin prvek buď do množiny patří, nebo nepatří. Ve fuzzy logice lze popsat situace, kdy prvek do množiny patří pouze z části.

Mějme rozhodovací pravidlo ve strategické hře : Pokud je blízko soupeřova armáda a je velká, začni rekrutovat hodně vojáků. Pojmy blízko, velká, hodně jsou neurčité. Bez fuzzy logiky bychom pravidlo mohli zapsat ve tvaru Pokud je soupeřova armáda vzdálená do 100m a má sílu větší než 1000 jednotek, začni rekrutovat 100 vojáků. Nedostatkem tohoto ostrého pravidla je, že pokud bude armáda mít sílu 999 jednotek, pravidlo se neaplikuje. Tento nedostatek lze eliminovat pomocí fuzzy pravidel. Fuzzy pravidla nemají ostré hranice, kdy se aplikují a kdy naopak ne. V případě, že se některé podmínky nesplní jen těsně, pravidlo se stále provede, ale s menším efektem. V daném příkladu by to mohlo znamenat pouze vytvoření menšího počtu vojáků.



Obrázek 5. Screenshot ze hry Dead-End. [32]

Základní myšlenka DDA založené na fuzzy pravidlech je jednoduchá. Mějme databázi fuzzy pravidel, kde každé z fuzzy pravidel může být aktivní, nebo vypnuté. Jestliže se hra jeví příliš těžká, vybere se některé z pravidel, a to se vypne. Naopak v případě, kdy se hra jeví příliš jednoduchá, jedno pravidlo se znovu aktivuje. Tímto způsobem se dynamicky upravuje umělá inteligence hráčů.

3.3.1. Dead-end

Autoři přístupu zvolili pro jeho testování hru jednoduchou hru Dead-end.[32] Hráč je obklopen třemi stěnami a na čtvrté straně je východ. Jeho cílem uniknout tímto východem. Jeho snažení brání nepřátelské figury, které se naopak snaží hráče chytnout. Hráč má několik životů. Jestliže dojde ke kolizi hráče s nepřátelským duchem, jeden život ztratí. Ztratí-li všechny životy, hráč prohrává. Ve hře je navíc nastaven časový limit. Vyprší-li tento limit, hráč také prohrává. Naopak dostane-li se ven s alespoň jedním životem, vyhrává. Všechny postavičky se mohou pohybovat osmi směry.

Duchy lze rozdělit do dvou rolí. Duch nejníže vzhledem k souřadnici y tvoří předvoj a jistým způsobem ovládá ostatní duchy, obránce.

Rule	Antecedent							Consequent
	distToPlayer		*distToExitY*		*distToPlayerX*	*getPast*		
	near	*far*	*near*	*far*	*far*	*false*	*true*	
1	V		V			V		*chasePlayer*: *many*
2	V		V				V	*chasePlayer*: *many*
3	V			V		V		*chasePlayer*: *many*
4	V			V			V	*chasePlayer*: *many*
5		V	V			V		*chasePlayer*: *few*
6		V	V				V	*chasePlayer*: *many*

Obrázek 6. Část fuzzy pravidel pro předvoj. [32]

3.3.2. Fuzzy pravidla

Duši se rozhodují na základě 5 vstupních fuzzy proměnných, jedné ostré proměnné a pěti výstupních proměnných. Vstupními proměnnými pro pravidla předvoje jsou vzdálenost k hráči, vzdálenost k východu na ose y, vzdálenost k hráči na ose x a binární proměnná, jestli duch už prošel kolem předvoje. Výstupními proměnnými jsou proměnné pro akce, které může duch provést – chytej hráče, nebo ustupuj od hráče.

Vstupní fuzzy proměnné pro obránce jsou vzdálenost k hráči, vzdálenost k předvoji, vzdálenost k nejbližšími jinému obránci na ose x a stejná binární proměnná značící, jestli byl už duch obejit.

Možné akce obránců jsou – chytej hráče, přibliž se k předvoji, oddal se od předvoje, oddal se od nejbližšího jiného obránce. Příkladem fuzzy pravidla pro předvoj z kompletní tabulky 40 pravidel z [32] : Pokud je hráč blízko předvoje a je blízko východu a hráč ještě neprošel kolem předvoje, pak chytej hráče velmi. Viz první pravidlo z následujícího obrázku Obr. 6.

Uvedené příklady fuzzy proměnných a pravidel by měli pro základní představu postačovat. Kompletní seznam fuzzy pravidel a detailnější popis jednotlivých proměnných včetně grafů funkce příslušnosti lze najít v již zmiňovaném zdroji [32].

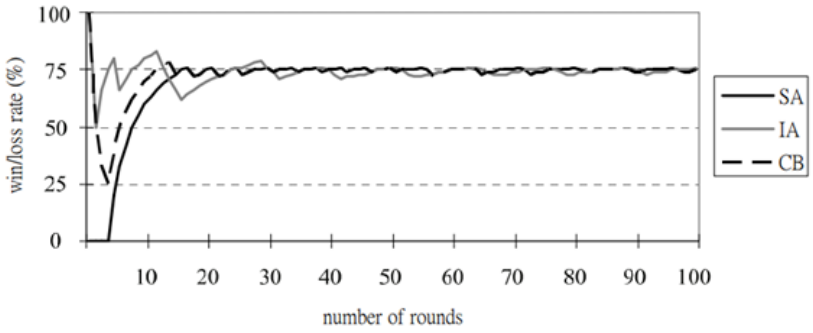
3.3.3. Adaptivní změna počtu pravidel

Jestliže se soupeř řídí všemi 40 pravidly, chová se velmi inteligentně. Hráč si na začátku hry může vybrat statickou část obtížnosti. Může ovlivnit požadovaný win-rate. Pokud tak neučiní, nastaví se win-rate na hodnotu 50%. Každá hra trvá maximálně 20 vteřin. V případě, že hráč vyhraje a aktuální poměr vítězství/proher je větší než cílená hodnota, hra se musí ztížit aktivováním momentálně vypnutého pravidla.

Naopak v případě, že hráč prohraje a aktuální hodnota win-rate je menší než cílená, hra je příliš těžká, zjednodušení proběhne deaktivací jednoho z pravidel.

Pokaždé, když se duch rozhodne dle jednoho z pravidel, zaznamená se to. Výsledný příspěvek se u pravidel obránců vydělí 4, jelikož jsou ve hře 4 obránci a jen jeden předvoj.

Pokud má dojít k deaktivaci pravidla, deaktivuje se pravidlo, které bylo využito nejméně krát, ale alespoň jednou. Kdyby se vyřadilo pravidlo, které se nevyužilo, hráč by nemusel vůbec změnu obtížnosti v příští hře poznat, jelikož by se mohli duchová



Obrázek 7. Přibližování se k požadovanému win-rate 75% během 100 kol proti třem různým hráčům. [32]

chovat zcela stejně. Pokud by se deaktivovalo pravidlo, které používal soupeř nejvíce, mohla by být změna obtížnosti příliš drastická a mohlo by to vést k neočekávaným výsledkům.

Reaktivace pravidla je o něco složitější proces. Nejdříve se všechna pravidla rozdělí do skupin dle jejich konsekventů. Příspěvek celé skupiny pravidel je roven sumě příspěvků jednotlivých pravidel ve skupině.

Poté se spočte suma vstupních příslušností hodnot pro každé z pravidel deaktivovaných dříve. Nakonec se zaktivuje pravidlo s největší sumou vstupních příslušností. Jestliže existují dvě pravidla se stejnou hodnotou sumy, zvolí se pravidlo jehož skupina má nejnižší příspěvek.

3.3.4. Výsledky

Na závěr připojuji jeden z grafů ukazující adaptivnost AI k třem různým hráčům s požadovanou hodnotou win-rate 75% Obr. 7. Ke srovnání na cílovou hodnotu došlo kolem 25. hry. Znamená to, že stačilo přibližně 25 her k upravení množiny pravidel, aby hráč vyhrával v 75% případů, což byla cílová hodnota win-rate.

3.4. Evoluční algoritmus

K zástupcům statické DDA patří evoluční algoritmy (EA). Evoluční algoritmy zpravidla vyžadují velký výpočetní výkon, a proto nejsou vhodné pro realtime použití. Využitím EA pro DDA se zabývají např. články Automatic Generation of Game Elements via Evolution[6] a Making Racing Fun Through Player Modeling and Track Evolution[33]. V prvním ze zmíněných článků využili EA pro generování různě obtížných úrovní logických her, naopak v druhém generovali závodní trať složenou z různých úseků.

Evoluční algoritmy jsou inspirovány evolucí známou s přírody, kde se nejsilnější jedinci dále množí a naopak nejslabší hynou, což vede ke stále silnějším generacím. Umělá evoluce je iterativní proces, který neustále opakuje několik kroků. Vždy se vybere několik jedinců, kteří se zkříží a vzniknou noví potomci. Následně může proběhnout mutace,

která může zanést do populace gen, který se v ní nevyskytoval. Noví potomci poté nahradí nejslabší a celý proces se opakuje.

Definice jedince, způsobu výběru nejsilnějších, křížení a mutace je vždy závislá na problému, kde je evoluční algoritmus použit. Konkrétní případy budou popsány zvlášť u každého problému v podsekcích této sekce.

3.4.1. Generování bludišť

V práci [6] byl evoluční algoritmus použit pro generování logických hádanek s předem zadanou obtížností.

Testovacími hrami dva typy bludišť. V obou hrách byl stejný cíl, nalézt cestu mezi dvěma body. Obě bludiště dále sdílely hrací plán složený ze čtverců a nepřímo znázorněné překážky. Obtížnost hry je dána minimálním počtem kroků nutných k průchodu bludištěm. Neplatí zde zcela přímá úměra, obecně neplatí, že čím větší minimální počet kroků k dosažení cíle, tím je hra těžší. V případě, že bychom minimální počet kroků zcela maximalizovali, získáme bludiště, kde na začátku každý z tahů vede blíže k výhře.

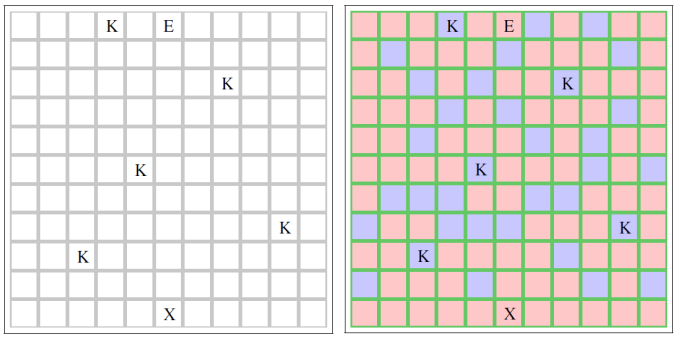
Hráč nemá dostatek možných tahů na výběr, a tedy je bludiště jednodušší než kdyby minimální počet kroků byl menší, ale naopak počet možných tahů by se zvýšil a ne všechny tahy by vedly blíže k cíli. Tuto hypotézu podporují informace získané při automatickém průchodu bludišť nutného pro ohodnocení obtížnosti hry. Algoritmus řešící bludiště se střední hodnotou minimálního počtu kroků pro daný typ bludiště běžel kratší dobu než pro bludiště s hodnotami blízkými minimu a maximu minimálnímu počtu kroků.

V první hře hráč pohybuje šachovou figurkou dle jejích platných tahů. Dále se na hracím plánu vyskytují nepřátelské šachové figury, které se nepohybují. Tyto figury ohrožují některá hrací pole. Jestliže hráč vkročí svou figurou na ohrožené hrací pole, prohrává. Jeho úkolem je figurku navést z její počáteční pozice do cílové a přitom nevkročit na žádné z ohrožovaných polí. Hráč nemá přímo označena ohrožená pole, musí využít své představivosti. Před generováním bludišť byly vždy pevně nastaveny druhy figur na hrací ploše, figura hráče a jeho startovní a cílová pozice. Cílem genetického algoritmu bylo rozmístit nepřátelské figury. Příklad jednoho bludiště na obr. 8.

Gen byl složen z pozic figur. Pro stejný druh figur nezáleželo na jejich pořadí. Bylo použito uniformní křížení. První ze dvou potomků má stejnou šanci pro získání pozice každé z figur jak od matky, tak i otce. Zbytek získá druhý potomek. Jestliže se po křížení nacházely u jednoho z potomků dvě figury na jedné pozici, potomci se zahodili, křížení se opakovalo. Při mutaci byla pozice jedné z figur náhodně přegenerována na ještě neobsazené pozice.

Druhou hrou bylo pestrobarevné bludiště. Každý čtverec bludiště má přidělenou jednu barvu z předem dané uspořádané množiny barev. Hráč se může mezi dvěma čtverci pohnout pouze v případě, kdy přechází mezi čtverci s barvami stejnými, nebo sousedními v rámci uspořádání. Opět při nepovoleném tahu hráč prohrává.

Gen je zde tvořen celým bludištěm uspořádaným po řádcích do jednoho řetězce o



Obrázek 8. Příklad jednoho vygenerovaného bludiště se 6 jezdci a hráčem ovládajího věž. Vstup do bludiště je označen E, výstup X. Vlevo z pohledu hráče, vpravo znázorněné ohrožované pozice. [6]

délce počtu čtverců bludiště. Bylo použito dvoubodové křížení a při mutaci se náhodně vybral jeden čtverec a vygenerovala se mu nová barva dle uniformního rozdělení pravděpodobnosti.

U obou her EA pracoval se stálou (steady-state) populací a selekce rodičů probíhala metodou turnaje. Vždy bylo vybráno náhodně 7 jedinců z celé populace, dva nejlepší z nich byli vybráni pro křížení a noví jedinci nahradili nejhorší dvojici vybranou pro turnaj.

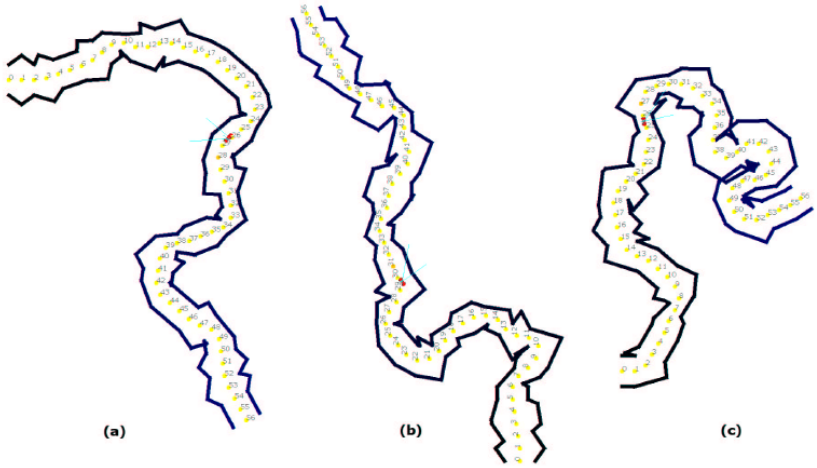
Byly použity dvě fitness funkce. První maximalizovala minimální počet kroků k vyřešení bludiště. Druhá měla parametr - cílovou hodnotu minimálního počtu kroků - a tato fitness funkce minimalizovala rozdíl skutečného minimálního počtu kroků od požadovaného. Pro výpočet této hodnoty autoři článku využili metodu dynamického programování.

V proběhnutých experimentech algoritmus dokázal vyvinout bludiště s předem zadanou obtížností, a tedy splnil svůj účel.

3.4.2. Generování tratě

Cílem práce [33] bylo procedurální generování co nejzábavnějších tratí pro závodní hry. Autoři článku za základě vlastních zkušeností a zkušeností dotazovaných hráčů definovali, jaká by měla být závodní hra, aby byla zábavná:

- Hráči rádi jezdí co největší rychlostí. Trať by měla umožňovat dosažení maximální rychlosti aut.
- Naopak pouze jízda po dlouhých rovných úsecích není zábavná. Trať by měla být určitou výzvou pro hráče.
- Lidé mají rádi různorodé výzvy. Jednotlivé tratě by měly poskytovat dostatečnou variabilitu výzev.
- Ježdění ve smyku a skoky autem se zdají být jednou ze základních složek zábavy závodních her.



Obrázek 9. Příklad tří vygenerovaných tratí. První generována pro špatného hráče, druhá a třetí pro dobrého. Při generaci třetí tratě byla použita pouze jedna z fitness funkcí. [33]

Pro zjednodušení se generovaná trať skládala s předem daného počtu stejně dlouhých úseků.[1] Úsek mohl být rovný, nebo zatáčkou vlevo, či vpravo. Každá ze zatáček měla na výběr mezi 3 ostrostmi zatáčky. Dále gen obsahoval jednu ze tří šířek konce úseku. Začátek úseku vždy musel odpovídat šířkou konci úseku předchozího. Poslední variabilitou bylo umístění překážky na jeden z krajů úseku, nebo doprostřed. Dalším zjednodušením bylo vypuštění podmínky na kruhové trati. Trať byla pouze virtuálně kruhová - na jejím konci se hráč teleportoval na začátek se stejnou rychlostí a směrem jízdy. Křížení genů nebylo využito. Mutace změnila náhodně jeden z úseků na nový dle uniformní pravděpodobnosti.

Metodou selekce byl kaskádový elitismus. Nejdříve se nagenerovalo 100 jedinců. Dle první fitness funkce se vybralo 50 nejlepších jedinců, kteří postoupili do druhého kola. V druhém kole se použila jiná, druhá fitness funkce, která vyřadila dalších 20 nejhorších jedinců. Ze zbylých 30 skončilo 15 nejlepších dle poslední třetí fitness funkce. První fitness funkci minimalizovala rozdíl cílené(předem dané parametrem) a skutečné hodnotě průměrné rychlosti na trati. Druhá fitness funkce maximalizovala maximální dosaženou rychlost na trati. Cílem bylo, aby trať obsahovala alespoň nějakou posloupnost rovných úseků. Poslední fitness funkce se zaměřila na měření množství výzev na trati. Vycházela z variance rychlosti na trati a počtu ujetých kol za danou dobu.

Každá z vygenerovaných tratí byla vyzkoušena virtuálním hráčem nahrazující lidského hráče založeného na neuronových sítí. Vstupem do sítě byla aktuální rychlost, směr k dalšími checkpointu a 6 senzorů sledujících překážky.

Během experimentů byly mimo jiné vytvořeny tři tratě zobrazené na obr. 9. Dle nastaveného parametru měla být první trať jednoduché, další dvě složitější. Závěrem lze říci, že se povedlo vygenerovat tratě s cílenou obtížností, kde první je bez ostrých zatáček, ale naopak v dalších dvou se ostré zatáčky objevují.

