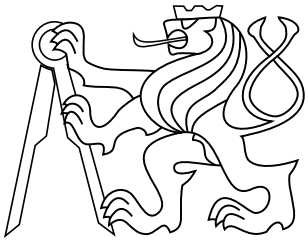
diplomová práce

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

Text of acknowledgement...

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.

Abstrakt

Text abstraktu česky...

Klíčová slova

Dynamické vyvažování obtížnosti her; adaptivní UI; teorie her; ...

Abstrakt

Text of abstract in English...

Keywords

Dynamic difficulty game balancing; adaptive AI; game theory; ...

Obsah

Seznam obrázků

Zkratky

Preliminary text...

abbrv.	explanation
...	...

1 Úvod

Tady něco chybí ...

1.1 Cíl diplomové práce

Cílem diplomové práce je prozkoumat možnosti přístupů k dynamické adaptivitě v počítačových hrách. Zaměřit se na přístupy, jež lze využít v nezměněné, či modifikované podobě v oblasti teorie her. Dalším cílem je vymyslet nový přístup a ten spolu s existujícími algoritmy implementovat a otestovat na třech vybraných jednoduchých hrách. Pomocí metrik změřit kvalitu jednotlivých algoritmů a vzájemně je porovnat. V ideálním případě by měl vlastní přístup předčit přístupy existující.

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.

Se zkratkou DDA se nejčastěji setkáme v oblasti počítačových her a simulací, ale toto prostředí není závaznou podmínkou. Příklady budou uvedeny v sekci 1.3. Cílem DDA je přizpůsobovat proměnné systému co nejlépe aktuálním potřebám uživatele. S typem aplikace úzce souvisí druh potřeb uživatele. Lišit se budou potřeby pacientů využívajících rehabilitační programy a potřeby hráčů počítačových her. Dále se v textu zaměřím na využití DDA v počítačových hrách, případně v tzv. serious game, které mají za úkol udržet uživatele u užitečné činnosti formou zábavné hry.

Synonymy pro DDA jsou např. dynamic difficulty balancing, dynamic game balancing, auto-dynamic difficulty, adaptive difficulty. Ve čtyřech z pěti pojmů se vyskutuje slovo difficulty (obtížnost), které jsem doposud nezmínil. Vyvažování obtížnosti je základem velké většiny hry využívající koncept 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. U komerčních produktů se pak jedná o ztrátu financí. U programu využívaném při rehabilitaci se pak

jedná o ztrátu motivace v pokračovaní rehabilitací. Obtížnost hry tedy přímo souvisí s její zábavností a udělat hru zábavnou pro co největší masu lidí je cílem DDA. Míra zábavnosti hry není závislá pouze na její obtížnosti. Jak lze definovat zábavu a jak ji lze měřit více popíši v kapitole 2.2.2.

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?

Dle svých odpovědí každý určitě zvládne zařadit hru 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

První dělení rozlišuje stav, kdy je hráč obeznámen s dynamickou obtížností a kdy naopak je mu to zatajeno. 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 je si vědomo podvádění v závodních hrách jako je Mario Kart, či Need For Speed, 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. 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, kde se při vstupu na nové území generují nepřátelé, a to dle levelu hráčova avatara. Offline adaptivitu lze považovat za jedinou možnou při vytváření různých 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[2], 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. 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 šťouch, který může odstranit z plochy soupeřovu koule, a tedy mu tím může pomoci dorovnat skóre.[3]

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 umělou inteligenci.
4. Příběh.
5. Herní scénáře a úkoly.

Všechny jednotlivé komponenty mohou být adaptivní a přizpůsobovat se konkrétnímu hráči. *Svět a objekty v něm* : V [2] vytvářejí adaptivně logické úlohy, Hamlet [4] 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. [5] *NPC a jejich umělou inteligenci* : V Pro Evolution Soccer 2008 se soupeř přizpůsobují strategii hráče. [6] *Příběh* : Ve Valve považují výběr typu nepřátel dle aktuálního stavu hráče ve hře Left 4 Dead jako formu vyprávění příběhu. [7] Dále se touto problematikou zabývali např. Barber a Kudenko [8]. *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. Adaptivní obtížnost programu může zefektivňovat léčbu nemocných lidí, nahrazovat osobního trenéra či učitele. 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 hardcore 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 stránce [9] 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 konci úrovně a má plné zdraví i munici, hra vygeneruje těžkého soupeře „Tank“. [7]

Max Payne 3

Hra Max Payne 3 obsahuje celkem 5 statických obtížností(Easy, Medium, Hard, Hardcore, Old School), 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(painkillers), 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 painkiller navíc až do maximálního limitu devíti painkillerů. Na těžkou obtížnost je dynamická obtížnost více limitovaná. Jestliže hráč zemře 5 krát po sobě, dostane jeden painkiller. Pokud zemře podesáté, dostane druhý painkiller. Další léky mu hra již nepřidává. [10]

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 tak slabí nepřátelé se nepohybují. [11]

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. Nejlepší strategií je projet do posledního kola na druhé pozici, což moc nedává smysl. [3]

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 finty může fungovat jednou, dvakrát, ale později již naprosto stejná finta nevede k vítězství. [6]

1.3.2 Cvičení

Herní zařízení jako jsou Microsoft Kinect a Nintendo Wii 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 [12] se zaměřili na jogging v páru.

Podpora pohybu starších lidí

Evropská populace stárne a odmítání pohybu staršími lidmi se stává závažným problémem. Z nedostatku fyzické námahy klesá síla a ohebnost těchto lidí, ztrácejí kostní hmotu a tím zvyšují pravděpodobnost pádu a zlomení některé z končetin. Z tohoto důvodu se skupina z Technologického institutu zaměřila na vývoj hry, jež má starší lidi motivovat k pohybu a jejíž nedílnou součástí je vyvažování obtížnosti. [13]

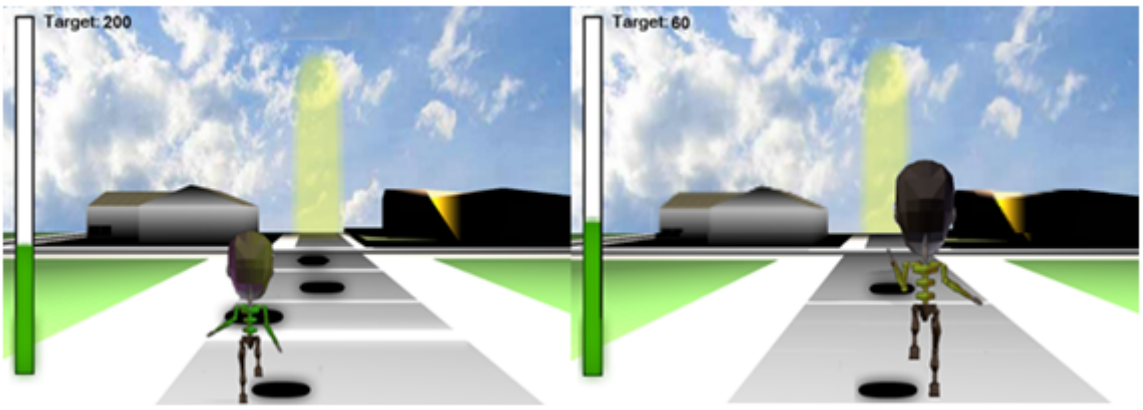
Hra je vytvářena v HTML5 pro běžné použití ve webových prohlížečích a využívá Microsoft Kinect ovládání.

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 a tím je motivuje k opakovanému hraní.

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.



Obrázek 1 Screenshoty HTML5 hry. Vlevo hraje nadaný hráč, vpravo s menší dovedností. [13]

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

V článku Balancing Exertion Experiences [12] 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. [14]

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. [15]

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.

Ve sci-fi seriálu Stargate:SG1 ukázali možnost využití DDA při vojenském výcviku. Čím lépe si hrdina vedl ve virtuální realitě, tím více překážek mu bylo kladeno do cesty. [16]

Dále přiblížím komerční hru pomáhající s výukou na elektrickou kytaru a diplomovou práci o rozšiřování slovní zásoby předškolních dětí zábavnou formou.