[1] Délka úseku dána délkou střední čáry.

3.4.3. Mravenčí feromony

Optimalizace pomocí simulace umělých mravenčích kolonií patří mezi známé přístupy inspirované přírodou. Princip optimalizace mravenčími koloniemi je následující :

1. Daný problém se vhodně modeluje jako graf skládající se z uzlů a hran mezi nimi
2. Umělí mravenci se pohybují po hranách a zanechávají na nich feromon. Čím lépe si vedou, tím více feromonu zanechají.
3. Mravenci se na každém uzlu rozhodují, kterou další hranou se vydají. S větší pravděpodobností zvolí hrany, na kterých je více zanechaného feromonu.
4. Časem feromon vyprchává, je třeba obnovovat.
5. Optimalizace končí, jestliže se ustálí cesty v grafu.

Tímto algoritmem se inspirovali vývojáři jedné ze serious games určené pro rehabilitaci částečně ochrnutých končetin lidí, jež utrpěli mozkovou mrtvici[34].

Cílem práce bylo vytvořit hru, která budu procvičovat znovu ovládnutí pohybu ruky. DDA mělo za úkol adaptivně měnit obtížnost hry a přizpůsobovat se individuálním potřebám pacientů.

Výsledkem je jednoduchá hra odehrávající se na desce o rozměrech přibližně 1,5m na 1,5m rozdělená do buněk menší velikosti. Hra vždy označí některou z buněk za cílovou a pacient se jí má pokusit dosáhnout pomocí své ruky. Algoritmus mravenčích feromonů zde byl použit pro určení, které buňky jsou pro pacienta už snadno dosažitelné, a které ne.

Hráčův profil

Schopnosti pacienta jsou zaznamenávány do tabulky Zóna schopnosti(ability zone). Zóna schopnosti je matice o velikosti m x n. Každá z buněk má přiřazeno reálné číslo reprezentující snadnost dosažení dané buňky. Cílem je vytvořit model obrazu schopností pacienta. Tato definice pouze popisuje strukturu a zamýšlenou funkci zóny schopnosti, ale již nezmiňuje, jak takovou strukturu vytvořit. Existuje více různých cest.

Jednou z možností je uložit do každé buňky statistickou úspěšnost dosažení buňky pacientem, jestliže to dostal za úkol. Jinou možností jsou biologicky inspirovaní umělí mravenci.

Pacientova ruka představuje mravence, který se pohybuje po mřížce. Na místech, která navštíví, zanechá feromon. Feromon zanechá i na okolních buňkách, ale o něco méně. Čím více feromonu je na buňce zanecháno, tím je pro pacienta jednodušší této buňky dosáhnout, a proto je vhodnější cílit pacientovu ruku do buněk jiných.

Zákon propagace

Nově přidaná úroveň feromonu $level_s(c)$ na buňku c mravencem s pozicí s lze spočítat následovně.

$$level_s(c) = \begin{cases} A(1 - \frac{dist(s,c)}{w}) & dist(s,c) \leq w \\ 0 & jinak \end{cases} \tag{6}$$

Ve vzorci konstanta A značí nominální úroveň feromonu, jež se přidává na buňku s pozicí mravence, konstanta w značí dosah vlivu feromonu do okolních buněk. Funkce dist vrací vzdálenost mezi dvěma pozicemi předanými argumenty.

Zákon vyprchávání

Vyprchávání feromonů je též velice důležité. Zajistí zapomenutí oblastí, které hráč zasáhl pouze náhodou. Jelikož pacientovi pohyby nejsou plně kontrolovatelné, může se stát, že neúmyslně zasáhne některé buňky, které nechce.

Z tohoto důvodu chceme, aby vyprchávali více feromony, jež byly zasáhnuty náhodou. S vědomostí, které buňky zóny schopností pacient zasáhl pohybem po cestě p můžeme upravit množství feromonů na nich.

$$F_{t+1} = F_t \frac{1}{vrcholyRychlosti(p)} \tag{7}$$

Čím více vrcholů v grafu rychlosti na dané cestě, tím více byly pohyby trhané a tedy méně koordinované.

Matice interakce

Matice interakce je maticí m x n binárních hodnot. Jednička značí možnost vygenerovat cíl hry z odpovídající buňky, 0 naopak. V případě, že je náročnost hry odpovídající aktuálním schopnostem pacienta, matice interakce se nemění a generují se z ní nové cíle k dosažení rukou. Delší série výher značí, že hra je jednoduchá a hráč může být brzy z nuděn a nemotivován ke hře, k rehabilitaci a naopak delší série proher může způsobovat frustraci se stejným dopadem, koncem rehabilitace a možná i nechutí v ní v budoucnu pokračovat. V takovém případě se matice interakce musí přepočítat.

V obou případech se vypočítá pro každou z buněk jednoduchým vzorečkem gradient.

$$grad(A_{i,j}) = \sum_{k \in \{i-1,i,i+1\}} \sum_{l \in \{j-1,j,j+1\}} A_{i,j} - A_{k,l} \tag{8}$$

V případě příliš obtížné hry bude matice interakce obsahovat 1 na místech kladného gradientu, v opačném případě při příliš lehké hře bude obsahovat 1 v místech záporného gradientu. Příklad na Obr. 10.

První matice (a) obsahuje hodnoty zóny schopností po jednoduchém pohybu „zdola nahoru“ do buňky 3, 3. V matice (b) jsou jedničky na pozicích záporného gradientu

(a)

0	0	0	0	0
0	0	0	0	0
0	0.5	1	0.5	0
0	1	1.5	1	0
0	1	1.5	1	0
0	0.5	0.5	0.5	0

(b)

0	0	0	0	0
1	1	1	1	1
1	0	0	0	1
1	0	0	0	1
1	0	0	0	1
1	0	1	0	1

(c)

0	0	0	0	0
0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0

Obrázek 10. (a) Zóna schopností, (b) Interakční matice pro stížení hry, (c) Interakční matice pro zjednodušení hry [34]

dle výše uvedeného vzorce. V matici (c) jsou naopak jedničky na pozicích kladného gradientu. Z matic lze snadno vyčíst ztížení hry u matice (b) a zjednodušení u matice (c).

3.4.4. Provedené experimenty

Autoři přístupu ho otestovali na skupině 10 lidí. Každý dostal nejdříve za úkol si představit, že drží v ruce gumu, a měl touto virtuální gumou vymazat co největší plochu. Následně každému z účastníků experimentu byly generovány pozice na desce, kterých měl účastník svou rukou dosáhnout. Cílem bylo dosáhnout co největšímu počtu pozic během dvou minut. V prvním případě se pozice generovali zcela náhodně, v druhém pomocí představeného DDA principu.

Po skončení experimentů měli účastníci odpovědět, jakou pociťovali obtížnost, frustraci a únavu během obou polovin experimentu.

Výsledky ukázaly, že v případě použití DDA účastníci dosáhli v průměru o 4,3 pozic více než při náhodném generování pozic. Dle subjektivního pocitu účastníci nepociťovali rozdíl v obtížnosti, frustraci a únavě v obou polovinách experimentu.

3.5. Částečně uspořádaná množina – Mistr

Partially-Ordered-Set Master (POSM) [35] je obecně použitelným algoritmem pro dynamicky vyvažovanou obtížnost ve hrách. POSM lze využít kdykoli a v jakémkoli herním žánru. Jediným požadavkem na vývojáře je definovat konečné množství nastavení obtížnosti, pro které existuje relace „je těžší než“. Tento předpoklad není nerealistický. Většina her v současnosti obsahuje několik nastavení statické obtížnosti. Cílem DDA je adaptivně přepínat mezi těmito předdefinovanými obtížnostmi.

Oproti jiným přístupům DDA nevyžaduje modelaci chování hráčů. Nepředpokládá jiné znalosti o konkrétní hře. Základní princip algoritmu je jednoduchý. Mistr (program) inicializuje obtížnost na prostřední ze všech obtížností dle relace „je těžší než“ $\geq$. Odehraje se část hry. Posléze se určí, jestli mistr zvolil obtížnost správně dle schopností hráče, nebo jestli hra byla příliš těžká, nebo příliš jednoduchá. Na základě této

informace upraví obtížnost hry správným směrem.

3.5.1. Algoritmus POSM

Vstupem algoritmu je částečně uspořádaná množina $(K, \geq)$ možných nastavení obtížnosti. Uspořádání je následující : $\forall i, j \in K$ píšeme $i > j$, pokud i je více obtížné než j. Po nastavení obtížnosti se odehraje kus hry a určí se hráčem pozorovaná obtížnost o_t.

- $o_t = 0$, jestliže obtížnost mistr zvolil správně a nemá se měnit
- $o_t = +1$, jestliže obtížnost byla příliš jednoduchá
- $o_t = -1$, jestliže obtížnost byla příliš těžká

Posledním vstupem algoritmu je učící konstanta $\beta \in (0, 1)$. Čím blíže je konstanta hodnotě 1, tím je učení pomalejší, obtížnost je více statická.

Kroky algoritmu POSM:

Algoritmus 1 Partially-Ordered-Set Master

1: $\forall k \in K : w_1(k) = 1$
2: **for** $i \leftarrow 1, 2, \ldots$ **do**
3: $\quad \forall k \in K : A_t(k) = \sum_{x \in K, x \geq k} w_t(x)$
4: $\quad \forall k \in K : B_t(k) = \sum_{x \in K, x \leq k} w_t(x)$
5: $\quad k_t = \arg\max_{x \in K} \min(A_t(x), B_t(x))$
6: $\quad$ Pozoruj reálnou obtížnost $o_t \in \{0, +1, -1\}$
7: $\quad$ **if** $o_t = 1$ **then**
8: $\quad\quad \forall k \in K : w_{t+1}(k) = \begin{cases} \beta w_t(k) & k \leq k_t \\ w_t(k) & jinak \end{cases}$
9: $\quad$ **end if**
10: $\quad$ **if** $o_t = -1$ **then**
11: $\quad\quad \forall k \in K : w_{t+1}(k) = \begin{cases} \beta w_t(k) & k \geq k_t \\ w_t(k) & jinak \end{cases}$
12: $\quad$ **end if**
13: **end for**

Na prvním řádku se inicializuje vektor w představující hodnoty „správnosti“ volby každé z obtížností. Na 3. řádku se uloží ke každé obtížnosti suma správnosti obtížností, které jsou těžší, nebo stejné než ona. Na 4. řádku se naopak uloží ke každé z obtížností suma správnosti obtížností, kterou jsou lehčí, nebo stejné než ona. Na pátém řádku se volí nová obtížnost dle uvedeného vzorce. Poté proběhne kus hry a algoritmus získá odpověď o reálné obtížnosti vzhledem k hráči. Na následujících řádcích se „správnosti“ jednotlivých obtížností vhodně upraví do další iterace hry.

3.5.2. Příklad s balónky

Algoritmus se lépe vysvětlí na příkladě.[36] Mějme jednoduchou hru sestřelování padajících balónků. Hráč je má sestřelit dříve než se dotknou země. Obtížností hry může být rychlost padání. Mějme předpřipraveno 10 různých rychlostí odpovídajících 10 nastavením obtížnosti.

Při inicializaci se vektor w nastaví na 10 jedniček. Při prvním běhu bude vektor A obsahovat $(10; 9; 8; 7; 6; 5; 4; 3; 2; 1)$ a vektor B $(1; 2; 3; 4; 5; 6; 7; 8; 9; 10)$. Na 5. řádku se vybere náhodně 5. nebo 6. obtížnost, protože v obou případech :

$$\max \min \{5, 6\} = \max \min \{6, 5\} = 5$$

Po nastavení nové obtížnosti 5 může hráč hrát po předem stanovenou dobu, např. 15 vteřin. Na 6. řádku se určí, jestli mistr dobře zvolil poslední obtížnost. V naší konkrétní hře, pokud hráč sestřelí všechny 95-100% balónků, hra byla jednoduchá. Jestliže spadne na zem 5-15% balónků, hra je vyvážená. Ve zbylých případech je hra příliš těžká.

V našem příkladu máme zkušeného hráče a při obtížnosti 5 sestřelil všechny balónky. Z toho vyplývá $o_t = +1$. Na řádcích 7 až 9 se provede úprava vektoru w. V tuto chvíli obtížnosti 1 až 5 nejsou vhodné, upraví se jejich správnost. Pro adaptivní konstantu $\beta = 0,5$ bude vektor $w_2 = (0,5; 0,5; 0,5; 0,5; 0,5; 1, 1; 1; 1; 1)$.

Pokračujme druhou iterací.

$A_2 = (7.5; 7, 0; 6, 5; 6, 0; 5, 5; 5, 0; 4, 0; 3, 0; 2, 0; 1, 0)$,

$B_2 = (0, 5; 1, 0; 1, 5; 2, 0; 2, 5; 3, 5; 4, 5; 5, 5; 6, 5; 7, 5)$

Pro přehlednost vektor minim z hodnot na odpovídajících si indexech.

$m = (0.5, 1.0, 1.5, 2.0, 2.5, 3.5, 4.0, 3.0, 2.0, 1.0)$

Maximum z vektoru m je hodnota 4.0, která je na 7. pozici. Hra se ztíží na 7. obtížnost.

Na uvedeném příkladě je vidět, že POSM je jednoduchým algoritmem s minimem předpokladů na znalosti konkrétní hry, a proto je použitelný pro velkou škálu užití.

3.5.3. Provedené testy

Algoritmus POSM byl vyzkoušen a testován na deskových hrách dáma a čínské šachy. [36]

V případě čínských šachů si pro experimenty vytvořili dva škálovatelné hráče(se jmény Poziční a Materiální) se 7 úrovněmi inteligence. Každý z hráčů používal jinou heuristiku. Při experimentech byl vždy jeden z hráčů POSM, nebo Poziční s úrovní 7 a druhý hráč byl Materiální postupně s úrovněmi 1 až 7. Pokaždé proběhlo 100 a her a zaznamenali

Tabulka 2. Výsledky experimentu algoritmu POSM hrajícímu proti předdefinovaným hráčům.

	Poziční 7			POSM		
	výhra	remíza	prohra	výhra	remíza	prohra
Materiální 1	97	2	1	11	78	11
Materiální 2	96	4	0	11	63	26
Materiální 3	79	21	0	7	61	32
Materiální 4	76	24	0	11	57	32
Materiální 5	54	45	1	9	62	29
Materiální 6	52	42	6	8	62	30
Materiální 7	34	58	8	4	69	27

se výhry, remízy a prohry Pozičního, respektive POSM hráče. Pro účely experimentu byla remízou označena hra, která neskončila do 100 tahů.

Výsledky zobrazuje tab. 2. Z tabulky lze vyčíst, že POSM dobře vyvažoval hru k velkému počtu remíz.

3.6. Dynamická úroveň

Autoři tohoto přístupu se zaměřili na dynamické vyvažování obtížnosti v deskových hrách. [28] Sami svůj algoritmus nijak nenazvali, my ho budeme nazývat dynamická úroveň na základě jeho hlavního principu.

Algoritmus byl navržen k adaptivitě umělé inteligence jednoho z hráčů v dvouhráčové hře.

3.6.1. Popis algoritmu

Dynamická úroveň má dva předpoklady k jejímu využití. Vyžaduje funkci, jež umí ohodnotit všechny následující stavy ve hře pomocí funkce hodnosti. Hodnost vyjadřuje kvalitu dané situace z pohledu hráče, který je právě na tahu. Určuje šanci na výhru aktivního hráče.

Druhá funkce vrací status hry. Kdo vyhrává a jak moc. Kladné hodnoty statusu značí vedení, 0 remízu, záporné hodnoty prohrávání. Nejen znaménko statusu se bere v potaz, i hodnota je důležitá. Mělo by platit, že čím větší je hodnota statusu, tím větší šance na výhru daným hráčem.

Dynamický hráč se při volbě nového tahu řídí následujícími kroky:

1. Uprav odhad soupeřovy úrovně na základě status funkce.
2. Vygeneruj všechny možné následující stavy ze stavu aktuálního.
3. Přiřaď hodnosti vygenerovaným stavům.
4. Vyber následující tah jako nejvhodnější vzhledem k odhadované soupeřově úrovně.

Pseudokód algoritmu by mohl vypadat následovně:

Algoritmus 2 Dynamická úroveň

1: $level_1 \in \langle 0, 100 \rangle$
2: **for** $i \leftarrow 1, 3, 5, \ldots$ **do**
3: $status_t = statusFnc(state_t)$
4: $$level_t = \begin{cases} 0 & \text{pokud level} + \frac{status_t}{\alpha} < 0 \\ 100 & \text{pokud level} + \frac{status_t}{\alpha} > 100 \\ level + \frac{status_t}{\alpha} & \text{jinak} \end{cases}$$
5: $S_t = nextStates(state_t)$
6: $\forall s \in S_t : r_t(s) = rankFnc(s)$
7: $state_{t+1} = s, s \in S_t$, kde $r_t(s)$ je v $\frac{level_t}{100}$ pořadí seřazených s dle $r_t(s)$
8: $state_{t+2} = opponentTurn(state_{t+1})$
9: **end for**

Na prvním řádku se určí inicializační úroveň jako odhad úrovně soupeře. V článku[24] není přesně definováno, jak počáteční úroveň volit. Možností je náhodně generované číslo, případně statický výběr hráčem z několika předdefinovaných možností. Např. „lehká“ = 25, „střední“ = 50 atd.

Třetí a čtvrtý řádek slouží k novému odhadu úrovně oponenta. Figuruje zde konstanta α, určující dynamičnost změny úrovně hráče. Konstanta je závislá na konkrétní hře. Může být volena experimentálně, nebo se může i v průběhu hry měnit např. pomocí algoritmu simulovaného žíhání.

Na řádcích 5 – 7 se vygenerují následující stavy a vybere se ten s nejvhodnější hodností dle úrovně soupeře. Příklad : existuje 20 možných tahů, které jsme ohodnotili a dle ohodnocení seřadili. Jestliže je úroveň rovna 75, zvolí se tah v $\frac{3}{4}$ seřazených tahů, tedy 5. Nejlepších tah. Po úroveň 50 se zvolí tah v polovině, tedy tah 10. ze seřazených tahů. Na osmém řádku se počká na tah oponenta a celý cyklus se opakuje od začátku.

3.6.2. Ohodnocovací a status funkce

Algoritmus používá dvě různé funkce k ohodnocení stavu hry z pohledu hráče - status funkci a funkci hodnosti. Status funkce vyjadřuje, jak vnímá svůj status ve hře člověk, naopak funkce hodnosti hodnotí stav podrobněji z pohledu počítače.

Význam funkcí lze lépe pochopit z pozice jejich umístění v algoritmu. Status funkci používá dynamický hráč pro určení, jestli z pohledu lidského oponenta prohrává, či vítězí a dle toho se upraví jeho úroveň. Funkce hodnosti již slouží k co nejpřesnějšímu vybírání dalšího tahu.

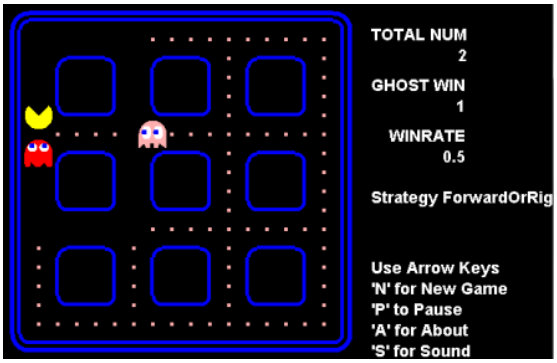
Ve hře backgammon je funkce hodnosti relativně komplexní a je složena z parametrů : počet kamenů již mimo hru, součet vzdáleností zbylých kamenů od konce, šance na zajetí svého kamenu soupeřem v příštím tahu, jsou-li všechny kameny alespoň ve 4. (posledním) kvadrantu hry apod.

Naopak status funkce popisuje stav hry méně složitě, více z lidského pohledu. Spočítá skóre každého hráče jako počet kamenů již mimo hru + suma vzdáleností zbylých kamenů od konce hry. Status je rozdílem těchto dvou hodnot.

3.7. Monte-Carlo prohledávání stromu

Autoři z Pekingské univerzity vyzkoušeli spojení Monte-Carlo Tree Search(MCTS) algoritmu s neuronovými sítěmi. [37][38]

V článcích upozorňují na nevýhodu tradičních metod DDA, kdy se obtížnost uměle mění např. přidáváním dalších a silnějších nepřátele, ale jejich inteligence zůstává stejná. Hráč se v takovém případě může cítit podveden. V toto případě se jedná o dynamického vyvažování umělé inteligence a svůj přístup aplikovali na zjednodušené verzi známé hry Pac-Man.



Obrázek 11. Zjednodušená verze Pac-Mana. Obrázek převzat z [37]

3.7.1. Pravidla hry Pac-Man

Cílem hráče hrajícího za žlutou postavu Pac-Mana je sníst všechny kuličky v bludišti a zároveň se vyhýbat nepřátelům, duchům. Pac-Man zvítězí, jestliže sní všechny kuličky, duši zvítězí, jestliže chytnou Pac-Mana. Jestliže do 55 kroků žádná z těchto událostí nenastane, hra končí remízou. Oproti původnímu Pac-Manovi jsou ve hře další úpravy.

- Bludiště je zmenšeno na velikost 16x16 a neobsahuje žádné Power upy.
- Ve hře jsou pouze dva duši místo původních 4 a mají stejnou rychlost jako Pac-Man. Z toho vyplývá, že jeden duch nikdy nemůže sám chytit Pac-Mana, duši musí spolupracovat.
- Pac-Man i duši se rozhodují pouze na křižovatkách. Jejich možné akce jsou jít vpravo/vlevo/nahoru/dolů, případně u křižovatek u kraje bludiště jejich podmnožina, procházení zdí je zakázáno.

3.7.2. Tvorba DDA pomocí MCTS

Výkon umělé inteligence soupeřů založených na MCTS záleží na množství času, které MCTS poskytneme. Čím déle algoritmus běží, tím je pravděpodobnější inteligentnější chování duchů.

Postava Pac-Mana byla simulována hráčem, který měl nastaven několik pevně daných pravidel chování.

Pomocí opakovaných simulací s pevně daným simulačním časem získali závislost poměru vítězství (win-rate) na délce simulace. Několik získaných diskrétních hodnot proložili polynomem 4. stupně.

$$y = -5,67 * x^4 + 17,6 * x^3 - 11,1 * x^2 - 0,81 * x + 65,6 \tag{9}$$

V předchozí rovnici je x časem výpočtu MCTS v ms a y výsledné win-rate. Běžný hráč chce vyhrávat přibližně v polovině případů. Vyřešením rovnice získáváme čas 105ms. Začínající hráč může upřednostnit častější vyhrávání, při win-rate 30% (duchy) algoritmus potřebuje 28ms na výpočet.

3.7.3. Využití neuronových sítí

Další nevýhodou škálování AI pomocí změn simulačního času je horší využitelnost u real-time her, kde si nemůžeme dovolit věnovat stovky ms k výpočtu AI. Tento nedostatek lze odstranit využitím neuronové sítě místo MCTS.

Neuronovou síť se snažíme naučit chování odpovídající MCTS s daným simulačním časem. Při simulacích pomocí MCTS se při každém rozhodování duchů uložil stav hry, jako 23 proměnných a výsledné rozhodnutí o novém směru každého ducha.

Takto získaná data bylo použita pro učení neuronové sítě, která posléze nahradila původní algoritmus MCTS. Vstupními proměnnými byly např. pozice a směr hráče, pozice duchů a obsah sousedních dlaždic, vzdálenost mezi duchy a hráčem, čas simulace atd. Ve výstupní vrstvě bylo po 4 neuronech pro každého ducha, kde každý neuron představoval jeden zvolený směr.

Neuronové sítě odstranily časovou divergenci pro různé AI, ale naopak přinesly do systému jistou míru nepředvídatelnosti.

4. Vlastní přístup

V této kapitole navrhnu vlastní inovativní přístup k dynamicky vyvažované obtížnosti. Mnou navržené algoritmy vychází z již existujících algoritmů teorie her a z tohoto důvodu se v první sekci této kapitoly věnuji základům z teorie her a popisu algoritmů, jež jsem převzal a upravil pro využití v DDA. V druhé sekci se již věnuji popisu vlastních algoritmů.

4.1. Teorie her

S teorií her se lze setkat především v ekonomické oblasti. Pro tuto sekci jsem vybral pouze témata z teorie her, jež budou využity. Nejdříve neformálně zadefinuji různé druhy her, hry v extenzivní formě a zero-sum hry. Následně nastíním algoritmy Minimax a Monte Carlo prohledávání stromu, které se používají pro hráče her v extenzivní formě.

4.1.1. Základní definice

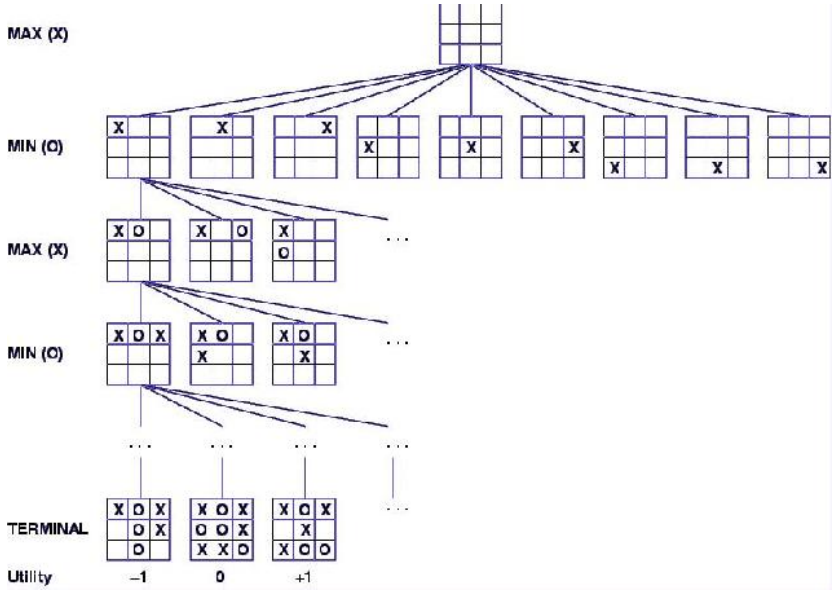
Dělení her

Hry můžeme dělit dle obsahu nedeterminismu nebo dle úplnosti informace. U deterministických her je stav hry určen pouze rozhodnutím hráčů. Naopak u nedeterministických her je stav hry určen kombinací rozhodnutí hráče a projevu náhody. Mezi deterministické hry patří např. dáma a šachy. Naopak k nedeterministickým hrám se např. řadí hry využívající hrací kostku jako je Člověče, nezlob se.

Dle úplnosti informace dělíme hry na hry s úplnou a neúplnou informací. U her s úplnou informací všichni hráči znají kompletní stav hry a žádná informace jim není skryta. Naopak u her s neúplnou informací existuje část stavu neznámá všem hráčům. Mezi hry s neúplnou informací patří většina karetních her, kdy hráči neznají karty v soupeřově ruce.

Tabulka 3. Klasifikace her dle úplnosti informace a determinismu.

	S úplnou informací	S neúplnou informací
Deterministické	šachy, dáma	Hledač min
Nedeterministické	Člověče, nezlob se	Prší



Obrázek 12. Výřez herního stromu hry Tic-Tac-Toe.

Extenzivní forma hry

Hry v teorii her můžou být znázorněny ve dvou základních formách - normální a extenzivní forma. Hry v normální formě jsou používány pro hry, kde se hráči rozhodují současně. My budeme potřebovat formu vhodnou pro hry, kde se hráči ve svých rozhodnutích střídají a kde se rozhodují na základě dřívějších rozhodnutích. Průběh takové hry můžeme znázornit pomocí grafu. Tento graf se nazývá herním stromem. Graf je orientovaný s jedním kořenem a je bez cyklů. Uzly stromu představují možné stavy hry. Stav hry mimo jiné obsahuje informaci o hráči, který je na řadě. Hrany vedoucí z uzlu odpovídají všem možným tahům hráče, který je v daném stavu na řadě. Kořen představuje počáteční stav hry a listy stavy koncové. Všechny listy mají přiřazenou utilitu/zisk pro každého hráče. Jednoduchá utility funkce přiřadí hodnotu +1 výherci a -1 všem ostatním hráčům, v případě remízy přiřadí 0 všem hráčům.

Příklad herního stromu varianty piškvorek 3x3 Tic-Tac-Toe jen na Obr. 12. V této hře se hráči pravidelně střídají, a proto uzly ve stejné hloubce odpovídají tahům stejného hráče. Aktuální hráč je v obrázku znázorněn vlevo. Ve spodní části jsou ukázány listy s utilitou pro hráče hrajícího křížky.

U nedeterministických her existují vnitřní uzly dvou typů. První odpovídá tahu aktuálního hráče stejně jako u her deterministických. Druhým typem je uzel náhody (chance node). Uzel náhody odpovídá stavu hry, kdy je na řadě náhodná událost - např. hod kostkou. Hrany z tohoto uzlu odpovídají náhodným jevům, možným stavům hry po náhodné události. Hrany jsou navíc ohodnocené číslem, které představuje pravděpodobnost jednotlivých náhodných jevů. V případě hodu kostkou budou všechny hrany ohodnoceny $\frac{1}{6}$.

U her s neúplnou informací hráči přesně nevědí, v jakém konkrétním uzlu herního stromu se nacházejí. Z jejich pohledu může být více uzlů představovat aktuální stav hry.

Tato množina navzájem od sebe nerozlišitelných uzlů se nazývá *informační množinou*. Přestože hráč přesně neví, v kterém uzlu v rámci informačního stromu se nachází, může přiřazovat různé pravděpodobnosti jednotlivým stavům na základě předchozích tahů soupeřů.

Herní strom můžeme upravit pro hry s neúplnou informací. Uzel takového stromu nemusí představovat pouze jeden stav hry, ale celou množinu stavů, odpovídající informační množinu.

Zero-sum hry

Podkategorii her tvoří zero-sum hry. U těchto her platí, že součet utilit pro všechny hráče v každém listu se rovná 0. Tuto vlastnost využívají některé algoritmy, mezi které patří Minimax s alfa, beta prořezáváním.

U dvouhráčových zero-sum her můžeme pracovat pouze s utilitou pro jednoho hráče, utilita druhého hráče se snadno odvodí. Příkladem zero-sum hry je Tic-Tac-Toe. Z tohoto důvodu byla v herním stromu na Obr. 12 ukázána utilita pouze jednoho hráče.

4.1.2. Algoritmy používané pro hry

Existuje celá řada algoritmů umělé inteligence pro hry v extenzivní formě. V této podsekci popíši základní algoritmy Minimax s prořezáváním a Monte-Carlo prohledávání stromu, na jejichž základě navrhnu vlastní algoritmy pro DDA.

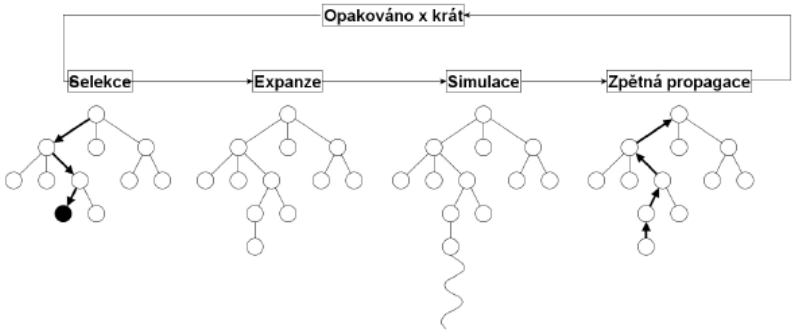
Minimax

V základní formě algoritmus Minimax pracuje pouze s dvouhráčovými zero-sum hrami. Každý z hráčů se snaží maximalizovat svojí utilitu a zároveň ví, že jeho soupeř se snaží jeho utilitu minimalizovat a tím maximalizovat utilitu svojí.

Algoritmus prochází herní strom do hloubky až do listů. Ve chvíli, kdy získá vnitřní uzel utilitu ze všech potomků, vybere z utilit tu nejmenší/největší, kterou propaguje o patro výše. V soupeřově uzlu hráč vybírá minimum z potomků, ve svém uzlu vybírá maximum z potomků. Ve chvíli, kdy se dopropagují utility ze všech potomků ke kořeni, hráč vybere tah, který odpovídá uzlu s největší utilitou.

Ne vždy je možné procházet celý herní strom až do jeho listů. Často se omezuje maximální hloubka procházeného stromu. Jestliže algoritmus skončí v uzlu, který není listem, musí ohodnotit stav hry pomocí heuristické funkce a propagovat výše pouze odhadnutou utilitu. Heuristické funkce bývají herně specifické. Např. u hry dáma může být odhadnutá utilita rozdíl počtu zbývajících herních kamenů obou hráčů.

Rozšířenou variantou pro vícehráčové a ne nutně zero-sum hry je algoritmus MaxN. U těchto her se v listech stromu nenachází pouze jedna hodnota, ale každý hráč zde má svojí utilitu. Oproti základní verzi se zde propaguje výše celá skupina utilit. Každý hráč ve svém uzlu maximalizuje svojí utilitu.



Obrázek 13. Schéma jedné iterace MCTS.

Alfa-beta prořezávání

Alfa-beta prořezávání slouží k urychlení základního algoritmu Minimax aniž by to ovlivnilo jeho chování. Urychlení spočítá v prořezávání, neprocházení větví stromu, u kterých je již zřejmé, že nemohou obsahovat lepší tah pro aktuálního hráče v daném uzlu.

K ořezávání se používají proměnné alfa a beta. Alfa představuje maximální spodní hranici a beta naopak minimální horní hranici pro utilitu řešení v dané větvi herního stromu. V průběhu algoritmu se alfa může zvětšovat a beta naopak zmenšovat. Ve chvíli, kdy se překříží, si můžeme být jisti, že další expandování uzlu nepovede k řešení a další potomky neprocházíme.

Zrychlení algoritmu je silně závislé na pořadí procházení jednotlivých uzlů. Může výpočet až dvakrát zrychlit, ale může se stát, že se výpočet vůbec nezrychlí.

Monte-Carlo prohledání stromu

Jiným přístupem k vytvoření hráče je metoda Monte-Carlo prohledávání stromu. Oproti Minimaxu nepracuje s hotovým herním stromem. Herní strom si vytváří ve svém průběhu. Na začátku je herní strom tvořen pouze kořenem představující aktuální stav hry. Poté algoritmus v iteracích opakuje 4 kroky - selekce, expanze, simulace a zpětná propagace.

Během *selekce* se vybere list z částečně postaveného stromu. Při výběru se začne v kořeni a podle předem daného pravidla se volí potomci až se algoritmus dostane do listu. Vybraný list se *expanduje*. Z listu se stane vnitřní uzel, přidají se mu potomci odpovídající možným tahům z daného stavu. Z každého přidaného uzlu se provede *simulace* hry ideálně až do konce. Je nutné, aby simulace nebyla výpočetně náročná. Z tohoto důvodu se hráči volí tahy náhodně. Nakonec se výsledek hry *propaguje* přes rodiče až do kořene.

Každý uzel stromu obsahuje stav hry, aktuální aproximovanou hodnotu hry a počet navštívení uzlu během fáze selekce.

Pravidlo pro selekci potomků není předem dáno. Jednou z často používaných metod selekce je UCT (Upper Confidence bounds applied to Trees). Tato metoda se snaží vhodně vyvážit procházení slibných větví (s vysokou aktuální hodnotou) a procházení

doposud málo navštívených uzlů. Každý z potomků je ohodnocen následujícím vzorcem:

$$d_i = v_i + C * \sqrt{\frac{lnN}{n_i}}$$

Poté se vybere uzel s největší hodnotou d_i a proces se opakuje. Ve vzorci v_i značí aktuální hodnotu i-tého potomka, n_i počet navštívení potomka, N počet navštívení rodiče. Pomocí konstanty C se volí, jestli bude upřednostňováno procházení málo navštívených uzlů, nebo uzlů s vysokou hodnotou v_i.

Volba počtu iterací je na programátorovi. Čím více iterací zvolí, tím lepšího chování dosáhne na úkor výpočetního času. Počet iterací nemusí být předem znám. Programátor se může rozhodnout pro ukončení algoritmu po určitém čase. Algoritmus má od konce první iterace vždy nějaké řešení, tah, který hráč má zvolit. Opakování iterací toto řešení zlepšuje.