Výuka hry na elektrickou kytaru

Známé herní vydavatelství Ubisoft vydalo během loňského podzimu hru Rocksmith, která 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."[17] Tímto lze popsat statickou část obtížnosti, ale autoři se zaměřili i na dynamické vyvažování obtížnosti a sami to vyzdvihují ve svém propagačním videu.[18] 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[19] zabýval využitím DDA u výukových her. Vytvořil hru Mijn naam is Haas(holandsky Moje jméno je Zajíc), která 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 a 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 žáky druhých tříd(groep 2) základních škol v Holandsku. [1] 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.

[1] Groep 2 navštěvují pětileté děti. Navštěvování první třídy ve 4 letech je dobrovolné, druhou třídu musí děti navštěvovat povinně. Číst, psát a počítat se začnou učit až v groep 3, která věkem dětí odpovídá prvním třídám v ČR. [20]

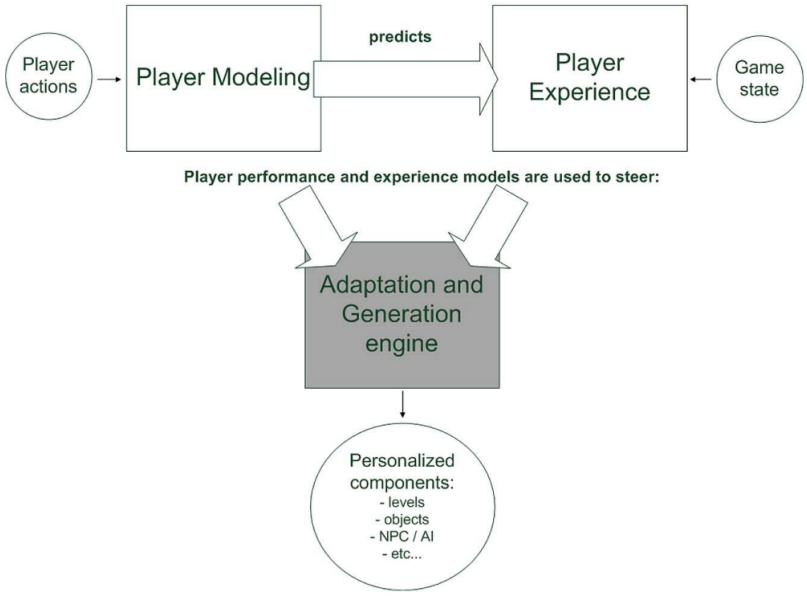
2 Obecné

2.1 Model architektury

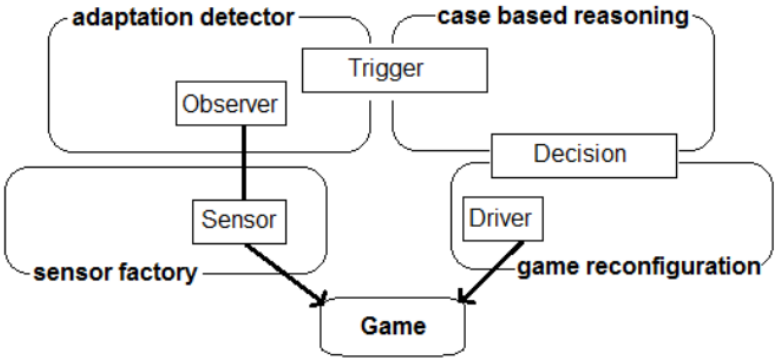
Ať už použijeme DDA s online nebo offline adaptivitou, implicitní nebo explicitní, je zde spousta společných rysů, a proto je na místě vytvořit obecný model architektury dynamického vyvažování obtížnosti. Ve všech případech se snažíme podchytit zjednodušený model hráče na základě jeho projevů ve hře, a poté tyto informace dodáme hře, která herní svět upraví k lepšímu zážitku hráče.

Jeden z možných modelů DDA znázorňuje obr. 2. V principu se zaznamenávají akce hráče a herní proměnné jako je např. počet životů hráče. Na základě těchto logů se vytváří model hráče, model jeho zkušeností, dovedností, preferencí a osobnosti. Model hráče v kombinaci s aktuálním stavem hry slouží k odhadu očekávaného zážitku hráče v dalším stavu hry. A nakonec model zážitku s model výkonu hráče slouží jako vstup adaptačnímu a generačnímu enginu, který posléze upraví herní komponenty jakými je např. umělá inteligence NPC.

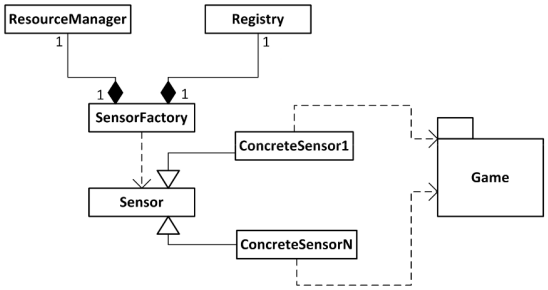
V [21] se zabývali modelem DDA z pohledu návrhových vzorů objektového programování. Abstraktní model je znázorněn na obr. 3. Senzory sbírají důležitá herní data, dle kterých se bude dále rozhodovat. Návrhový vzor Observer je připojen k Senzorům a v případě, že zaznamená zatelnou změnu v systému, vytvoří událost, trigger. Jednotlivé



Obrázek 2 Přehled principů architektury adaptivních her. [1]



Obrázek 3 Návrhové vzory DDA [21]



Obrázek 4 Návrhový vzor sensor factory. [21]

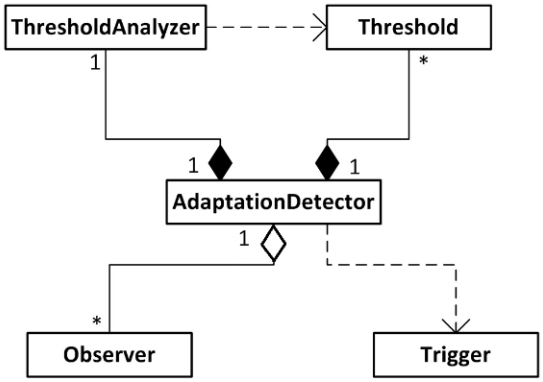
události jsou spojeny s akcemi a dohromady spolu tvoří pravidla uložená v databázi. V případě, že se spustí trigger spojený s akcí v některém z pravidel, rozhodne se v provedení této akce, což má na starost řadič, který má za úkol provést požadovanou změnu do stavu hry.

Dále se podíváme na jednotlivé návrhové vzory detailněji.

2.1.1 Sensor factory

Senzory jsou objekty, které pravidelně čtou herní data[1] a upozorňují na změny zbytek DDA systému. Schéma návrhového vzoru znázorňuje obr. 4. Senzor je abstraktní třídou, která zahrnuje periodické sbírání dat a upozorňující mechanismus. Konkrétní senzory z této třídy dědí a musí přepisovat abstraktní metodu refreshValue(), která zaznamenává konkrétní proměnnou systému. Třída SenzorFactory je zodpovědná za vytváření jednotlivých senzorů a je implementací návrhového vzoru factory. Továrna na senzory vyžaduje název senzoru a objekt, který má monitorovat. Vytvořené senzory si ukládá do registru. V případě, že uživatel zažádá o senzor, který už někdo vytvořil, dostane referenci na tento senzor. V opačném případě zkontroluje v ResourceManageru, jestli vytvořením nového senzoru se poruší některá z omezení zdrojů a pokud ne, senzor vytvoří.

[1] Senzory nemusí zaznamenávat pouze herní data. Mohou zaznamenávat i prostředí uživatele např. pomocí Kinectu, nebo snímat aktuální tep hráče.



Obrázek 5 Návrhový vzor sensor factory. [21]

2.1.2 Adaptation detector

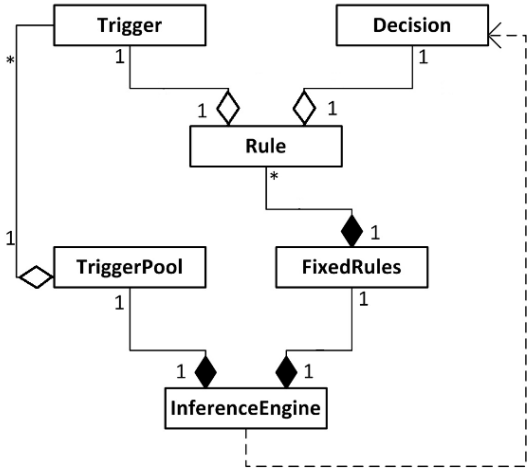
Hrubá data získaná senzory se musí dále zpracovat. Tyto data získává AdaptationDetector pomocí observeru z návrhového vzoru senzor-observer. Na tomto místě se rozhoduje, jestli senzory již zaznamenaly dostatečnou změnu systému. Nedostatečnou změnou může být vystřelení jednoho náboje z plně nabitého revolveru. Naopak vystřílení půlky zásobníku může být významné. O významnost změny se stará TreshholdAnalyzer s Tresholdem. Treshold uchovává parametr hranice a její typ. (menší rovno, větší apod.) V případě dosažení prahu TresholdAnalyzer dá vědět AdaptationDetectoru, který vytvoří trigger, spouštěč. Trigger s sebou může nést další dodatečné informace jako je např. množství přeživších nepřátel. Viz obr. 5.

2.1.3 Case based reasoning

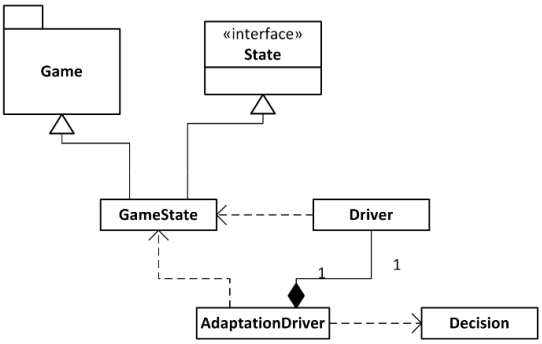
Trigger spustí rozhodování na základě případů. Tento návrhový vzor(obr. 6) se použije, jestliže je možné vyvažování obtížnosti definovat konečným množstvím případů. InferenceEngine obsahuje dvě datové struktury: TriggerPool a FixedRules. Fixed rules obsahují pravidla, která jsou úzce spojená s konkrétní hrou. Pravidlo je kombinací triggeru a akce/rozhodnutí. TriggerPool funguje jako fronta událostí. Do fronty se řadí spuštěné triggery a jsou obsluhovány od nejstaršího. InferenceEngine vždy odebere jeden trigger z poolu, najde ho v databázi FixedRules pravidel a s ním nalezne vhodné rozhodnutí, které se má dále provést.

2.1.4 Game reconfiguration

Posledním krokem je provedení požadované změny v herním světě. Návrhový vzor Game reconfiguration(obr. 7) v sobě obsahuje jiný návrhový vzor, adapter. AdaptationDriver dostane ke zpracování rozhodnutí z InferenceEnginu. AdaptationDriver provede rozhodnutí za pomoci Driveru. Driver mění objekty, jež implementují rozhraní State, přes které zjišťuje aktuální stav objektu. S provedením jeho změny čeká dokud se nestane objekt neaktivním.



Obrázek 6 Návrhový vzor Case based reasoning. [21]



Obrázek 7 Návrhový vzor Game reconfiguration. [21]

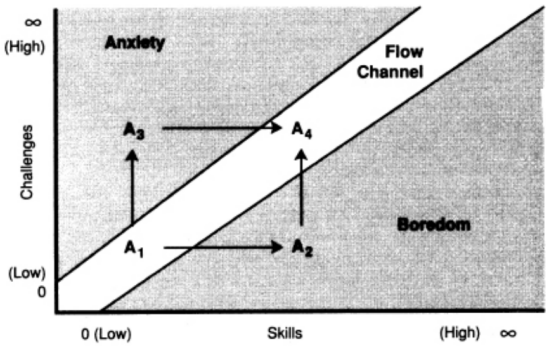
2.2 Zábava

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.

2.2.1 Flow

Flow (tok) je stav mysli, kdy je osoba v průběhu provozování činnosti naprosto soustředěná, pociťuje nadšení, úspěch. „Flow je stav vědomí, kdy je člověk plně zaujatý svou činností. Nezabývá se jinými stimuly z okolí ani svými myšlenkami nebo pocity. Je naprosto soustředěný na prováděnou činnost. Dosahuje většinou, na své poměry,



Obrázek 8 Příklad pohybu jedince ve flow grafu. [23]

nadprůměrných výkonů, ale přitom mu to nepůsobí výraznou námahu. Je to harmonický zážitek, kdy tělo a mysl spolu bez námahy spolupracují. Tento stav je většinou spojen s pocity energie, radosti, harmonie a seberealizace.“ [22] 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. Viz obr. 8.

Jestliže se člověk seznamuje s novou činností, začíná v levém dolním rohu flow grafu. V tu chvíli nemá žádné dovednosti a pravděpodobně se začíná učit provádět činnost od jednoduchých částí po složitější. V knize [23] uvádí Csikszentmihalyi jako příklad takové činnosti hraní tenisu a vysvětluje to na obr. 8. 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.

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. [23]

Dle Csikszentmihalyi se flow skládá z devíti elementů. [24] Ne všechny jsou nutně potřebné k dosažení stavu flow.

1. Jasné cíle

2. Zpětná vazba
3. Vyrovnanost náročnosti úkolů a schopností
4. Splynutí činnosti a vědomí
5. Koncentrace
6. Žádné obavy z neúspěchu
7. Pocit kontroly
8. Změna vnímání času
9. Vnitřní motivace

Rozepisování jednotlivých bodů není záměrem této práce. Bližší informace lze nalézt např. na [24], [22], nebo v originální knize [23].

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.

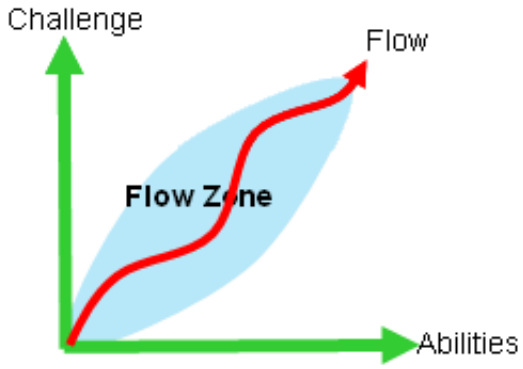
Jsou-li splněny všechny tři body, hráč může ztratit pojem o čase a zcela se do hry ponořit.

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 osmi složkami vycházejícími z 9 složek flow od M. Csikszentmihalyi.

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. Hráč by měl být schopen vnímat svůj pokrok ve hře.

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.



Obrázek 9 Obecně je vhodné hráče udržovat ve flow zóně. [25]

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.

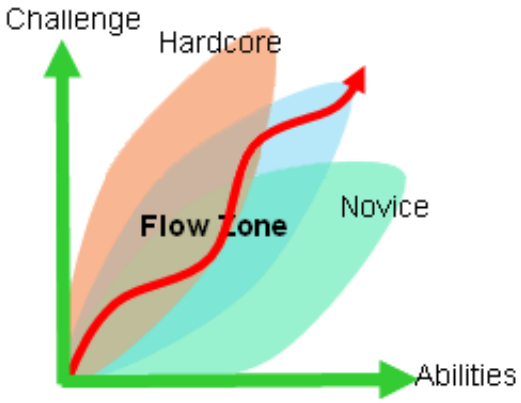
Ponoření se do činnosti : Podobné koncentraci. Hra má pohlcovat a udržovat hráče v maximální pozornosti, ale tak, aby mu to bylo stále příjemné.