4.2. Hráč prostředí

V této sekci popíši vlastní přístup k dynamickému vyvažování obtížnosti. Zmíním jeho omezení a naopak výhody oproti ostatním existujícím metodám. V podsekcích se zmíním o 4 různých algoritmech založených na společném principu - využití hráče prostředí.

Hráč prostředí(HP) je imaginárním hráčem vloženým do hry, který ovlivňuje náhodné jevy a neznámou informaci ve hře. Oproti běžným hráčům má zcela jiný cíl. Snaží se hru udělat více zábavnou. Ve druhé kapitole jsem uvedl několik metrik, které lze použít při měření zábavnosti hry. HP se těmito metrikami řídí a snaží se maximalizovat jejich váženou sumu. Suma je vážená koeficienty, které jsou specifické pro každou hru. Pomocí koeficientů lze upravit zaměření zábavnosti jen na některé její složky. Dále v textu hráč popisuje běžného hráče a HP hráče prostředí.

HP nerozlišuje hráče ovládané člověkem, nebo počítačem. Jedním z jeho úkolů je pomáhat slabším hráčům a přitom nerozlišuje, jestli pomáhá počítači, nebo člověku.

V nedeterministických hrách HP provádí tahy, které by v klasickém přístupu prováděl náhodný generátor čísel. V herním stromu nahrazuje uzly náhody (chance node).

U her s neúplnou informací je situace odlišná. V takových hrách můžeme měnit pouze informace, které jsou skryté všem hráčům. V opačném případě by se mohlo stát, že by HP měnil hru před očima některého z hráčů. Při klasickém přístupu hráči pracují s informačními množinami(IM) stavů, ale samotná hra pracuje s jedním konkrétním stavem, ve kterém se hráči nacházejí. V našem přístupu bude s IM stavů pracovat i hra. IM pro hru bude obsahovat pouze stavy, které jsou průnikem stavů ze všech IM jednotlivých hráčů. Stavy z průniku představují informaci, která je neznámá všem hráčům.

V případě, že ve hře nastane událost, která má zmenšit IM jednoho z hráčů, dostane se ke slovu HP. V dané chvíli odkrývaná informace neexistuje, HP ji musí vytvořit provedením tahu. Po provedení tahu se zároveň zmenší IM hry. Ve chvíli, kdy v IM hry

zůstane pouze jediný stav, HP již nemůže do hry nijak zasahovat.

Pro ujasnění funkce HP uvedu příklad na hře Solitaire. Solitaire je hrou pro jednoho hráče a hraje s balíčkem karet. Pravidla hry pro nás nyní nejsou důležitá. Důležitá je pouze informace, že některé karty hráč na počátku hry vidí pouze rubem vzhůru a nezná jejich hodnotu.

Při klasické přístupu se zpravidla karty před začátkem zamíchají a rozdají se před hráče, některé zakrytě. Hráč hodnotu zakrytých karet nezná, ale hra ano. Když hráč odkryje některou z karet, karta má již přiřazenou hodnotu.

V našem přístupu bychom před začátkem hry nepřiřadili konkrétní hodnoty zakrytým kartám. Pouze bychom si pamatovali, že dané karty hráč doposud neodkryl a zároveň bychom si pamatovali, které karty jsme doposud nikam nepřiřadili. Ve chvíli, kdy se hráč rozhodne některou z karet odkrýt, HP rozhodne, kterou kartu odkryl. Může vybírat pouze z karet, které jsou ještě k dispozici.

Z popisu hráče prostředí vyplývá hlavní omezení jeho využití - hráč prostředí může být využit pouze ve hrách s neúplnou informací nebo s náhodou. Tento přístup nelze využít u her jako jsou šachy nebo dáma.

Naopak má velkou výhodu oproti většině přístupů popsaných v předcházejících kapitole, které často byly založeny na ovlivňování oponentů ve hře. HP může být použit i u her pro jednoho hráče, kde žádný NPC není. HP lze také použít u vícehráčových her, kde jsou všichni hráči zastoupeni lidmi. V takovém případě také nelze použít DDA založené na adaptivitě NPC.

4.2.1. E-HillClimber

Nejjednodušší z algoritmů založených na HP prostředí je E-HillClimber. Nejdříve se dle předem dané pravděpodobnosti rozhodne, jestli bude se bude rozhodovat deterministicky, nebo nedeterministicky.

Při nedeterministickém chování HP vždy náhodně vybere jeden z možných tahů. Hra se chová zcela přirozeně stejně jako by v ní žádný HP nebyl.

Při deterministickém rozhodování HP ohodnotí všechny následující stavy pomocí vážené sumy metrik zábavnosti a vybere tah s nejlépe ohodnoceným následujícím stavem. Především v počátku hry se může stát, že více stavů je ohodnoceno stejně, v takovém případě se HP rozhodne náhodně mezi nejlepšími stavy.

Poměr nedeterministického a deterministického chování je zcela na autoru algoritmu. Je žádoucí, aby se HP alespoň v některých případech choval náhodně. V opačném případě by se mohlo stát, že by lidský hráč snadno odhalil přítomnost HP, protože by se HP vždy choval stejně.

Tento jednoduchý algoritmus má oproti následující trojice velikou výhodu. Nevyžaduje žádnou simulaci ostatních hráčů. Nepotřebuje znát ani jejich možné tahy. Z tohoto důvodu je snadno použitelný nejen u tahových her, ale i u her v reálném čase, kde je často velmi obtížné až nemožné specifikovat možné tahy hráčů.

Další nemalou výhodou tohoto algoritmu je jeho rychlost. Algoritmus nevyžaduje téměř žádný výpočetní výkon. Časová náročnost je lineárně závislá na počtu možných tahů HP hráče. Označíme-li b počet možných tahů. Časová náročnost $O(b) = b$.

4.2.2. E-MaxMax

E-MaxMax je druhým jednoduchým algoritmem. Oproti E-HillClimberu již vyžaduje herní strom, možné tahy jednotlivých hráčů. Tento algoritmus simuluje jednoduché rozhodování hráčů a ohodnocuje jednotlivé tahy HP hráče až na základě stavů v předem dané hloubce herního stromu.

Stejně jako u E-HillClimberu a i následně u dalších algoritmů zde platí, že by se HP neměl rozhodovat vždy zcela deterministicky.

V případě deterministického rozhodování HP pro každý ze svých tahů odsimuluje část hry. Tah následně ohodnotí dle konečného stavu simulace. V jednotlivých krocích simulace každý z hráčů včetně HP maximalizuje svůj užitek pouze v rámci jednoho patra herního stromu. HP ve svých uzlech vybírá nejlepší tah stejně jako E-HillClimber. Ostatní hráči pracují s heuristickou funkcí, která jim ohodnotí vhodnost všech následujících tahů a také vyberou pro sebe nejlepší možný tah.

Nakonec HP vybere nejlépe ohodnocený tah.

Tento algoritmus je pomalejší než algoritmus předchozí, ale je stále velice rychlý. Horní odhad výpočetní složitosti je dán počtem ohodnocovaných uzlů herního stromu. Algoritmus v každém uzlu vybírá pouze jeden z následujících tahů. Označíme maximální větvící faktor stromu b a hloubku stromu d. $O(b, d) = b^2d$. Součin bd odpovídá časové náročnosti jedné simulace. Těchto simulací je maximálně b.

4.2.3. E-MaxN

Algoritmus E-MaxN patří k přístupu, který se snaží o přesnou simulaci hry po provedení svých tahů. HP z každého možného stavu, který následuje po jeho tahu, provede stejné rozhodování, které provádějí jednotliví hráči. Tato varianta předpokládá, že se soupeři rozhodují na základě algoritmu MaxN, ale nemělo by být obtížné upravit tento algoritmus pro soupeře založené na jiných algoritmech.

HP nezasahuje do průběhu simulace hry. Předpokládá, že se hráči rozhodují, jako by ve hře žádný HP nebyl, a tedy s uzly náhody pracuje stejně jako je obvyklé. Simulace hry se liší pouze v propagaci informací z listů ke kořeni herního stromu. Heuristika používaná hráči se propaguje jako obvykle. Ohodnocení stavu HP hráčem se přes uzly ostatních hráčů propaguje s heuristikou hráčů. Propagace se liší pouze v uzlech náhody. Hráči z nich propagují váženou sumu pravděpodobností tahů a jejich heuristik. HP ví, že dané uzly nejsou náhodné, že se v nich rozhoduje on. Z tohoto důvodu v uzlech náhody volí nejlepší ohodnocení z potomků.

Časová složitost algoritmu roste exponenciálně s hloubkou d procházeného stromu. $O(b, d) = b^{d+1}$. MaxN pro každý tah má časovou složitost b^d a MaxN se spouští až pro

b tahů.

4.2.4. E-MonteCarlo

Posledním navrženým algoritmem je E-MonteCarlo založeným na Monte Carlo prohledávání herního stromu(MCTS). Algoritmus se chová zcela stejně jako bylo popsáno dříve v této kapitole s jediným rozdílem.

Na konci fáze Simulace dochází k ohodnocení stavu. E-MonteCarlo stav ohodnotí pomocí metrik zábavnosti stejným způsobem jako předchozí algoritmy. Zbylé fáze MCTS se opět neliší.

Na závěr E-MonteCarlo vybere nejlépe ohodnoceného potomka aktuálního stavu hry.

Časová náročnost algoritmu je závislá na počtu iterací MCTS a neliší se od časové náročnosti MCTS.

5. Testující prostředí

V této kapitole popíši testující prostředí pro otestování navržených algoritmů hráčů prostředí a porovnání s dvěma existujícími přístupy.

Pro testování byly použity tři relativně jednoduché hry popsané dále. Jedná se o variantu bludiště, variaci Člověče, nezlob se s názvem Ludo a o karetní hru Ztracená města.

Prostředí obsahuje dva módy. Herní mód a dávkové spouštění algoritmů. V herním módu může jednotlivé hry hrát člověk. Hry mají jednoduché grafické ztvárnění a lze je ovládat pomocí myši. Po skončení hry si hráč může zobrazit Flow diagram.

Mód dávkového spouštění umožňuje provádění experimentů v dávkách. U každého experimentu lze nastavit druh hry, počet iterací experimentu, druhy jednotlivých hráčů včetně hráče prostředí a případně herně specifické parametry jako je velikost bludiště. Před spuštěním experimentů může uživatel ovlivnit počet vláken, v kterých se budou provádět experimenty. Vícevláknové spouštění je zde z důvodu maximálního využití vícejádrových procesorů. Po spuštění dávky se spouští postupně jednotlivé experimenty. Do paměti se ukládají informace o každé iteraci všech experimentů. Především jsou zaznamenávány metriky zábavnosti.

Po skončení dávky může uživatel napočítané informace uložit na disk. Pro rychlé zhodnocení proběhlých algoritmů se v aplikaci zobrazují agregované informace. Uživatel má na výběr mezi střední hodnotou, odchylkou, minimem a maximem z naměřených hodnot v rámci každého experimentu. Dále lze zobrazit histogramy pro jednotlivé hodnoty.

Bližší popis prostředí a jeho ovládaní lze nalézt v příloze.

5.1. Použité hry

Pro testování navržených algoritmů v předchozí kapitole jsem si zvolil následující tři relativně jednoduché hry - Bludiště, Ludo a Ztracená města. Hry se liší počtem hráčů, úplností informací a determinismus. Zařazení do jednotlivých kategorií znázorňuje následující tabulka 4.

Schéma podsekcí jednotlivých her je stejné. Nejdříve popíši cíl a pravidla hry. Následuje popis hráče prostředí (HP) v dané hře. Nakonec popisuji funkci hodnosti, status funkci a výpočet uvěřitelnosti, které jsou využívány metrikami pro měření zábavnosti. Pojmy funkcí hodnosti a statusu vychází z algoritmu Dynamická úroveň (3.6).

5.1.1. Bludiště

Bludiště je jednoduchou hrou pro jednoho hráče. Teroristа v bludišti umístil několik bomb. Cílem hráče je tyto bomby včas najít a zneškodnit. Hráč má k dispozici plánek s umístěním všech bomb, ale bohužel jsou v plánku vyznačeny i neexistující bomby. Hráč musí zkontrolovat všechny místa označená na mapě. Buď zjistí, že dané místo není dostupné, a tudíž se tam bomba nenachází, nebo místo dostupné je a bombu musí zneškodnit. (stačí dojít na její místo)

Herní plán bludiště se skládá z 2D čtvercové sítě. Jednotlivé čtverce mapy představují zdi, dveře, bomby, nebo průchozí pole. Hráč vidí bludiště z ptačí perspektivy a bludiště objevuje postupně. Vždy po otevření dveří se mu zobrazí chodba za nimi. Všechny bomby vybuchnou ve stejný čas. Čas je zde měřen na počet kroků hráče.

Teroristа navíc na některé dveře připevnil senzory, a poté dveře přemaloval namodro, nebo červeně. Jestliže hráč otevře modré dveře, získá čas navíc. Otevře-li červené, čas se mu zkrátí.

Hráč prostředí

Člověk si má myslet, že při hraní objevuje již dříve vygenerované bludiště. Ve skutečnosti se bludiště tvoří postupně, jak ho hráč objevuje. Když hráč otevře nové dveře, vytvoří se mu nová chodba.

Druh dveří na každé možné konkrétní pozici je určen před začátkem hry. Z toho vyplývá, že hráč prostředí může barvu dveří ovlivňovat pouze nepřímo. Může ovlivnit umístění dveří, které následně určí jejich barvu.

Generování chodeb má jasně daná pravidla. Chodba může být libovolně dlouhá, i přes celý herní plán. Chodba může být zakončena zdí, nebo mohou na konci následovat další dveře. Z vygenerované chodby vedou vždy maximálně jedny dveře vlevo a jedny vpravo. Může se stát, že díky pozdějšímu napojení dvou chodem na sebe vznikne chodba s více dveřmi po levé, či pravé straně, ale nikdy nejsou na žádné straně vygenerovány dvě dveře naráz. Toto omezení je vynahrazeno možností ukončit chodbu dveřmi a ne zdí. Více dveří na jedné straně lze tak vygenerovat postupně pomocí dvou chodeb vytvořených ve stejném směru.

I bez volby druhů dveří zbývá příliš mnoho možných tahů. Na mapě 40x40 polí může být délka generované chodby maximálně 39 polí dlouhá. Počet kombinací umístění dveří vlevo nebo vpravo se zakončením dveřmi, nebo zdí je $2 * 40^2$ (pozice 1-39, plus možnost

Tabulka 4. Klasifikace her do různých tříd dle počtu hráčů, determinismu a úplnosti informace.

Název hry	Počet hráčů	Determinismus	Informace
Bludiště	1	Bez náhody (z pohledu hráče)	Neúplná
Ludo	4	S náhodou	Úplná
Ztracená města	2	S náhodou	Neúplná

nedat žádné dveře). Větvící faktor větší než 3200 by nebyl příliš použitelný pro herní algoritmy.

Pro snížení větvícího faktoru nahradíme vždy několik tahů jedním. Provedeme abstrakci. HP neurčuje přesnou pozici dveří, ale jen přibližnou. Chodba je rozdělena do několika úseků a HP hráč si volí pouze, ve kterém úseku chce dveře. Konkrétní místo se vybere náhodně v rámci úseku. Dále HP nevybírá, jestli chodbu zakončí zdí, nebo dveřmi. Opět náhodná volba. Tímto byl větvící faktor HP hráče u bludišť o velikosti 40x40 snížen na maximální hodnotu 26.

Heuristika a uvěřitelnost

Heuristika pro hodnost se spočítá jako manhattonská vzdálenost mezi hráčem a nejbližší bombou. Výsledná hodnost je dána rozdílem počtu kroků do konce hry a desetinásobkem spočítané vzdálenosti.

Pro jednoduchost se hráč ve hře pohybuje skokem mezi jednotlivými dveřmi a vždy se změří reálná vzdálenost takového pohybu. Z tohoto důvodu ve všech stavech mimo prvního hráč stojí na pozici nově otevřených dveří, nebo čerstvě odstraněné bomby. Z tohoto důvodu bonus/penalizace za otevření různých druhů dveří je už v heuristice započítán v proměnné počet kroků do konce hry.

Status funkce se v této hře výrazně liší od funkce hodnosti. Je složena se součtu vzdáleností k jednotlivým aktivním bombám($sumDist$) a počtu ještě neprozkoumaných čtverců ($undefinedTiles$) a samozřejmě počtu kroků do konce hry ($stepsToGameOver$). Výsledný vzorec : $status = stepsToGameOver - \frac{3}{2}sumDist - (undefinedTiles/2 * coef)$. Koeficient coef je roven nule, jestliže hra skončila a hráč vyhrál, jinak je roven 1.

Uvěřitelnost v této hře je definována na základě délek nově vytvářených chodeb. Cílem bude tvořit bludiště bez příliš dlouhých chodeb a s přibližně stejným zastoupením různých délek. V našem konkrétním případě se zaznamená počet chodeb do délky 2, délek 2/3, 4/5, 6/7, 8/9 a zvlášť počet chodem o délce 10 a delších. Výsledná uvěřitelnost hry je dána součtem tisícinásobku počtu chodeb delších než 9 a uvěřitelnosti četností pole prvních 5 délek.

5.1.2. Ludo

Ludo patří mezi zástupce stíhacích her. U nás je nejznámější zástupcem stíhacích her hra Člověče, nezlob se. Nejdřív uvedu společného rysy obou her. Znalci Člověče, nezlob se mohou zbytek odstavce přeskočit. Každý hráč má k dispozici 4 figury a jeho cílem je dovést všechny jeho figury z počáteční pozice do cíle. Herní plán se skládá ze 4 startovních a koncových pozic za každého hráče v jeho barvě a z 40 sdílených pozic uspořádaných do kružnice. Hráči se střídají ve svých tazích. Jeden tah se skládá z hodu kostkou a posunu jedné z figur hráče o počet pozic vpřed dle hodnoty na hozené kostce. Na žádném z polí nemohou být dvě figury na stejné pozici. Pokud by se tak mělo stát, figura nehrajícího hráče stojícího na stejné pozici jako nově posunutá figurka

aktuálního hráče, je přesunuta zpět na startovní pozici. Ze startovní pozice na hlavní herní plán hráč může přesunout figuru, jestliže hodí na kostce šestku. Tuto šestku už nemůže použít pro pohyb jiné nebo stejné figury. Jestliže nemá hráč ani jednu z figur na hlavním plánu, může házet až třikrát, dokud nehodí šestku. Vítězí hráč, který jako první přesune všechny své figury do cílových pozic.