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

Zóna flow pro různé hráče

Vraťme se znovu ke flow diagramu 8. 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, jak znázorňuje obr. 9.

Bohužel takový flow diagram nebere v úvahu individualitu hráče. Existují hráči, kteří mají rádi větší výzvy než jsou v tu chvíli schopni zvládnout, lze je nazvat hardcore hráči. Naopak existují příležitostní hráči, kteří netouží po velkých výzvách a nejraději se budou pohybovat lehce pod flow zónou. Těchto specifik si všímá např. práce [5]. Ideální průběh hry pro příležitostné/začínající hráče, běžné hráče a hardcore hráče znázorňuje graf na obr. 10. 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 10 Flow zóna a specifika pro příležitostné a hardcore hráče. [25]

2.2.2 Metriky zábavnosti

Hlavním cílem počítačových her je pobavení hráče. Může být tedy dobré 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 [12]. 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 pojmů z algoritmu Dynamická úroveň 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 zisk jednoho hráče číslo opačné k zisku 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 [27] 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 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.

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.

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.

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.

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.

Vedení

Hra nebývá příliš zábavná, jestliže po většinu času je stejný 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 :

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

Tato metrika nemá žádný význam u jednohráčových her, jelikož vzhledem k jejímu výpočtu bude rovna vždy nule.

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

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.

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.

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. Můžeme ale nalézt společný základní kámen pro metriku uvěřitelnosti, definujeme si uvěřitelnost pole četností.

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.

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 a je potřeba ji určit heuristicky.

Výsledný vzorec pro uvěřitelnost pole četností :

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

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 částečně pozorovatelné markovské procesy, 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

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
POMDP		✓		✓			✓
Producent – konzument		✓	✓	✓	✓	✓	
Case-base reasoning		✓		✓	✓		
Fuzzy pravidla		✓		✓	✓		
Evoluční algoritmus	✓	✓	✓			✓	
Mravenčí feromony		✓		✓			✓
POSM	✓	✓	✓	✓	✓	✓	✓
Dynamická úroveň	✓	✓		✓	✓		
MCTS		✓		✓	✓		

algoritmus z nich posléze vybírá nejvhodnější obtížnost v danou dobu. Z tohoto důvodu 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 Ne z teorie her

3.1.1 Částečně pozorovatelné markovské rozhodovací procesy

Částečně pozorovatelné markovské rozhodovací procesy (POMDP) byly úspěšně použity pro regeneraci rukou lidí po mrtvici [14] a u počítačového asistenta při mytí rukou člověkem trpícím demencí.

V obou případech byla osoba se systémem modelováno pomocí POMDP, které jsou schopné pracovat se sekvenčními dynamickými systémy, ve kterých jsou některé stavy preferované před jinými a ne vše související s procesem je plně pozorovatelné. V těchto případech nelze přímo pozorovat aktuální schopnosti uživatele.

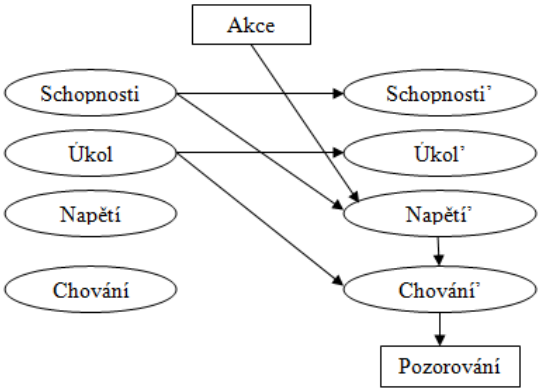
V každém kroku je uložen belief state (pravděpodobnostní distribuce nad všemi stavy). Policy (politika) říká, kterou akci v daném kroku vybrat na základě aktuální belief state, které se na základě vybrané akce a nového pozorování aktualizuje.

Influence diagramy

Obecná POMDP mohou být řešena více exaktními způsoby, ale nejsou řešitelná s dostatečně krátkou odezvou pro naše využití. POMDP pro popsané úlohy může být redukováno na influence diagramy, které jsou snáze řešitelné.

Obecný influence diagram pro adaptivní systém si lze prohlédnout na Obr. 11. Apostrofované proměnné představují odpovídající neapostrofované proměnné v následujícím stavu. Např. hodnota napětí v následujícím stavu závisí pouze na hodnotách schopnosti a provedené akce z předchozího stavu.

V každém kroku se provede právě jedna akce. Stav je popsán proměnnými 4 kategorií. Proměnná schopnosti(ability) je odhadem aktuálních schopností uživatele. Proměnná úkol(task) popisuje aktuálně prováděný úkol osobou. Proměnná napětí(stretch) znázorňuje náročnost aplikace vzhledem k aktuálnímu uživateli. Popisuje rozdíl úrovně obtížnosti zvolené akce a úrovně schopností jedince. Ideálně jsou úrovně shodné, napětí je nulové. Pokud je napětí vysoké, úkol je příliš obtížný. Jestliže je napětí záporné, úkol je příliš snadný. Proměnná chování(behavior) je přímo pozorovatelná. U pomocníka při mytí rukou je jím pozice rukou (ve vodě, na kohoutku atd.) a stav puštění vody.



Obrázek 11 Influence diagram pro adaptivní systémy

Příklad popisu stavu hry

Pro lepší pochopení skupin proměnných a jejich významu uvedu příklady proměnných uvedených v článku [14], rehabilitace po mozkové mrtvici.

Schopností je rychlost učení, jak rychle se každý uzdravuje. Úkol popisuje vektor n(r), kde pro jednotlivé hodnoty odporu ovladače je uložena maximální vzdálenost, kterou je osoba schopna dosáhnout. Napětí zde má zachován svůj význam z obecného popisu. Chování je zde nahrazeno celkovou únavou, která souvisí s následujícími pozorovanými veličinami. TTT, čas potřebný k dosáhnutí na cíl, CTRL, reprezentuje, jestli bylo cvičení vykonáno s pomocí kontroly, COMP, jestli si osoba snažila pomáhat vrchní polovinou těla místo využívání pouze její paže.

Konkrétní influence diagram můžeme vyřešit např. pomocí algoritmu PERSEUS, který najde v dostatečně krátkém čase přibližné řešení. Na základě pozorování a provedených akcí dokáže odhadnout schopnosti uživatele a vzhledem k tomu připravit odpovídající následující úlohu.

3.1.2 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 integrovaný do Half-Life SDK[4], který vyvažuje obtížnost hry právě pomocí výměny zdrojů mezi světem a hráčem. Half-Life patří mezi klasické zástupce first-person shooter (FPS, „střílečky“).

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.

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.

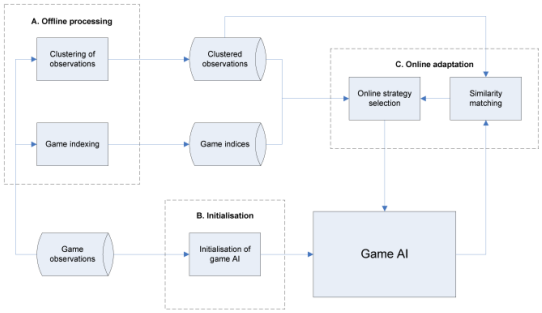
Na straně poptávky manipulovat s nepřáteli (změnou jejich třídy, množství, počtu jejich životů, určením místa jejich objevení se na mapě). Stejně jako u hráče lze přizpůsobit sílu a přesnost jejich zbraní.

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 Case-base reasoning

Další možností pro implementaci DDA je Case-base reasoning. Rozhodování se v dané situaci dle úspěšné strategie použité v minulosti v situaci podobné.



Obrázek 12 Proces adaptivní AI založené na CBR [28]

Tento přístup využili Nizozemci u strategické hry Spring. [28] Proces celé adaptivní AI znázorňuje diagram Obr. 12. Začne se sběrem dat (game observation). Proběhne simulace 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).

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.

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 surovina a jednotek jednotlivých hráčů.

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.

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.

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.

Na závěr nutno podotknout, že tento přístup nedává jistotu stejného chování a výsledku AI hráče jako ve vybrané hře z databáze. Při výběru se porovnávají pouze zjednodušené abstrakce stavu hry a navíc pouze s agregovanými shluky stavů. Výsledné chování hráče se může lišit od očekávaného.

3.1.4 Fuzzy pravidla

Máte-li umělou inteligenci ve své hře založenou na fuzzy pravidlech, může vás zajímat následující způsob tvorby adaptivní AI.

Základní myšlenka 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.

Dead-end

Autoři přístupu zvolili pro jeho testování hru jednoduchou hru Dead-end. 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.

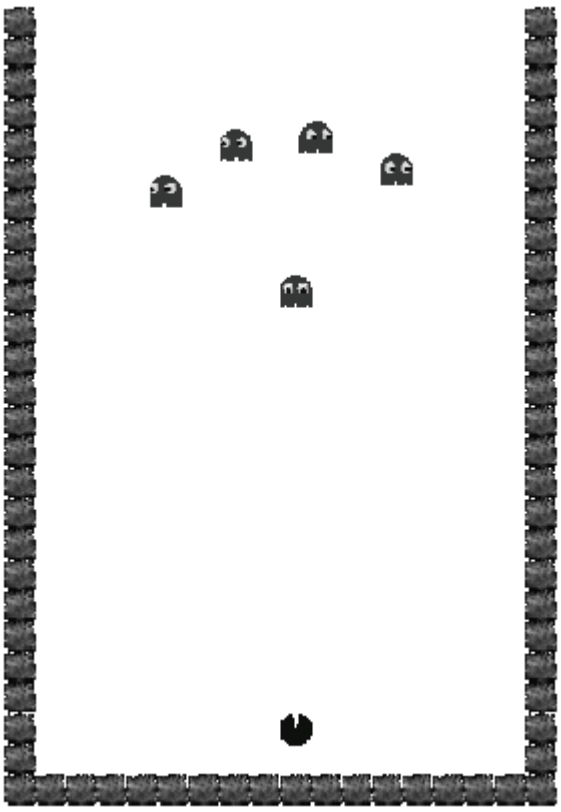
Duchy lze rozdělit do dvou rolí. Duch nejníže vzhledem k souřadnici y tvoří předvoj a jistým způsobem ovládá ostatní duchy, obránce.

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



Obrázek 13 Screenshot ze hry Dead-End. [29]

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*

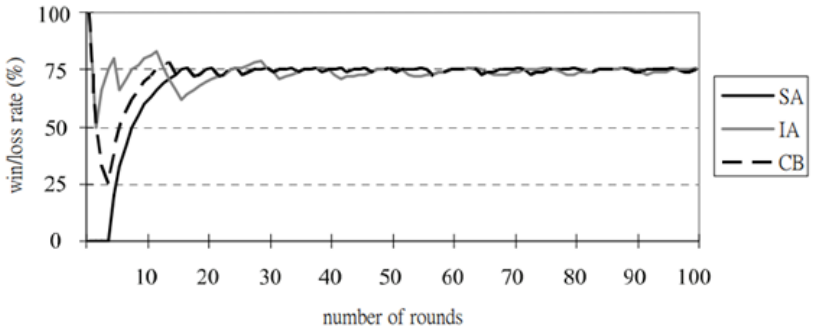
Obrázek 14 Část fuzzy pravidel pro předvoj. [29]

tabulky 40 pravidel z [29] : 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. 14.

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 [29].

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



Obrázek 15 Přibližování se k požadovanému win-rate 75% během 100 kol proti třem různým hráčům. [29]

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 chovat zcela stejně. Proč naopak nedeaktivovat pravidlo, které využíval soupeř nejvíce? V takovém případě by mohla 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.

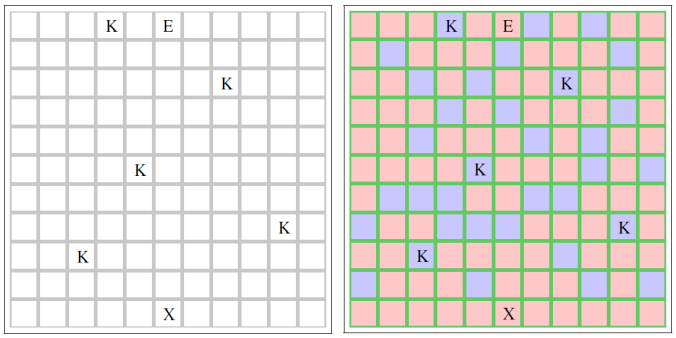
Poté se spočte suma vstupních příslušností hodnot pro každé z pravidel deaktivovaných dříve. Nakonec se zaktivuje pravidlo s největší sumou vstupních příslušností. Jestliže existují dvě pravidla se stejnou hodnotou sumy, zvolí se pravidlo jehož skupina má nejnižší příspěvek.

Výsledky

Na závěr připojuji jeden z grafů ukazující adaptivnost AI k třem různým hráčům s požadovanou hodnotou win-rate 75% Obr. 15. Ke srovnání na cílovou hodnotu došlo kolem 25. hry.

3.1.5 Evoluční algoritmus

K zástupcům statické DDA lze bez váhání zařadit evoluční algoritmy (EA). Evoluční algoritmy zpravidla vyžadují velký výpočetní výkon, a proto nejsou vhodné pro realtime



Obrázek 16 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. [2]

použití. Využitím EA pro DDA se zabývají např. články Automatic Generation of Game Elements via Evolution[2] a Making Racing Fun Through Player Modeling and Track Evolution[30]. V prvním ze zmíněných článků využili EA pro generování různě obtížných úrovní logických her, naopak v druhém generovali závodní trať složenou z různých úseků.

Generování bludišť

V [2] byly 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. 16.

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 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.

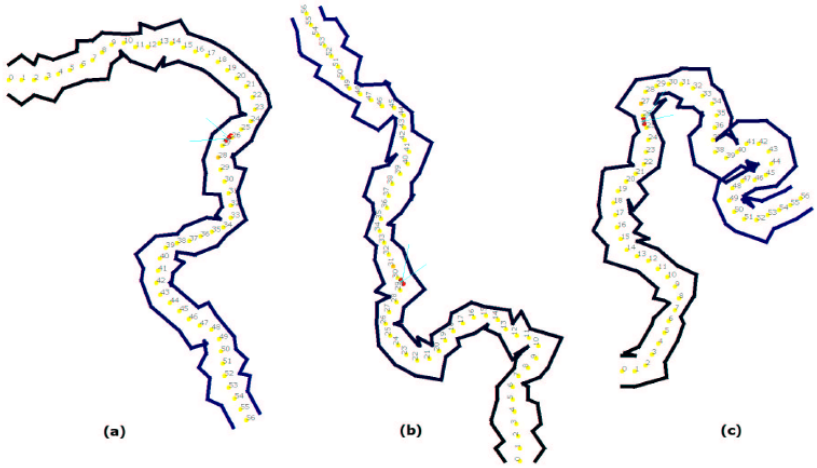
Generování tratě

Cílem práce [30] 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.
- Příliš časté nehody také nejsou ideální. Trať by měla být výzvou tak akorát.
- 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.

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

[1] Délka úseku dána délkou střední čáry.



Obrázek 17 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. [30]

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.

3.1.6 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[31]. Využití mravenčích feromonů je zde velice specifické a v tuto chvíli mě nenapadá jejich širší využití v jiné oblasti než rehabilitace částečně ochrnutých horních končetin, např. v zábavním průmyslu. Přesto je zde uvádím jako zajímavost, která stojí za zmínku.

Cílem práce bylo vytvořit hru, která budu procvičovat znovu ovládnutí pohybu ruky a zároveň bude svou obtížnost přizpůsobovat 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.

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.