Hra Ludo do hry přidává bezpečné pozice. Některé pozice na herním plánu jsou odlišeny od ostatních. Jestliže na ní hráč má figuru, nemůže být odstraněna oponentem. Pokud oponent hodí kostkou přesně hodnotu, která by ho posunula na obsazenou bezpečnou pozici, tah nemůže provést. Další změnou je neexistence bonusových hodů po hození hodnoty 6, a to ani v případě, že pomocí 6 hráč figuru nově nasadil na herní plán.

V obou zmíněných variantách hráč má na začátku hry všechny figury na startovních pozicích v tzv. startovním domečku. Pro urychlení hry, ale i odstranění počáteční frustrace hráčů s menším štěstím, v implementované variantě hry všechny figury všech hráčů začínají na hlavním plánu.

Hráč prostředí

Hráč prostředí zde ovládá hrací kostku. Vzhledem k pravidlům hry je vliv HP hráče velice vysoký. Teoreticky je pouze na něm, který z ostatních hráčů vyhraje. Např. nemusí dovolit žádnému z hráčů hodit na kostce poslední potřebnou hodnotu pro umístění figury do cílové pozice. Z tohoto důvodu je zde zásadní metrikou uvěřitelnost.

Použitá heuristika a uvěřitelnost

Heuristiky pro hodnotící a status funkci jsem použil totožné. Heuristika pro jednotlivé hráče vyjadřuje přibližně, kolik průměrně kol hráč potřebuje k dokončení hry. Hodnota se rovná součtu počtu kol potřebných k dosažení cíle všech figur hráče.

Výpočet pro jednu figuru je následující. Průměrný hod má hodnotu 3,5. Počet kol k dosažení cíle je roven počtu polí před figurkou děleno 3,5. Pro nasazení figury je potřeba hodit 6. Abychom měli alespoň 50% hození 6, musíme kostkou hodit 4 krát. $(1 - \frac{5}{6}^4)$ Uvažujme případ, kdy hráč nemá žádnou figuru na hracím plánu. V takovém případě hráč hází 3 krát, a tedy potřebuje $\frac{4}{3}$ tahů, které se přičtou k heuristice pro každou nenasazenou figuru. Figury před cílovým domečkem musí na kostce hodit hodnotu odpovídající volné pozici v cílovém domečku. Uvažujme pesimistický případ, kdy je vhodná pouze jedna hodnota. V takovém případě přičteme další 4 potřebné tahy k heuristice.

Jestliže figura nestojí na bezpečné pozici a zároveň za ní stojí do vzdálenosti 6 figura oponenta, přičteme k heuristice $\frac{1}{6}$ z potřebných tahů k dostání se do aktuální pozice figury.

Uvěřitelnost hry závisí na hodech kostkou. Každý z hráčů očekává, že během hry každou hodí každou hodnotu kostky přibližně stejněkrát. Zároveň je nepravděpodobné,

že by hráč po sobě hodil 3 a vícekrát stejnou hodnotu. Uvěřitelnost hráče se skládá ze dvou částí. Je součtem uvěřitelnosti četnosti hodů kostkou a penalizací za hod stejné hodnoty 3 a vícekrát. Výsledná uvěřitelnost je rovna sumě uvěřitelností pro jednotlivé hráče.

5.1.3. Ztracená města

Ztracená města patří mezi jednoduché karetní hry od společnosti Albi. Herní balíček se skládá ze 60 karet 5 barev představujících jednotlivé expedice. Každá expedice může být složena až ze 12 karet. První 3 karty jsou sázkové, zbylé mají hodnotu 2 až 10. Cílem hry je vytvářet co nejdelší a nejhodnotnější expedice a získat větší bodový zisk než soupeř. Na začátku hry si každý z hráčů lízne 8 karet ze zamíchaného balíčku. Hráči se postupně střídají v kolech. Každé kolo se skládá ze dvou částí - zahrání karty a dolíznutí karty. Při zahrání karty má hráč na výběr v přiložení karty do jeho existující expedice, nebo odložení karty na vrchol společného odkládacího balíčku příslušné barvy. Při přiložení karty do expedice musí být splněno pravidlo, že nově přidaná karta má vyšší hodnotu než je nejvyšší hodnota v dané expedici. Sázkové karty hráč může přidat do expedice pouze v případě, že v ní nemá již kartu nesázkovou. V druhé fázi si hráč dobere kartu do celkového počtu 8 karet. Má na výběr mezi zakrytým dobíracím balíčkem, nebo si může vzít kartu z vrchu jednoho odkládacího balíčku.

Bodové hodnocení expedic hráče se spočítá následovně. U každé expedice se sečtou hodnoty běžných karet. Z expedice se odečtou náklady v celkové hodnotě 20. V případě, že hráč má vyloženu 1, 2, nebo všechny tři sázkové karty, výslednou hodnotu vynásobí 2, 3, nebo 4. Expedice může mít ve výsledku zápornou hodnotu. Např. má-li hráč v expedici jednu sázkovou kartu a karty s hodnotami 4 a 5, bodové hodnocení se spočítá $(4 + 5 - 20) * 2$. Jestliže je expedice složena alespoň z 8 z celkových 12 karet, hráč si přičte dalších bonusových 20 bodů. (Tyto body se nenásobí)

Kompletní originální pravidla hry s vysvětlujícími obrázky lze nalézt na [**Neni**].

Hráč prostředí

Na začátku hry je náhodně rozdáno 8 karet pro každého hráče. Vždy, když se hráč rozhodne táhnout kartu z balíčku, kartu vybere hráč prostředí.

Použitá heuristika a uvěřitelnost

Podobně jako u hry Ludo, i zde je využívána stejná heuristika pro funkce status a hodnosti. Heuristika je vždy počítána jako by se jednalo o hru s úplnou informací. Můžeme ji počítat tímto způsobem, protože umělá inteligence nikdy nevolá heuristiku přímo na aktuální stav, ale vždy na stav ze stejného informačního setu, který stochasticky vygeneruje.

Heuristika odhaduje předpokládaný počet bodů získaných z jednotlivých expedic. Počítáme body pouze z expedic, které již byly založeny. Pro každou založenou expedici

spočteme aktuální počet bodů a výsledek vynásobíme konstantou 10. Poté projdeme všechny karty v ruce, v balíčku a vrchní kartu z odložených karet, které ještě hráč může zahrát. (mají vyšší hodnotu než nejvyšší vyložená karta) Součet bodů těchto karet vynásobíme konstantami 8, 4, nebo 3 a 7 pro karty z ruky, balíčku a vrchní kartu. Pro karty z balíčku se vybere konstanta 2, jestliže hráč založí 4. expedici, jinak je 3. Heuristika je převzata z [39] včetně koeficientů.

Aby hra byla *uvěřitená*, jednotliví hráči musí během hry získat podobné počty barev a hodnot karet. Uvěřitelnost barvy je méně důležitá. Naopak pokud se pokusíme, aby každý hráč dostal od každé barvy během hry stejný počet karet, můžete to negativně ovlivnit průběh hry. Hodnoty karet rozdělíme do 4 kategorií - sázkové karty, hodnoty 2, 3, 4, hodnoty 5, 6, 7 a nakonec 8, 9, 10. Hra bude více uvěřitelná, když každý hráč během hry získá podobně karet z každé kategorie.

Výsledná uvěřitelnost je sumou uvěřitelností četnosti kategorií hodnot a poloviny uvěřitelnosti četností barev pro každého hráče zvlášť.

5.2. Použité umělé inteligence

V aplikaci jsou implementovány dva druhy hráčů. První druh představuje hráče prostředí, kteří se starají o náhodný prvek ve hře. Druhým typem jsou běžní hráči, kteří se chovají relativně rozumně.

Pro všechny tři hry jsem připravil hráče založené na DDA algoritmech Dynamická úroveň a POSM uvedených ve třetí kapitole. Tito hráči slouží pro porovnání mého nového přístupu s již existujícími.

5.2.1. Hráči prostředí

Prvním hráčem prostředí je hráč hrající náhodně. S tímto hráčem se všechny hry chovají zcela přirozeně jako by v nich tento typ hráče vůbec nebyl. Tento hráč byl přidán pro srovnání naměřených metrik u her, které neovlivňuje hráč prostředí, a které naopak ovlivňuje.

Dále je v aplikaci implementována čtveřice hráčů prostředí navržených v předcházející kapitole.

Hra Bludiště disponuje dvěma hráči navíc. Obsahuje hráče založené na algoritmech POSM a Dynamická úroveň. Tito hráči dělají ze hry Bludiště hru se dvěma soupeřícími hráči. Klasický hráč má za úkol splnit úkol, nalézt všechny bomby a jeho soupeř se mu v tom snaží zabránit.

5.2.2. Běžní hráči

V testovacím prostředí existuje několik hráčů postavených na různých algoritmech. Aplikace byla navržena tak, aby každý hráč byl použitelný pro každou hru. Některé

kombinace her a hráčů nebyly příliš rozumné, a proto v aplikaci jsou povoleny pouze některé.

Ve všech hrách lze využít hráče hrajícího náhodně. Slouží pouze jako jeden z benchmarků pro ostatní algoritmy. Pomohl např. objevit chybu v dřívější heuristice pro Člověče, nezlob se. Náhodný hráč vyhrával častěji než hráč využívající heuristiku.

U her Bludiště a Ludo je aktivní další jednoduchý hráč nazvaný HillClimber. Ke své funkčnosti potřebuje funkci, jež dokáže získat všechny následující tahy a funkci, která je ohodnotí hodností. Ohodnocené tahy seřadí dle hodnosti a vybere nejlepší z nich.

Pro hru Ztracená města je určená varianta s IS (informační set). HillClimber IS před rozhodnutím provede několik desítek iterací. Na začátku iterace vygeneruje náhodně stav hry ze stejného informačního setu jako je aktuální stav. Nad tímto stavem provede základní algoritmus Hill Climber, ohodnotí všechny tahy z něj. Pro jednotlivé tahy si ukládáme sumu hodností přes všechny iterace. Můžeme si to dovolit, protože ve hře Ztracená města nejsou možné tahy hráče v jednom kole závislé na neúplné informaci, na soupeřově kartách v ruce. Výsledná sumární ohodnocení pro tahy seřadíme a opět vybereme nejlepší z nich.

Bludiště žádné jiné hráči nevyužívá, jelikož hráči nemají být seznámeni se způsobem tvorby bludiště.

Ve hře Ludo nalezneme hráče založené na Monte Carlo prohledávání stromu a na vícehráčové variantě algoritmu Minimax, na algoritmu Max N. Oba algoritmy byly stručně popsány na začátku předchozí kapitoly.

Pro hru Ztracená města jsem se snažil vytvořit hráče, který nebude podvádět (nebude znát karty v soupeřově ruce a ani skryté v balíčku) a zároveň bude lepší než jednoduchý algoritmus HillClimberIS. Nejdříve jsem se pokoušel o různé varianty hráče založené na Monte Carlo prohledávání stromu. Bohužel všechny varianty hráče se chovali v průměru hůře než jednoduchý HillClimber IS. Z tohoto důvodu jsem se rozhodl pro implementaci hráče navrženého Pedrem Vasconcelosem [39] pro stejnou hru a nazval jsem ho Minimax IS.

Hry Ludo a Ztracená města navíc obsahují dva hráče implementující DDA mechanismus. Jsou to hráči Dynamická úroveň a POSM pojmenovaní dle algoritmů popsaných ve třetí kapitole.

5.2.3. Minimax IS

Algoritmus Minimax IS se od klasického Minimaxu liší obdobně jako HillClimber IS od HillClimberu.

Algoritmus v iteracích provádí dva kroky. Nejdříve vytvoří na základě aktuálního stavu stav ze stejné informační množiny. V našem případě se jedná o zamíchání neznámých soupeřových karet s balíčkem a z takto zamíchaných karet se doplní soupeřova ruka. Nad novým stavem se zavolá Minimax s prořezáváním alfa, beta. Zapamatuje se nejlepších tah dle Minimaxu a pokračuje se další iterací. Nakonec hráč zahraje tah, který byl nejvíce krát vybrán Minimaxem.

Vasconcelos herní strom prořezává do hloubky i do šířky. Prořezávání do hloubky je dle očekávání. V určité hloubce stromu se stav ohodnotí dle heuristické funkce. Při prořezávání do šířky se z každého uzlu ohodnotí všechny následující tahy dle stejné heuristické funkce, Minimax prochází pouze tři nejlepší z nich.

Hloubku stromu a počet iterací mění v průběhu hry. Čím méně karet zbývá v balíčku, tím provádí méně iterací, ale naopak zvedá maximální hloubku stromu.

Oproti originální verzi algoritmu jsem provedl několik změn. Každou změnu jsem vždy otestoval vůči hráčům HillClimber IS i proti HillClimber, který pracuje s úplnou informací. Ukázalo se, že šířka stromu 3 hned od první hloubky je příliš málo. Z tohoto důvodu v hloubce 1 ořezávám strom na šířku 10 a až poté ořezávám na šířku 3. Dále jsem upravil poměr počtu iterací a hloubky stromu. Čas získaný snížením maximální hloubky stromu šlo využít pro zvětšení počtu iterací. Tato změna také přispěla k lepšímu chování.

Poslední, ale ne nedůležitou změnou, je úprava generování stavu ze stejné informační množiny. Každý hráč si pro všechny karty, které doposud neviděl, udržuje hodnotu pravděpodobnosti, s jakou je daná karta v soupeřově ruce. Pravděpodobnosti se upravují v průběhu hry na základě soupeřových tahů dle několika pravidel získaných od znalce hry. Příkladem je pravidlo, které se použije, pokud soupeř vyloží novou expedici. V takovém případě se zvedne pravděpodobnost pro karty stejné barvy a vyšší hodnoty než je vyložená karta. Naopak lze předpokládat, že soupeř nevlastní žádnou kartu nižší hodnoty stejné barvy, protože tah vyložení takové karty by dominoval provedenému tahu. Stav ze stejné informační množiny se generuje na základě těchto pravděpodobností a ne dle uniformního rozdělení pravděpodobnosti jako tomu bylo v původním algoritmu.

5.2.4. Škálování

Všechny uvedené algoritmy umožňují škálování umělé inteligence. Škálování potřebuje pro simulaci různě silných hráčů. Vedle algoritmu můžeme nastavit parametr úrovně mezi 1 a 100.

Každý z algoritmů pro běžné hráče je upraven tak, aby nevracel pouze nejlepší možný tah, ale aby vrátil ohodnocené všechny tahy. Ohodnocené tahy se seřadí vzestupně. Ze seřazených tahů se nevybírá nejlepší tah, ale nejpravděpodobněji se vybere ten, který se nachází v $\frac{úroveň}{100}$. Vybere se tah dle gaussovského rozdělení pravděpodobnosti se střední hodnotou v $\frac{úroveň}{100}$ a odchylkou $0,4$. Kdybychom vybírali tah přesně v daném zlomku, ztratili bychom výraznou část tohoto škálování. Např. u hry Člověče, nezlob se, kde má hráč v jednu chvíli na výběr maximálně 4 tahy, by se hráči s úrovněmi 76-100 chovali zcela stejně.

6. Provedené experimenty

V této kapitole popíši provedené experimenty a ukáži naměřené výsledky, které následně zhodnotím.

Nejdříve budu hodnotit algoritmy zvlášť v rámci jednotlivých her. V poslední sekci kapitoly zhodnotím testované DDA algoritmy celkově.

Každý z experimentů obsahoval celkem 1000 iterací.

Jednotlivé metriky mají diametrálně odlišné rozsahy hodnot. Z tohoto důvodu v grafech zobrazuji místo konkrétních hodnot poměr k nejlepší hodnotě. Výběr nejlepší hodnoty závisí na druhu metriky. Změnu vedení a svobodu se snažíme maximalizovat, ostatní metriky minimalizujeme. Z tohoto důvodu u změny vedení a svobody zobrazuji poměr vůči maximu, u zbylých metrik je tomu naopak.

Pro jednodušší vizuální porovnávání algoritmů mezi sebou upravuji metriky, které se minimalizují. Pracuji s převrácenou hodnotou těchto metrik. Po této úpravě zobrazuji do grafů poměr k maximální převrácené hodnotě minimalizujících metrik. Z tohoto důvodu čím vyšší hodnoty pro každou z metrik, tím je algoritmus lepší.

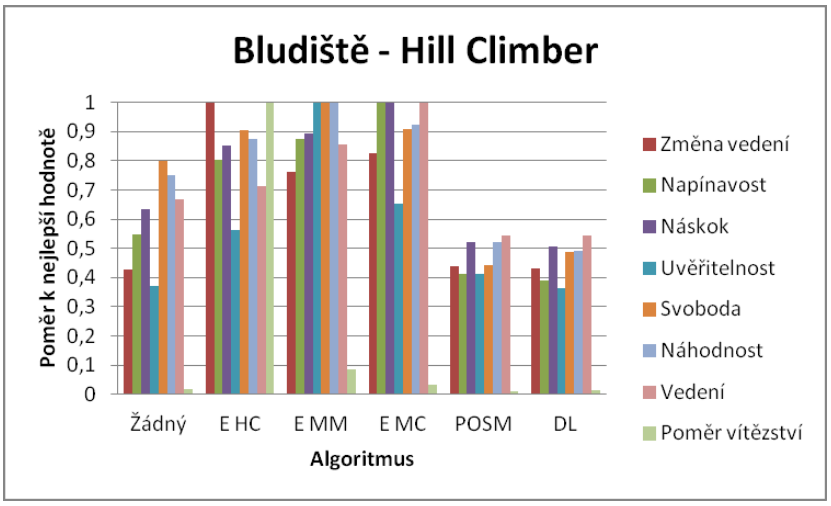
6.1. Bludiště

Pro testování bludiště jsem nastavil jeho velikost na šířku a výšku 41 čtverců, maximální počet kroků 777. Koeficienty metrik byly nastaveny na 10 pro uvěřitelnost, 100 pro napětí, 500 pro náskok a 100 pro svobodu. Zbylé koeficienty byly nastaveny na 0. Výsledek si lze prohlédnout na následujícím grafu (14) a tabulce(5).