(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 18 (a) Zóna schopností, (b) Interakční matice pro stížení hry, (c) Interakční matice pro zjednodušení hry [31]

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. 18.

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

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).

Jak jsem poukázal na začátku této kapitoly, jedná se spíše o zajímavost kvůli velmi specifickému užití metody mravenčích feromonů. Přesto je to inspirativní.

3.2 Využitelné v teorii her

3.2.1 Částečně uspořádaná množina – Mistr

Partially-Ordered-Set Master (POSM) [32] 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.

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.

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:

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.

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**

Příklad s balónky

Algoritmus se lépe vysvětlí na příkladě.[33] 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. Vyzkoušen a testován byl např. na deskových hrách dáma a čínské šachy. [33]

3.2.2 Dynamická úroveň

Autoři tohoto přístupu se zaměřili na dynamické vyvažování obtížnosti v deskových hrách. [27] Sami svůj algoritmus nijak nenazvali, tedy název této podkapitoly je mnou vymyšlen a google ho „nezná“. Alespoň ne ve spojitosti s tímto algoritmem. Možnosti DDA v deskových hrách se od možností DDA v ostatních počítačových hrách výrazně liší. Ve střílečce, real-time strategii můžete obtížnost hry vyvažovat změnou pravidel hry aniž by si toho hráč všiml. Obtížnost může být dána přesností mušky botů ve hře, či množstvím zdrojů, s kterými soupeř pracuje ať už je během hry mohl, či nemohl získat.

Článek označuje jako jedinou možnost DDA v deskových hrách ovlivňovat AI soupeřů. Bohužel s tímto tvrzením nemohu souhlasit. I když se zaměříme pouze na deterministické deskové hry s úplnou informací, kterými jsou např. šachy, máme k dispozici online i offline dynamické vyvažování obtížnosti hry, které by jistě napadli většinu hráčů šachů, jelikož to již mnozí využili. Za offline DDA můžeme považovat odebrání některých herních figur před začátkem hry. K online vyvažování může patřit různé časové omezení na provedení tahu u jednotlivých hráčů. Vedoucí hráčů bude mít méně času na přemýšlení a tím se zvětší pravděpodobnost provedení chybného času a vystřídání hráčů ve vedení.

Popis algoritmu

Zpět k „dynamické úrovni“. Algoritmus má dva předpoklady k jeho využití. Vyžaduje funkci, jež umí ohodnotit všechny následující stavy ve hře. 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.

Algoritmus je vytvořen pro dvouhráčové hry, kde jeden hráč má dynamickou úroveň. Při volbě nového tahu se 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:

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

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**

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.

Ohodnocovací a status funkce

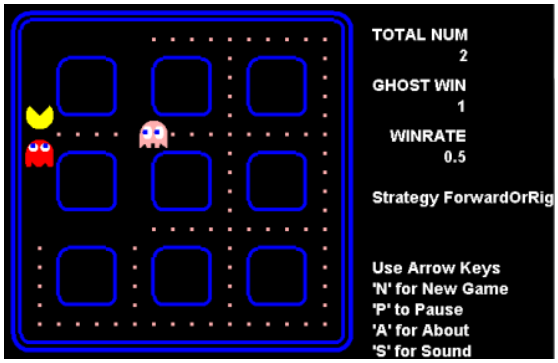
Jaký je rozdíl mezi rankFnc a statusFnc? Každá se využívá v jiné části algoritmu. Vnitřně mohou být stejné. Proč tedy autoři rozlišují funkce 2? V článku se k tomu obecně moc nevyjadřují, ale lze něco usoudit z jejich testovací hry Backgammon. Ohodnocovací rank funkce je zde složitá a bere v úvahu mnoho parametrů popisujících stav hry. Parametry jsou např. 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.

3.2.3 Monte-Carlo prohledávání stromu

Autoři z Pekingské univerzity vyzkoušeli spojení Monte-Carlo Tree Search(MCTS) algoritmu s dnes populárními neuronovými sítěmi. [34][35]

V článcích upozorňují na nevýhodu tradičních metod DDA, kdy se obtížnost uměle mění např. přidáváním dalších a silnějších nepřátele, ale jejich inteligence zůstává stejná.



Obrázek 19 Zjednodušená verze Pac-Mana. Obrázek převzat z [34]

Hráč se v takovém případě může cítit podveden. V Pekingu se vydali cestou dynamického vyvažování umělé inteligence a svůj přístup aplikovali na zjednodušené verzi známé hry Pac-Man.

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.

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.

Pro účely simulace hráč Pac-Mana využíval ForwardOrRight strategii. 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.

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

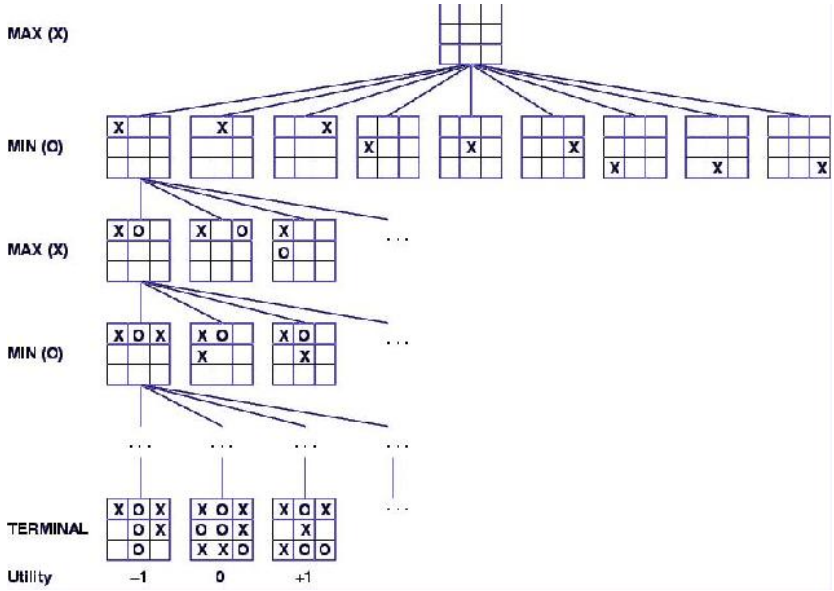
Dělení her

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 2 Klasifikace her dle úplnosti informace a determinismu.

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



Obrázek 20 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. 20. 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. 20 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.

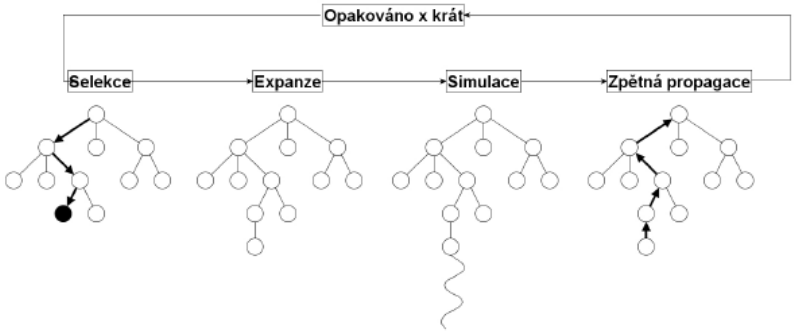
Minimax

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 21 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.

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. $O(b, d) = b^2 d$. 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

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 3.

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 (3.2.2).

5.1.1 Bludiště

Bludiště je jednoduchou hrou pro jednoho hráče. Terorista 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.

Terorista 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.

Tabulka 3 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

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 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 [36] 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 trojice 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 [36] 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.