V grafu jsou použity zkratky pro použité algoritmy. E-HC představuje E-HillClimber, E-MM je pro E-MaxMax, E-MC pro E-MonteCarlo, POSM pro Částečně uspořádaná množina – Mistr a DL pro Dynamickou úroveň. U Bludiště nebyl použit E-MN pro E-MaxN. Algoritmus E-MaxN má za úkol simulovat hráče. Hráč ve hře bludiště se rozhoduje pouze dle následujících stavů. Z tohoto důvodu zde je postačující Algoritmus E-MaxMax, který plní přesně stejnou funkci jako E-MaxN u dalších her. Algoritmus "Žádný"představuje naměřené metriky z hry, kdy nebyl použit žádný DDA mechanismus. Slouží pro srovnání s DDA algoritmy.

Z tabulky a především z grafu lze vyčíst, že algoritmy POSM a DL zaostávají nejen za ostatními DDA algoritmy, ale také za hrou, kde není použit žádný DDA mechanismus. Dopadají hůře i v metrikách napínavost a náskok, na kterých je testovali jejich autoři.

Při bližším zkoumání jsem zjistil, že důvodem pro špatné chování je absence přizpůsobování se metrice svoboda. Oba algoritmy příliš ořezávají možné tahy hráče na



Obrázek 14. Srovnání DDA alg. ve hře Bludiště s Hill Climber hráčem.

průměrných 7 tahů během hry. Bez použití DDA je průměrná hodnota svobody 12. Při nízké svobodě má hráč ve hře malé množství dveří, kterými by se vydal. Ve hře existuje málo větvení a naopak dlouhé chodby. Z tohoto důvodu se hráči rychle snižuje neprobádaný prostor - dlouhé chodby bez postranních dveří často odříznou část mapy od hráče.

Oba algoritmy pracují pouze s hloubkou 1 a často se dostanou do situace, kdy už musí vytvořit chodbu k poslední bombě, přestože hráči zbývá ještě dostatek času. Případně nastane zcela opačná situace. Na mapě je více jak jedna bomba, tedy tyto algoritmy mohou bombu, ke které směřuje hráč, odříznout a udělat z ní falešnou, ale jelikož jsou ve hře především dlouhé nevětvené chodby, hráč se musí vrátit o velké množství kroků a vyprší mu čas.

Algoritmy E-HC, E-MM a E-MC dopadly obdobně. Algoritmu E-HC se nejlépe dařilo

Tabulka 5. Porovnání metrik zábavnosti u jednotlivých algoritmů ve hře Bludiště. Metriky změna vedení a svoboda se maximalizují, zbytek minimalizuje.

Algoritmus	Počet kol	Změna vedení	Napínavost	Náskok	Uvěřitelnost
Žádný	93,127	8,217	343,6848421	239,851	81789,7937
E HC	114,905	19,176	233,9169735	177,951	53868,86
E MM	113,32	14,64	214,5392914	169,935	30377,59274
E MC	105,494	15,829	187,5847573	151,923	46656,61097
POSM	99,902	8,394	455,7661817	290,602	73884,68107
DL	95,431	8,29	483,7274291	300,486	83988,2358

Algoritmus	Svoboda	Náhodnost	Vedení	Rozptyl Vítězů
Žádný	12,19857537	15,05552294	33,04804979	0,121
E HC	13,79978394	12,92183276	31,00734155	0,002
E MM	15,26174269	11,29628595	25,74151376	0,024
E MC	13,84465841	12,22155687	22,05607333	0,065
POSM	6,76593984	21,72255666	40,54974306	0,176
DL	7,45030571	22,91638791	40,45994322	0,154

Obrázek 15. Srovnání bludišť vytvořených různými DDA algoritmy. Zleva-doprava, shora-dolů: žádný, E-MaxMax, POSM, DL

v metrikách poměr vítězství a změna vedení, E-MM zvítězil v uvěřitelnosti, náhodnosti a svobodě, E-MC dopadl nejlépe v napínavosti, náskoku a ve vedení. Celkově nejlépe si vedl v této hře algoritmus E-MaxMax.

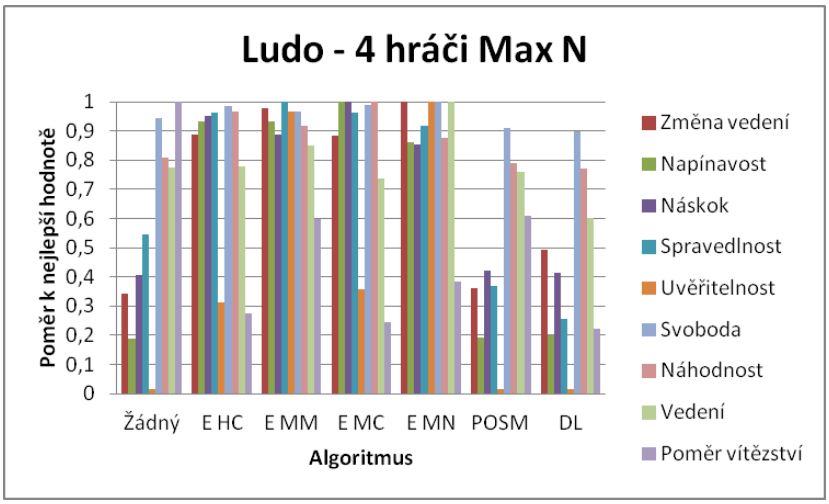
6.1.1. Vzhled bludiště pro různé DDA algoritmy

Druh použitého DDA algoritmu má velký vliv na výsledný vzhled bludiště. Typické zástupce vytvořených bludišť ukazuje obr. 15. Pro bludiště vytvořené náhodně jsou typické zpočátku dlouhé chodby a posléze vyplňování chodbiček mezi dlouhými chodbami. Z vlastních algoritmů jsem vybral zástupce E-MaxMax, jehož bludiště působí nejpřirozeněji.

Bludiště vytvořená pomocí POSM a DL vypadají obdobně. Je pro ně typický výrazný počáteční kříž chodeb, kde algoritmy brzy uzavřou tři možné cesty a nechají mu jednu polední.

6.2. Ludo

Oproti hře Bludiště je ve hře Ludo zásadní metrikou uvěřitelnost. V případě, že na tuto metriku nedáme důraz, HP nedovolí žádnému z hráčů hru vyhrát. Z jeho pohledu je ideální, když všichni hráči mají 3 figury v cíli a 4 těsně před cílem. Tato situace se mu



Obrázek 16. Srovnání DDA alg. ve hře Ludo se 4 Max N hráči. (úrovně 50, 100, 100, 100)

zdá napínavá a ona by i ve skutečné hře byla, ale ne v případě použití falešné kostky. Z tohoto důvodu jsem nastavil koeficienty u uvěřitelnosti na 1000, spravedlnost, vedení, napětí na 10, náskok na 3 a svobodu na 1.

S tímto nastavením jsem provedl 3 experimenty. V prvních dvou hráli pouze dva hráči, jednou s algoritmem Hill Climber, podruhé s Max N. Experiment s dvěma hráči jsem spouštěl, aby měli algoritmy Dynamický level a POSM prostředí, pro které byly navrhnuty - dvouhráčové hry. V třetí experimentu byly 4 hráči Max N. První měl úroveň 50, další tři 100. Pro test DL a POSM jsem nahradil všechny 3 poslední hráče právě algoritmy DL, nebo POSM.

Výsledky všech 3 experimentů byly obdobné. Zde uvedu pouze graf(16) a tabulku(6) posledního experimentu. Grafy zbylých dvou experimentů lze nalézt v příloze.

Algoritmy POSM a DL měli zde již svou tradiční úlohu - ovládali běžné hráče. Z tohoto důvodu neovlivňovaly metriky uvěřitelnost, svoboda, náhodnost a spravedlnost, které musely dopadnout podobně jako při nepoužití žádného DDA algoritmu. Oba zlepšily metriku změny vedení, ale zhoršily poměr vítězství. Náskok i napínavost měly podobnou, jako by se žádné DDA nepoužilo.

U této hry a ještě více u Ztracených měst se projevil zásadní problém těchto DDA mechanismů. Pokud hráči jimi ovládaní jsou ve vedení, zhoršují svoje chování. Zhoršují ho až do zcela iracionální podoby. U hry Ludo se iracionální chování projeví např. v situaci, kdy hráč má jednu figurku před cílem a na kostce padla správná hodnota, aby hráč mohl přesunout svou figurku do cíle. V případě, že je hráč již dlouho ve vedení, tak raději nechá v figurku v potencionálně nebezpečné pozici a pohne figurkou jinou.

Algoritmy E-HC, E-MM, E-MC a E-MN velmi předčily zbytek algoritmů i v metrice uvěřitelnost. V případě náhodného chování kostky se může s malou pravděpodobností stát, že padne jednomu hráči 6 třikrát po sobě. Je to relativně málo pravděpodobné, ale stát se to může. Algoritmy HP se snaží tyto málo pravděpodobné jevy eliminovat, a proto si dle metriky uvěřitelnost vedly lépe.

Celkově si nejlépe vedly E-MM a E-MN, které dokáží celkem věrně simulovat chování hráčů a následně se rozhodovat dle stavů hry dále v budoucnu. Mohou lépe naplánovat chování kostky z hlediska uvěřitelnosti a díky tomu se zaměřit více na ostatní metriky.

6.3. Ztracená města

I u Ztracených měst byla nejdůležitější metrikou uvěřitelnost. Bez ní prohrávající hráč získával postupně karty hodnot 10, 9, 8 atd. barev již vyložených expedic, dokud se nedostal do vedení. Nastavení koeficientů metrik jsem zvolil 1000 pro uvěřitelnost, 100 pro změnu vedení, spravedlnost, vedení a napětí, 10 pro svobodu a 0 pro náskok.

Opět byly spuštěny dva experimenty s různými v hráči. V jednom hráli proti sobě HillClimber IS s úrovněmi 50 a 100 a v druhém MiniMax IS se stejnými úrovněmi. Výsledky byly podobné, a proto se zaměřím na experiment se složitějším hráči (graf 17, tabulka 7). Graf druhého experimentu lze nalézt v příloze.

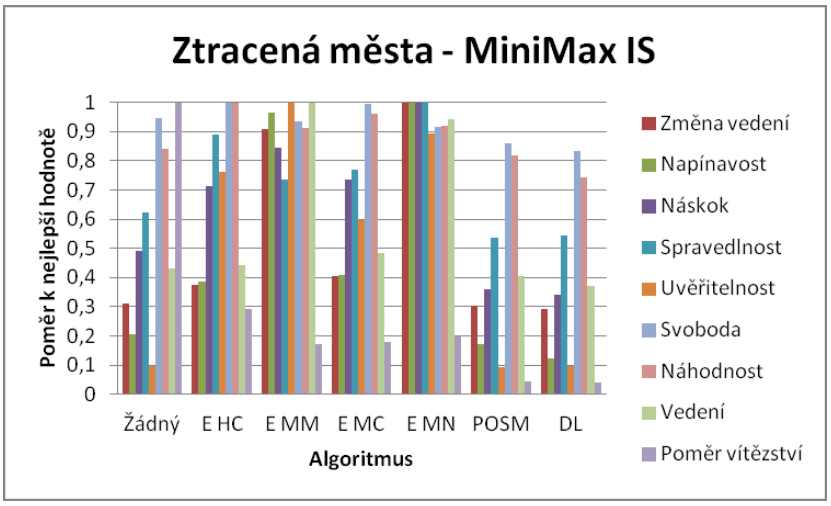
DL a POSM opět ovládaly jednoho z hráčů, a tedy nemohly ovlivnit čtveřici metrik spravedlnost, uvěřitelnost, svoboda a náhodnost. Oba dva algoritmy dopadly špatně z důvodu nastíněného u hry Ludo. Ve Ztracených městech člověk dokáže snadno rozlišit zcela iracionální tahy od těch použitelných. DL a POSM ale mají tyto iracionální tahy k dispozici. Jestliže je hráč ovládaný DL nebo POSM ve vedení, nejrychleji srovná skóre vyložením nové expedice, za kterou má záporné body. Bohužel takový tah může být zásadní a i kdyby posléze hrál daný hráč dokonale, nemusí mu to stačit k opětovnému dostání se do vedení. Z tohoto důvodu je u Ztracených měst lepší nepoužít žádný algoritmus než POSM a DL v nezměněné podobě.

E MC a E HC dopadly navzájem podobně, ale oba lépe než hra bez DDA. Jelikož E

Tabulka 6. Porovnání metrik zábavnosti u jednotlivých algoritmů ve hře Ludo. Metriky změna vedení a svoboda se maximalizují, zbytek minimalizuje.

Algoritmus	Počet kol	Změna vedení	Napínavost	Náskok	Spravedlnost
Žádný	321,629	33,207	10,96064284	10,69	0,053439508
E HC	367,71	86,401	2,191965172	4,576	0,030182817
E MM	374,831	95,074	2,197616324	4,911	0,029112477
E MC	370,011	86,126	2,047674062	4,351	0,030268374
E MN	349,386	97,275	2,378575689	5,096	0,031668999
POSM	329,262	35,024	10,57217622	10,292	0,079106286
DL	342,281	47,836	10,26501397	10,484	0,113375705

Algoritmus	Uvěřitelnost	Svoboda	Náhodnost	Vedení	Poměr vítězství
Žádný	24,57354378	1,86791164	1,189324764	67,3734372	0,021365861
E HC	1,226482879	1,95087896	0,996415593	67,020713	0,078176083
E MM	0,396586046	1,90887509	1,047952319	61,58128218	0,035686132
E MC	1,078624424	1,95431191	0,963174893	70,7910895	0,087272562
E MN	0,383798925	1,9757931	1,100121925	52,281059	0,055709066
POSM	25,21722299	1,79829072	1,216838264	68,67222926	0,035035696
DL	26,54645197	1,77747205	1,250394015	86,85364229	0,096187317



Obrázek 17. Srovnání DDA alg. ve hře Ztracená Města s dvěma MiniMax IS hráči. (úrovně 50 a 100)

HC je znatelně rychlejší, je v tomto případě vhodnější než E MC. Vítězem u Ztracených měst je jednoznačně E MN, který byl nejlepší v metrikách změna vedení, napínavost, náskok a spravedlnost.

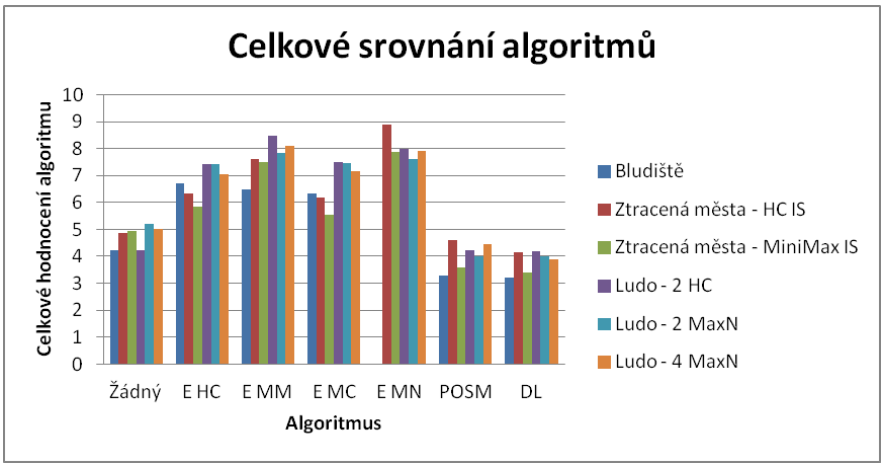
6.4. Celkové srovnání

Pro celkové srovnání použitých DDA algoritmů jsem každý algoritmus ohodnotil v jednotlivých hrách na základě všech 9 metrik a zvlášť na základě pouze 4 metrik změna vedení, napětí, náskok a poměr vítězství. Dvojí hodnocení jsem použil ze dvou důvodů. Algoritmy POSM a DL nebyly navrženy pro testování na daných metrikách a za druhé

Tabulka 7. Porovnání metrik zábavnosti u jednotlivých algoritmů ve hře Ztracená města. Metriky změna vedení a svoboda se maximalizují, zbytek minimalizuje.

Algoritmus	Počet kol	Změna vedení	Napínavost	Náskok	Spravedlnost
Žádný	51,21	5,17	169,3785106	159,11	2,97949985
E HC	51,652	6,219	90,11489792	109,66	2,083795471
E MM	51,058	15,105	35,89171719	92,52	2,51490921
E MC	51,076	6,763	84,52874032	106,63	2,407272356
E MN	51,284	16,647	34,66283799	78,3	1,853999951
POSM	53,785	5,038	204,2092341	217,97	3,458477317
DL	49,957	4,838	283,7280003	228,54	3,401727139

Algoritmus	Uvěřitelnost	Svoboda	Náhodnost	Vedení	Poměr vítězství
Žádný	4,930532754	30,0635238	12,59686374	15,751	0,016
E HC	0,612039822	31,8330098	10,61479482	15,426	0,055
E MM	0,467313632	29,8150478	11,6301367	6,802	0,093
E MC	0,78408408	31,6226395	11,05890933	14,119	0,089
E MN	0,52395678	29,1162948	11,52272589	7,209	0,081
POSM	4,956223714	27,4061938	12,97565858	16,7575	0,358
DL	4,944157754	26,5193542	14,25249958	18,3885	0,408



Obrázek 18. Srovnání alg. DDA na základě všech experimentů a metrik.

jejich použití u her Ludo a Ztracená města neovlivňuje 4 ze zbylých metrik.

Hodnocení algoritmu ve hře se rovná součtu hodnocení za jednotlivé metriky. Algoritmus, který si vedl nejlépe v daném metrice, získal celý bod za tuto metriku. Ostatní algoritmy si přičetly poměr jejich hodnoty k nejlepší hodnotě. Z toho vyplývá, že u prvního hodnocení je maximální hodnota 9, u druhého 4.

První hodnocení je znázorněno na grafu 18 a v tabulce 8. Nejlépe dopadly algoritmy E MM a E MN. E MN získal ve Ztracených městech téměř maximální počet 9 bodů, získal 8,87. Za nimi následuje dvojice E HC a E MC se zisky kolem 6,5 bodů. DL s POSM dopadly nejhůře se zisky kolem 4 bodů.

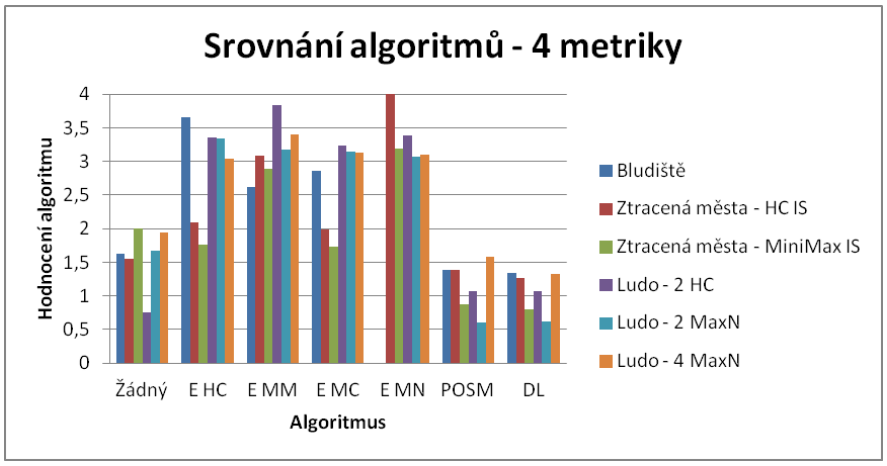
Zaměříme-li se pouze na 4 zmíněné metriky, dojde ke stejným závěrům. Zajímavostí je, že ve hře Ztracená města s dvěma HillClimber hráči algoritmus E MN dominoval ve všech 4 metrikách. DL a POSM i zde sbírají poslední místa.

Oba tyto algoritmy byly navrženy a testovány u her, kde jeden špatný tah zpravidla nerozhodne hru - testovanými hrami byl backgammon u DL a dáma s čínskými šachy u POSM.

Vzhledem výsledkům lze říci, že v případě dostatku výpočetního času a znalosti možných tahů hráčů, je nejvhodnější využít jeden z algoritmů E-MaxMax, nebo E-MaxN. Algoritmus založený na Monte Carlo prohledávání stromu dopadá obdobně jako E-

Tabulka 8. Celkové hodnocení algoritmů v 6 experimentech na základě 9 metrik.

Algoritmus	Bludiště	ZM-HC IS	ZM-MM IS	Ludo-2HC	Ludo-2MN	Ludo-4MN
Žádný	4,21	4,88	4,94	4,23	5,22	5,03
E HC	6,71	6,35	5,86	7,40	7,43	7,06
E MM	6,47	7,62	7,48	8,47	7,85	8,10
E MC	6,34	6,20	5,53	7,50	7,46	7,18
E MN	Neměřeno	8,87	7,87	8,00	7,61	7,89
POSM	3,30	4,59	3,59	4,23	4,00	4,43
DL	3,23	4,14	3,38	4,20	3,99	3,87



Obrázek 19. Srovnání alg. DDA na základě všech experimentů a metrik Změna vedení, Napětí, Náskok a Poměr vítězů.

HillClimber, ale má mnohem větší časovou náročnost a též potřebuje znalost možných tahů, a tedy se neukázal jako doporučitelný. E-HillClimber sice zaostává za první zmíněnou dvojicí, ale ne o moc. Průměrně dosáhl výsledku 6,8 bodů, E-MaxMax 7,7 a E-MaxN 8,1. Z důvodu jednoduché implementace algoritmu a dobrým výsledkům může být E-HillClimber velice vhodným kandidátem na požití pro techniky DDA ve velikém spektru aplikací. Algoritmy dynamická úroveň a POSM se neukázaly být vhodné pro použití u tohoto typu her.

Pro doplnění E-MonteCarlo dosáhlo průměrně 6,7 bodů, POSM 4,0 a DL 3,8. Bez DDA se dosáhlo 4,8 bodů.

Tabulka 9. Celkové hodnocení algoritmů v 6 experimentech na základě 4 metrik - změna vedení, napínavost, náskok a poměr vítězství.

Algoritmus	Bludiště	ZM-HC IS	ZM-MM IS	Ludo-2HC	Ludo-2MN	Ludo-4MN
Žádný	1,62	1,54	2,01	0,76	1,67	1,94
E HC	3,66	2,09	1,76	3,36	3,34	3,05
E MM	2,62	3,09	2,89	3,83	3,18	3,39
E MC	2,86	1,99	1,73	3,24	3,15	3,13
E MN	Neměřeno	4,00	3,20	3,38	3,07	3,10
POSM	1,38	1,39	0,88	1,07	0,60	1,59
DL	1,34	1,27	0,79	1,08	0,62	1,33

7. Závěr

Cílem diplomové práce bylo seznámit se s problematikou dynamického vyvažování obtížnosti v počítačových hrách především z hlediska algoritmů teorie her, zanalyzovat existující přístupy a metriky pro měření úspěšnosti jednotlivých přístupů, navrhnout nový přístup a metriky. Dalším cílem bylo navrhnout a implementovat testovací prostředí s třemi hrami a na nich nový přístup ohodnotit dle existujících i nových metrik.

Nalezené existující přístupy využívají velké množství metod umělé inteligence. Jsou zde zastoupeny neuronové sítě, POMDP, fuzzy logika, evoluční algoritmy a další. DDA v oblasti teorie her se ukázalo být zatím nedostatečně prozkoumáno, a proto by tato práce měla částečně zaplnit tuto mezeru.

Navrhl jsem algoritmy založené na hráči prostředí, který má za úkol přizpůsobovat náhodné události a skrytou informaci. Při používání DDA je jedním z hlavních rizik hráčovo zjištění, že je DDA ve hře použito. Z tohoto důvodu bylo zásadní navrhnout algoritmy tak, aby upravovaly náhodné události v uvěřitelné míře. Pro tento účel se ukázala čtveřice existujících metrik zábavnosti - poměr vítězství, změna vedení, náskok a napětí - nedostatečná. Z tohoto důvodu jsem postupně dodefinoval další pětici metrik - doba vedení, uvěřitelnost, spravedlnost, svoboda a náhodnost, které existující metriky vhodně doplnily. Algoritmy založené na hráči prostředí jsem nazval E-HillClimber, E-MaxMax, E-MaxN a E-MonteCarlo dle známých algoritmů z teorie her, z kterých vycházejí.

Pro ověření funkčnosti nových algoritmů jsem navrhl a implementoval testující prostředí s třemi hrami - Bludiště, Ludo a Ztracená města. Aplikace obsahuje režim pro hraní hry člověkem proti soupeřům a režim dávkového spouštění experimentů na více vláknech.

Každý z experimentů jsem nejdříve spustil bez jakéhokoli mechanismu DDA, poté se čtveřicí algoritmů hráče prostředí a nakonec s dvěma existujícími algoritmy dynamická úroveň a POSM pro porovnání. Výsledky experimentů jsou pozitivní. Hráč prostředí se ukázal jako vhodným konceptem, který předčil hru bez žádného DDA mechanismu i algoritmy dynamickou úroveň a POSM ve všech třech testovaných hrách.

V experimentech nejlépe dopadl E-MaxN, který v některých případech dokázal předčit všechny ostatní algoritmy dle většiny metrik. E-HillClimber dosáhl o něco slabších výsledků, ale stále oproti existujícím přístupům i proti nevyužití žádného DDA si vedl velmi dobře. E-HillClimber je jednoduchým algoritmem, který je díky svým malým požadavkům a velké rychlosti použitelný nejen v tahových hrách.

Do budoucna lze navázat na výstup této diplomové práce především otestováním al-

goritmů na lidech, důkladným zkoumáním vlivu změn koeficientů metrik u jednotlivých her, případně s návrhem jiné maximalizační funkce než je vážená suma metrik.

Diplomovou práci v této podobě považuji za kompletní. Věřím, že plně splnila zadání.

Příloha A.

Ovládání aplikace

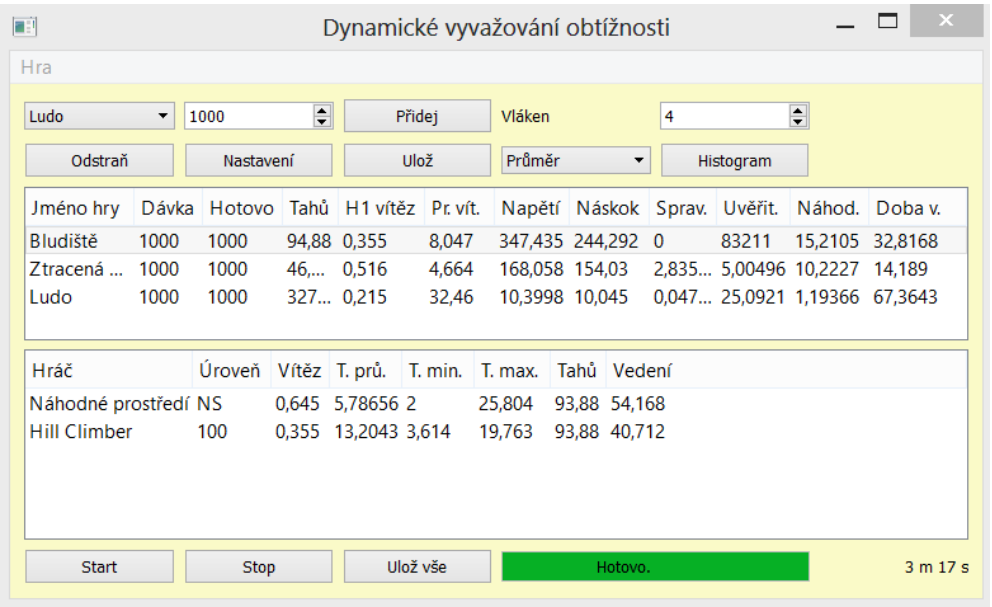
A.1. Aplikace

Ovládání aplikace by mělo být intuitivní. V hlavním menu jsou volby Nová hra, Změnit hru, Nastavení a Dávkové spouštění. Nová hra restartuje hru s novým nastavením. Ve změnit hru lze vybrat jednu ze tří implementovaných her a v Nastavení ji nakonfigurovat.

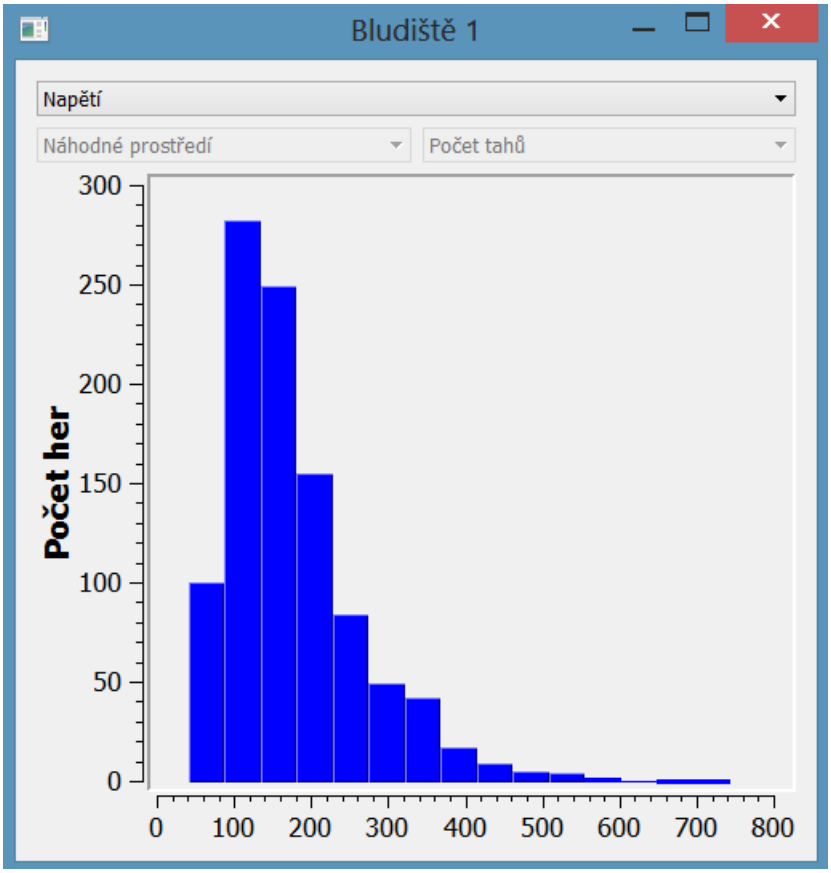
Menu Dávkové spouštění slouží k provádění experimentů. Zpět z Dávkového spouštění se uživatel dostane vybráním Nová hra z hlavní nabídky.

Uživatel přidá novou dávku zvolením dvou parametrů. Z výběrového menu vybere hru a nastaví počet iterací experimentu. Tlačítkem Přidej vytvoří jeden nový experiment, který se objeví v seznamu navolených experimentů.

Po označení experimentu v seznamu může uživatel využívat čtveřici tlačítek Odstraň, Nastavení, Ulož, Histogram. Odstraň vymaže experiment z dávky, Nastavení spustí nastavení pro hru, Ulož umožňuje uložit výsledky proběhlého experimentu, Histogram otevře okno, v kterém lze zobrazit jednotlivé metriky pro jeden experiment jako histo-



Obrázek 20. Uživatelské rozhraní dávkového spouštění experimentů.



Obrázek 21. Histogram metriky Napětí pro hru bludiště a 1000 iterací experimentu.

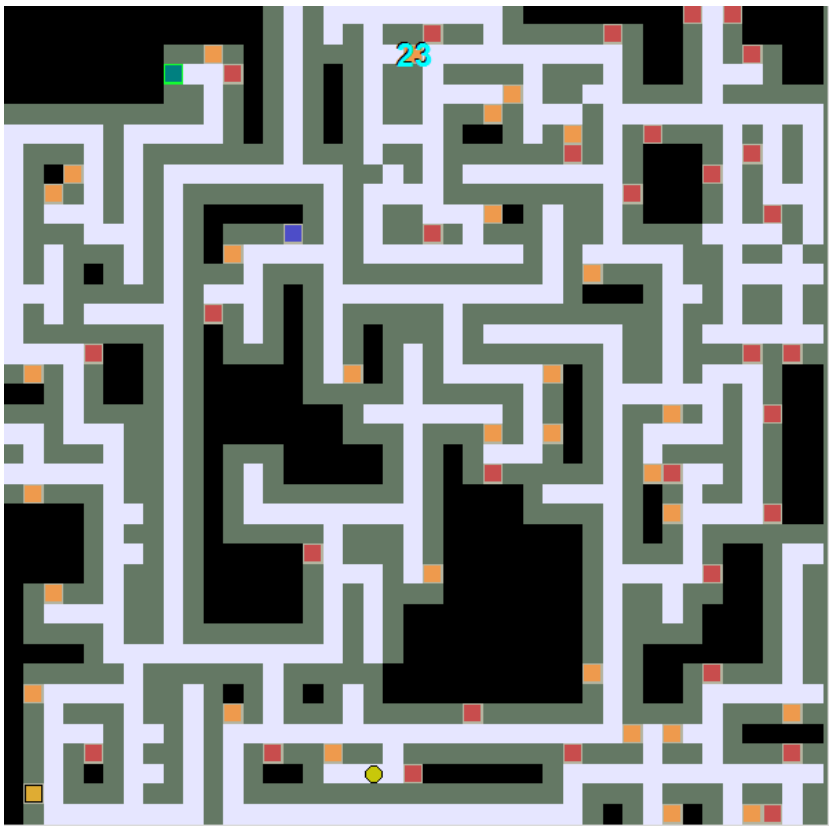
gram. Dále je ve stejném řádku možnost voleb Průměr, Směrodatná odchylka, Minimum a Maximum. Dle volby se po proběhnutí experimentů v jejich seznamu zobrazují příslušné hodnoty z daných experimentů.

Pomocí tlačítka Start se spustí navolené experimenty jeden za druhým. Iterace v rámci jednoho experimentu se rozdělí do předem definovaného počtu vláken. Tlačítkem Stop se experimenty zastaví, ale nechají se doběhnout aktuálně rozehrané hry. Pomocí tlačítka Uložit vše lze hromadně uložit výsledky všech experimentů do zvolené složky.

V průběhu experimentu se zobrazuje uplynulý čas a velice hrubý odhad skončení experimentů. Spočítá se dle průměrné doby na jednu iteraci od začátku spuštění experimentů a dle počtu zbylých iterací ve všech experimentech.

Po označení experimentu v seznamu se zároveň zobrazí ve spodním okně několik metrik k jednotlivým hráčům. Pro každého hráče je uvedeno jeho jméno, úroveň, poměr vítězství. T. prů. T. min., T. max. značí průměrný, minimální a maximální počet tahů hráče během hry. Poslední dvě metriky znamenají kolikrát byl hráč na tahu a kolik kol byl ve vedení.

V seznamu experimentů se zobrazuje jméno hry, celkový a proběhlý počet iterací experimentu, celkový počet tahů, poměr vítězství prvního hráče a metriky prohození vítězů, Napětí, Náskok, Spravedlnost, Uvěřitelnost, Náhodnost, Doba vedení v tomto pořadí.



Obrázek 22. Uživatelské rozhraní hry Bludiště. Zelený čtverec představuje možnou bombu.

A.2. Hry

V této části stručně popíši uživatelské rozhraní jednotlivých her. Nezmiňuji zde pravidla her, která byla uvedena v páté kapitole.

Nastavení her je stejné pro dávkové spouštění i při hraní člověk proti počítači. V levé polovině uživatel může nastavit algoritmy pro jednotlivé hráče a případně jejich úroveň, pokud to daný algoritmus umožňuje. Pod nastavením hráčů jsou speciální nastavení pro jednotlivé hry popsány níže.

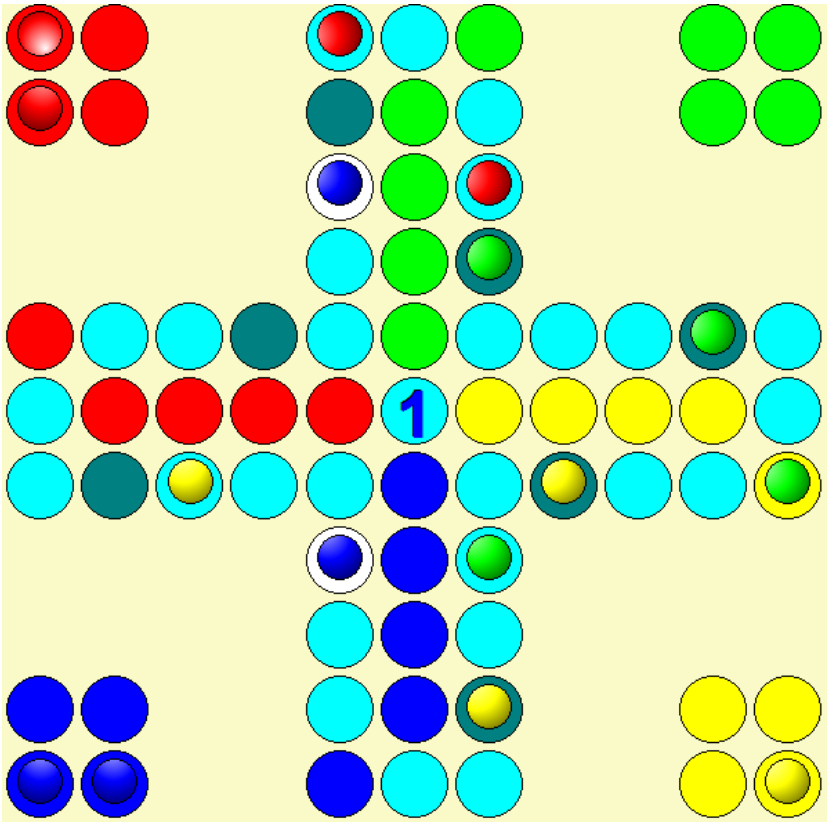
V pravé části nastavení se volí hráč prostředí a koeficienty pro jednotlivé metriky. Nastavení se stane platné až po uložení tlačítkem Uložit.

A.2.1. Bludiště

Hra se ovládá pomocí levého tlačítka myši. Hráč ovládá žlutou kuličku a snaží se dostat včas ke všem bombám, které jsou na mapě znázorněny zelenými čtverci. Hráč vždy kliknutím vybere, které další dveře se mají otevřít. K otevřeným dveřím postava dojde skokem. Dveře jsou znázorněny oranžovými, modrými a červenými čtverci. Dle barvy dveří a jejich vzdálenosti od předchozí pozice, se hráči odečtou zbývající kroky do konce hry.

Zbývající počet kroků do konce hry zobrazuje tyrkysové číslo v horní části hry.

V případě, že některá bomba se stane nedostupnou, tak byla falešná a zmizí z mapy.



Obrázek 23. Uživatelské rozhraní hry Ludo. Bílý podklad značí aktivní figurky.

Speciální nastavení : Před hrou hráč může upravit parametry velikosti bludiště a počáteční limit na počet kroků.

A.2.2. Ludo

Hra Ludo se také ovládá pomocí myši. Kostkou se hází automaticky. Hodnota, která padla, je znázorněna uprostřed herní plochy. Hráč může pohnout s figurami, které jsou na bíle zvýrazněném poli. Jestliže figurkou nemůže pohnout, tak buď mu v tom brání vlastní figurka, kterou by jinak vyhodil, nebo je před figurkou již obsazené bezpečné místo.

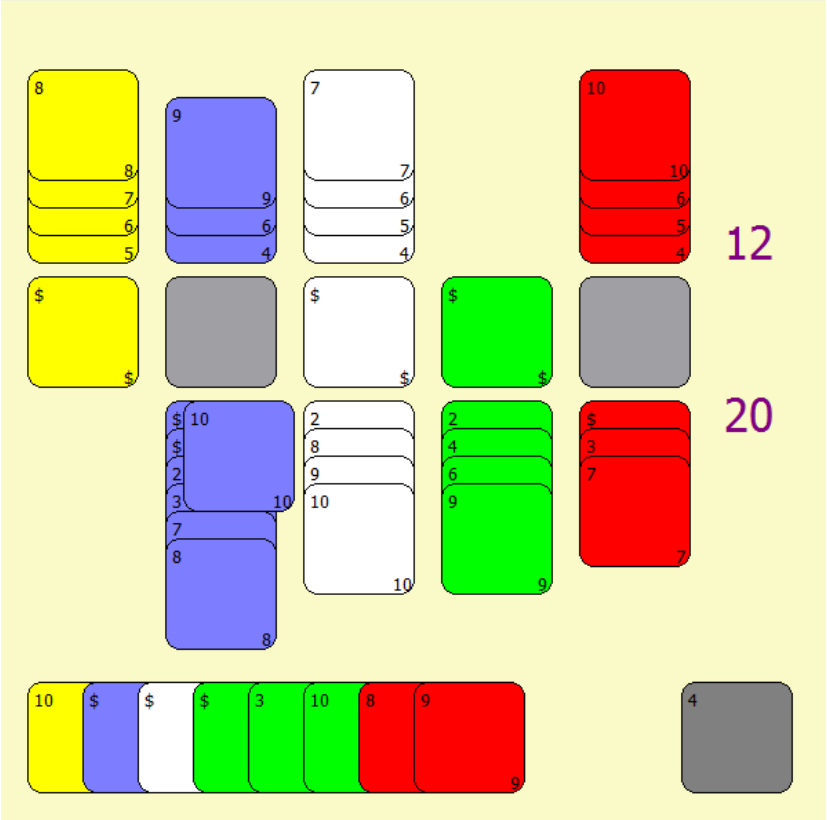
Bezpečná místa jsou zvýrazněna tmavší barvou.

V případě, že hráč nemá ani jednu figurkou na hlavním plánu, hází kostkou až třikrát. Pouze první hod se provede automaticky, ostatní hody provádí hráč kliknutím doprostřed herního pole.

Speciální nastavení : Lze nastavit hru pouze dvou hráčů. V tom případě se ignoruje nastavení druhého a čtvrtého hráče.

A.2.3. Ztracená města

Hra se opět ovládá pouze pomocí levého tlačítka myši. Každý tah se vždy skládá ze třech kliknutí. Prvním kliknutím na kartu v ruce hráč vybere, s kterou kartou bude



Obrázek 24. Uživatelské rozhraní hry Ztracená města.

hrát.

Při druhém kliknutí hráč vybírá, co s kartou udělá. Může jí buď zahodit vybráním příslušného odkládacího balíčku, které jsou na začátku hry znázorněny šedivým čtvercem se zaoblenými hranami. V případě, že kartu může zahrát, klikne pod odkládací balíček dané barvy.

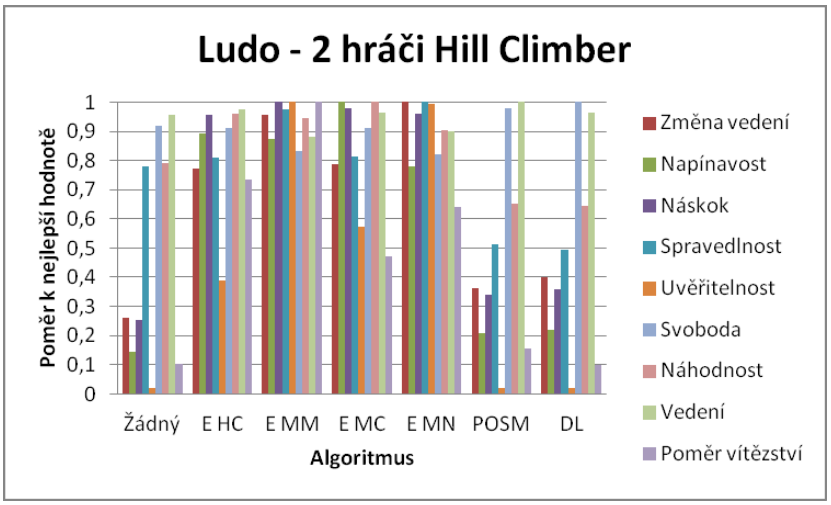
Nakonec třetím kliknutím hráč vybere, odkud dobere kartu. Jedna z možností je dobrat z dobíracího balíčku, který je znázorněn šedivě v pravém dolním rohu. Je na něm číslo udávající počet zbývajících karet v balíčku. V případě, že jsou na stole odložené karty, může hráč kliknout na jednu z nich a vzít si ji do ruky.

Hráčovi se vždy zvýrazňují oblasti ve hře, které jsou zrovna aktivní a hráč na ně může kliknout.

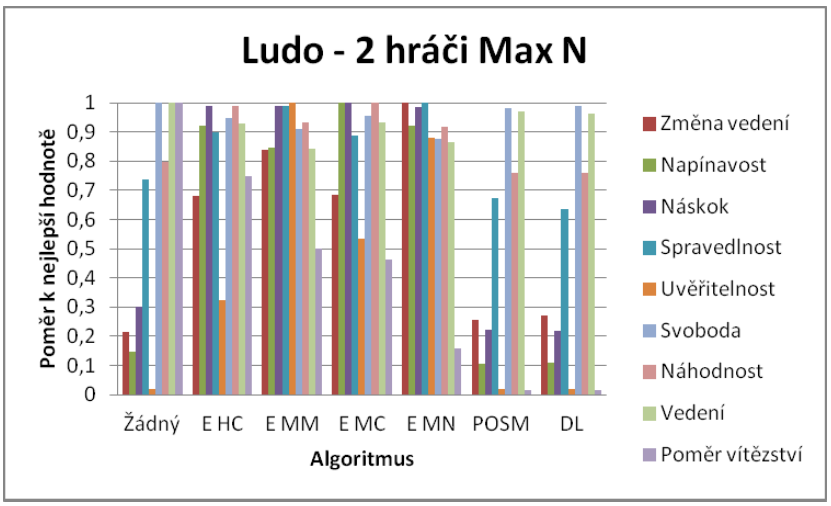
Speciální nastavení : Lze nastavit počet karet v ruce. Oficiální pravidla hry umožňují variantu 8 a 5 karet v ruce.

Příloha B.

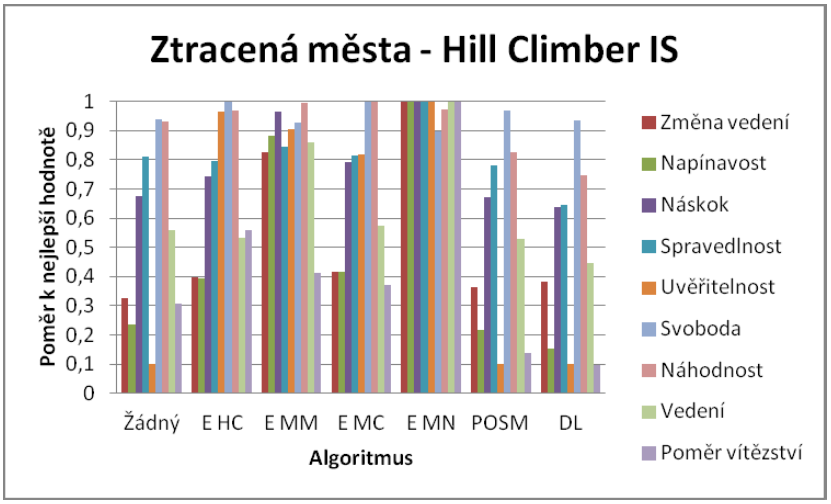
Doplňující grafy experimentů



Obrázek 25. Srovnání DDA alg. ve hře Ludo s dvěma Hill Climber hráči. (úrovně 50 a 100)



Obrázek 26. Srovnání DDA alg. ve hře Ludo s dvěma Max N hráči. (úrovně 50 a 100)



Obrázek 27. Srovnání DDA alg. ve hře Ztracená města s Hill Climber IS hráči. (úrovně 50 a 100)

Literatura

[1] R. Lopes a R. Bidarra. "Adaptivity Challenges in Games and Simulations: A Survey". In: *Computational Intelligence and AI in Games, IEEE Transactions on* 3.2 (červ. 2011), s. 85–99. ISSN: 1943-068X. DOI: 10.1109/TCIAIG.2011.2152841.

[2] A. Saltsman. *Game Changers: Dynamic Difficulty*. URL: http://www.gamasutra.com/blogs/AdamSaltsman/20090507/1340/Game_Changers_Dynamic_Difficulty.php (cit. 16. 02. 2013).

[3] TvTropes.org. *Dynamic Difficulty*. URL: http://tvtropes.org/pmwiki/pmwiki.php/Main/DynamicDifficulty (cit. 16. 02. 2013).

[4] BGG.com. *Power Grid*. URL: http://boardgamegeek.com/boardgame/2651/power-grid (cit. 21. 04. 2013).

[5] Bethesda Softworks LLC. *Fallout*. URL: http://fallout.bethsoft.com/index.html (cit. 21. 04. 2013).

[6] D. Ashlock. "Automatic generation of game elements via evolution". In: *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*. Srp. 2010, s. 289–296. DOI: 10.1109/ITW.2010.5593341.

[7] Robin Hunicke. "The case for dynamic difficulty adjustment in games". In: *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. ACE '05. Valencia, Spain: ACM, 2005, s. 429–433. ISBN: 1-59593-110-4. DOI: 10.1145/1178477.1178573. URL: http://doi.acm.org/10.1145/1178477.1178573.

[8] Guy Hawkins, Keith Nesbitt a Scott Brown. "Dynamic difficulty balancing for cautious players and risk takers". In: *Int. J. Comput. Games Technol.* 2012 (led. 2012), 3:3–3:3. ISSN: 1687-7047. DOI: 10.1155/2012/625476. URL: http://dx.doi.org/10.1155/2012/625476.

[9] Konami. *The Evolution of the Beautiful Game*. URL: http://uk.games.konami-europe.com/news.do?idNews=251 (cit. 16. 02. 2013).

[10] GiantBomb.com. *A. I. Director*. URL: http://www.giantbomb.com/ai-director/3015-2539/ (cit. 16. 02. 2013).

[11] Heather Barber a Daniel Kudenko. "Dynamic Generation of Dilemma-based Interactive Narratives". In: *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE 2007)*. Ed. Jonathan Schaeffer a Michael Mateas. Stanford, California, USA, 6. červ. 2007.

[12] RocstarGames.com. *Rockstar Game Tips: Difficulty Settings - Getting Started in Max Payne 3*. URL: http://www.rockstargames.com/newswire/article/34161/rockstar-game-tips-difficulty-settings-getting-started-in-max-pa.html/ (cit. 16. 02. 2013).

[13] J. Tolentino. *Good Idea, Bad Idea: Dynamic Difficulty Adjustment*. URL: http://www.destructoid.com/good-idea-bad-idea-dynamic-difficulty-adjustment-70591.phtml (cit. 16. 02. 2013).

[14] Microsoft Corporation. *Kinect for Windows*. URL: http://www.microsoft.com/en-us/kinectforwindows/ (cit. 21. 04. 2013).

[15] Nintendo. *Wii is more than a game machine*. URL: http://www.nintendo.com/wii (cit. 21. 04. 2013).

[16] J. Heer. "Balancing Exertion Experiences". In: *Movement-Based Gameplay* (2012).

[17] G. Doyle. "Motivating Elderly People to Exercise Using a Social Collaborative Exergame with Adaptive Difficulty". In: *ECGBL* (2012).

[18] Robby Goetschalckx et al. "Games with Dynamic Difficulty Adjustment using POMDPs". In: (2010).

[19] A. Mihailidis. "A Decision-Theoretic Approach to Task Assistance for Persons with Dementia". In: *International Joint conference on Artificial Intelligence* (2005).

[20] M. Bach. *Rocksmith - recenze*. URL: http://games.tiscali.cz/recenze/rocksmith-recenze-61003 (cit. 15. 02. 2013).

[21] Ubisoft. *Rocksmith - Dynamic Difficulty*. URL: http://www.youtube.com/watch?v=R0yhx5aDjws (cit. 15. 02. 2013).

[22] P. Peerdeman. "Mijn Naam is Haas, Intelligent Tutoring in Educational Games". master thesis. VU University Amsterdam, 2010.

[23] J. Samek. *Flow...?!* URL: http://flowsport.webnode.cz/ (cit. 03. 03. 2013).

[24] Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. First Edition. Harper Perennial, 13.1991. ISBN: 0060920432.

[25] J. Chen. "Flow in Games". Master. University of Southern California's School of Cinematic Arts, 2006.

[26] Lennart E. Nacke. "Flow in Games: Proposing a Flow Experience Model". In: *Fun and Games* (2012).

[27] S. Wright. *"In the zone": enjoyment, creativity, and the nine elements of "flow"*. URL: http://www.meaningandhappiness.com/zone-enjoyment-creativity-elements-flow/26/ (cit. 03. 03. 2013).

[28] Guillermo Gomez-Hicks a David Kauchak. "Dynamic game difficulty balancing for backgammon". In: *Proceedings of the 49th Annual Southeast Regional Conference*. ACM-SE '11. Kennesaw, Georgia: ACM, 2011, s. 295–299. ISBN: 978-1-4503-0686-7. DOI: 10.1145/2016039.2016115. URL: http://doi.acm.org/10.1145/2016039.2016115.

[29] A. Champandard. *The AI from Half-Life's SDK in Retrospective*. URL: http://aigamedev.com/open/article/halflife-sdk/ (cit. 22. 04. 2013).

[30] S. Bakkes, P. Spronck a J. van den Herik. "A CBR-Inspired Approach to Rapid and Reliable Adaption of Video Game AI". In: *ICCBR* (2011).

[31] M. Matteucci. *K-Means Clustering*. URL: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html (cit. 22. 04. 2013).

[32] H. Hsieh a L. Wang. "A Fuzzy Approach to Generating Adaptive Opponents in the Dead End Game". In: *Asian Journal of Health and Information Sciences*, s. 19–37.

[33] Renzo De Nardi Julian Togelius a Simon M. Lucas. "Making Racing Fun Through Player Modeling and Track Evolution". In:

[34] Abdelkader Gouaïch et al. "Digital-pheromone based difficulty adaptation in post-stroke therapeutic games". In: *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*. IHI '12. Miami, Florida, USA: ACM, 2012, s. 5–12. ISBN: 978-1-4503-0781-9. DOI: 10.1145/2110363.2110368. URL: http://doi.acm.org/10.1145/2110363.2110368.

[35] Olana Missura a Thomas Gärtner. "Predicting Dynamic Difficulty". In: *NIPS*. 2011, s. 2007–2015.

[36] L. Ilici et al. "Dynamic difficulty for checkers and Chinese chess". In: *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*. Zář. 2012, s. 55–62. DOI: 10.1109/CIG.2012.6374138.

[37] Ya'nan Hao et al. "Dynamic Difficulty Adjustment of Game AI by MCTS for the game Pac-Man". In: *Natural Computation (ICNC), 2010 Sixth International Conference on*. Sv. 8. Srp. 2010, s. 3918–3922.

[38] Bin Wu et al. "Dynamic difficulty adjustment based on an improved algorithm of UCT for the Pac-Man Game". In: *Electronics, Communications and Control (ICECC), 2011 International Conference on*. Zář. 2011, s. 4255–4259. DOI: 10.1109/ICECC.2011.6066649.

[39] P. Vasconcelos. *Lost Cities*. URL: http://www.ncc.up.pt/~pbv/stuff/lostcities/ (cit. 01. 04. 2013).