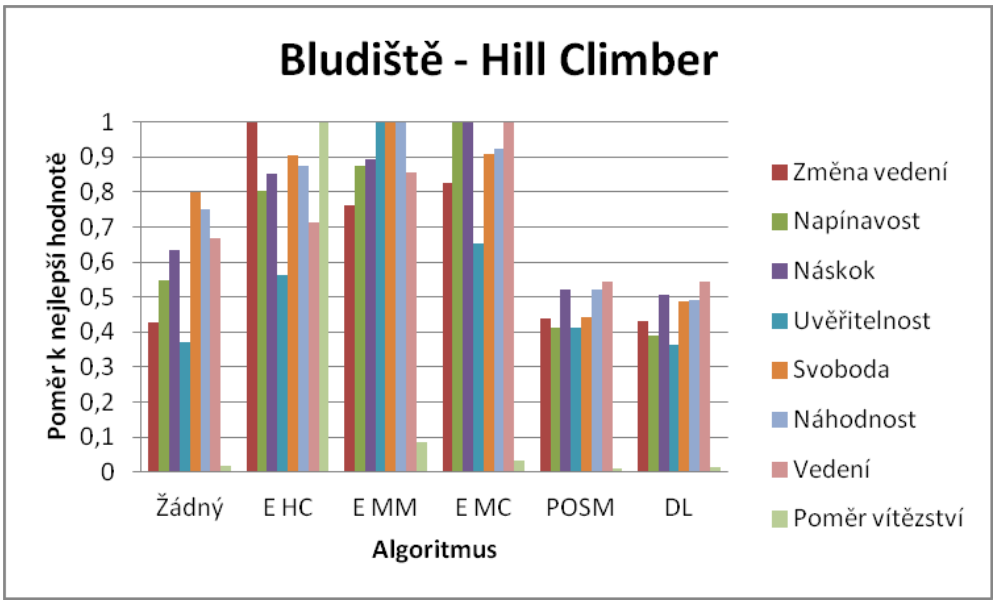
6 Provedené experimenty

6.1 Bludiště

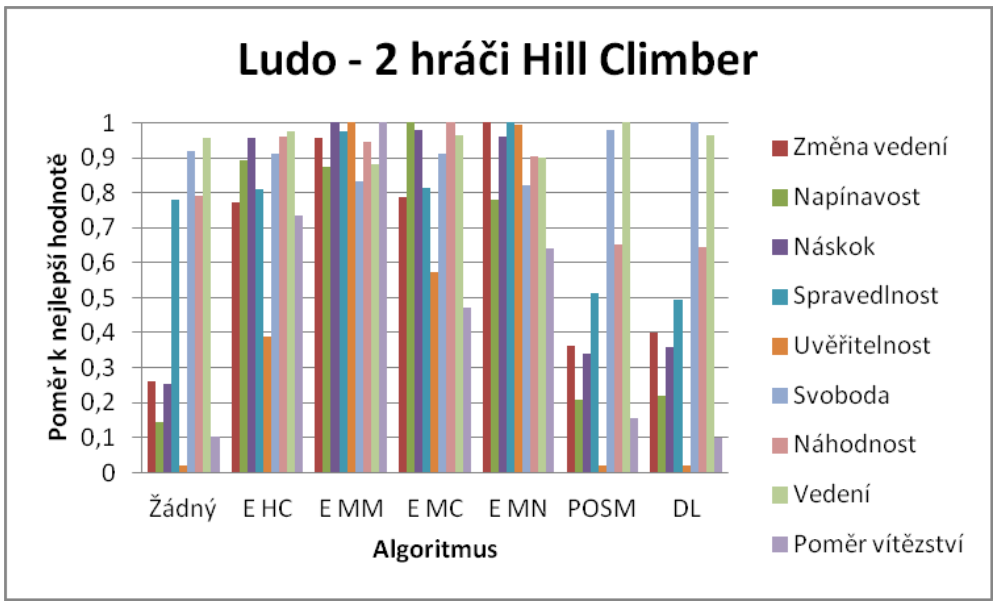
6.2 Ludo

6.3 Ztracená města

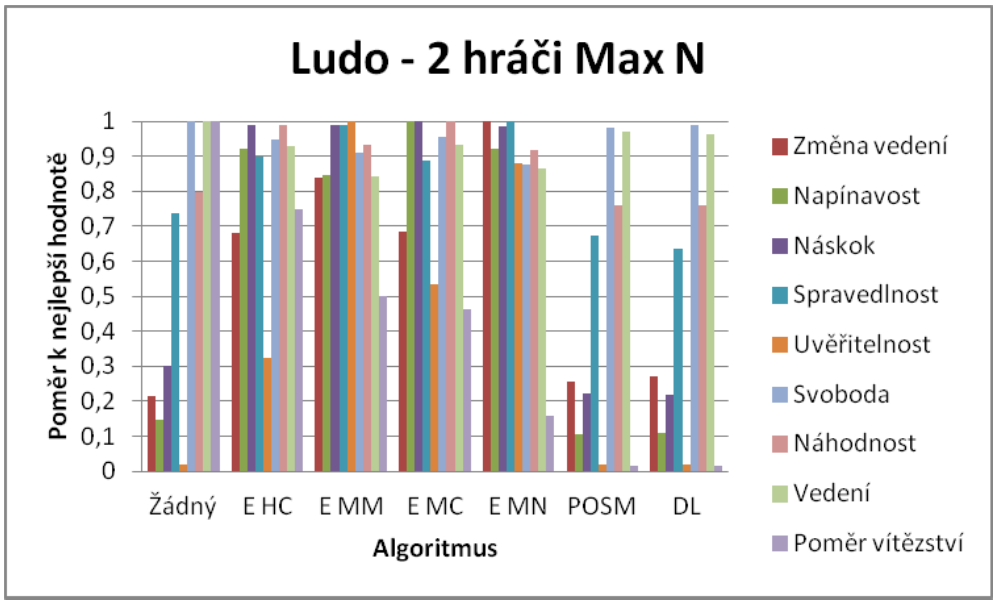
6.4 Celkové srovnání



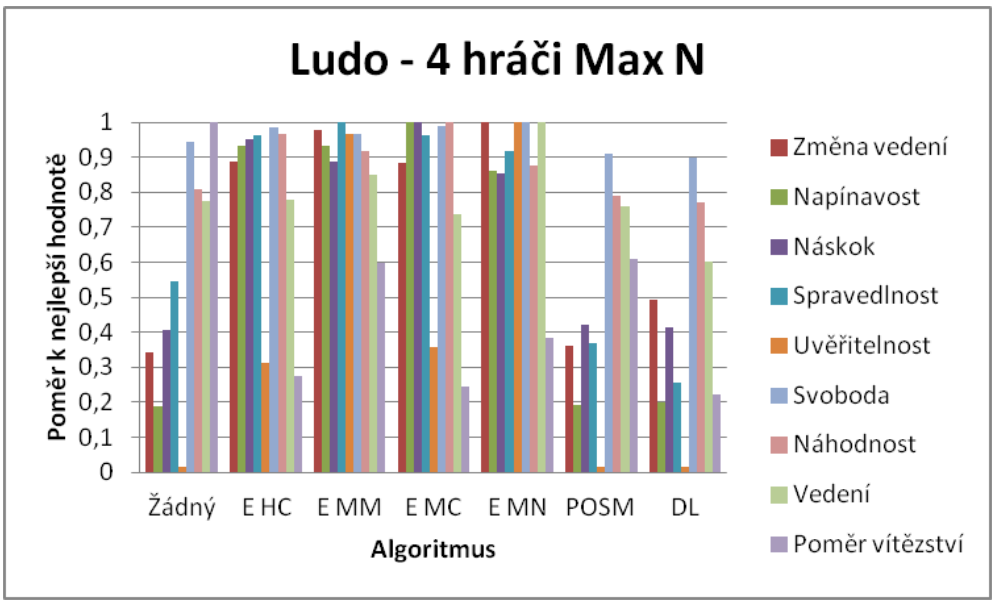
Obrázek 22 popisek



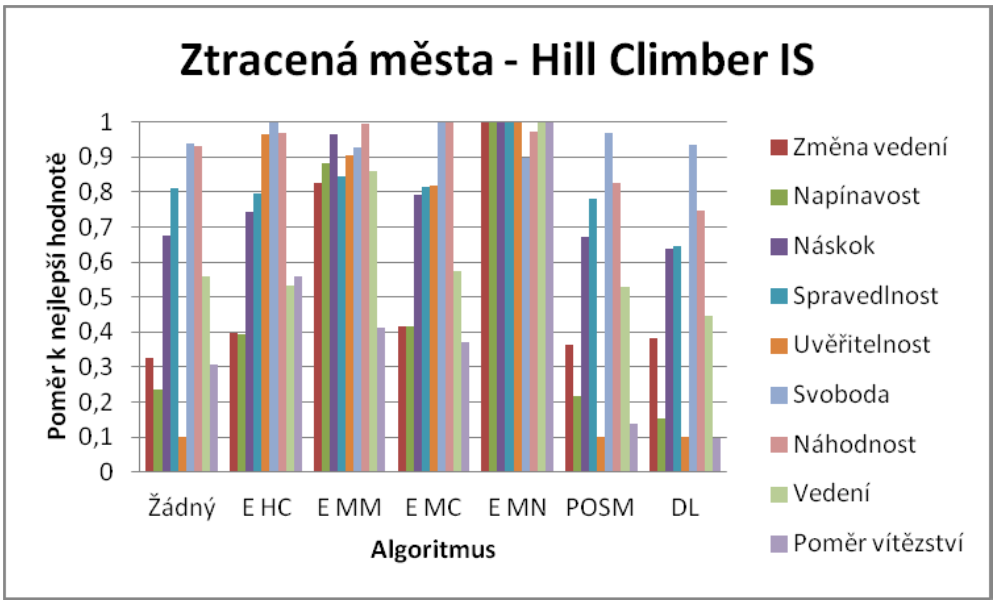
Obrázek 23 popisek



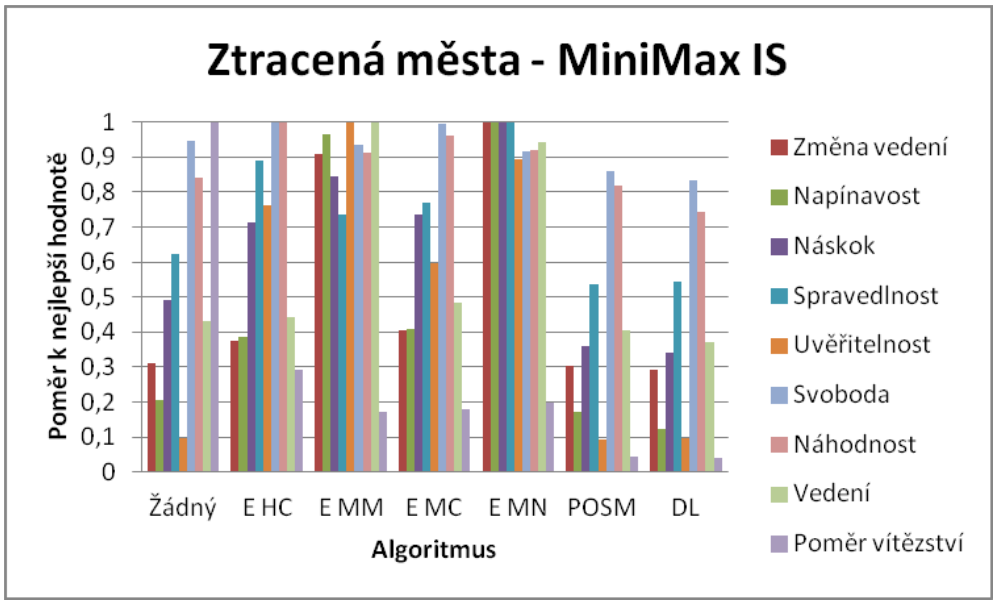
Obrázek 24 popisek



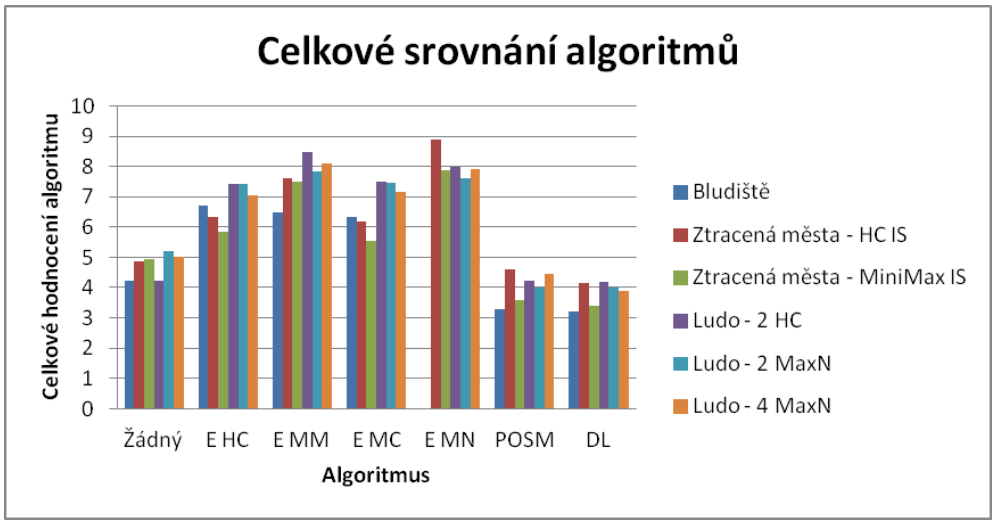
Obrázek 25 popisek



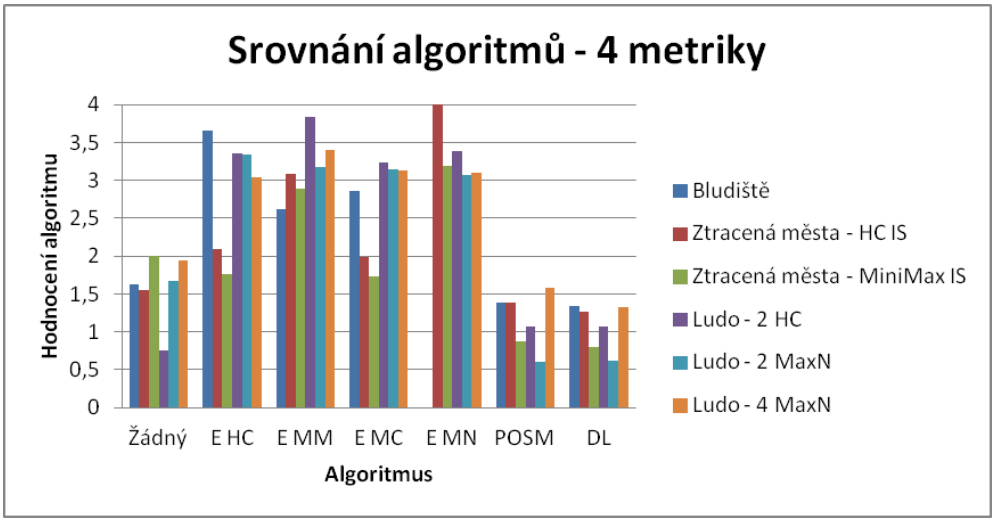
Obrázek 26 popisek



Obrázek 27 popisek



Obrázek 28 popisek



Obrázek 29 popisek

7 Závěr

Ahoj

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] 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.

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

[4] 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.

[5] 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.

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

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

[8] 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.

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

[10] 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).

[11] 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).

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

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

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

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

[16] IMDB.com. *Stargate SG-1: Season 8, Episode 6*. URL: http://www.imdb.com/title/tt0709042/ (cit. 16. 02. 2013).

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

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

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

[20] N. Elangovan. *Education in Holland*. URL: http://www.pages.drexel.edu/~ne34/edu.html (cit. 16. 02. 2013).

[21] M.I. Chowdhury a M. Katchabaw. "Software design patterns for enabling auto dynamic difficulty in video games". In: *Computer Games (CGAMES), 2012 17th International Conference on*. 30 2012-aug. 1 2012, s. 76–80. DOI: 10.1109/CGames.2012.6314555.

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

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

[24] 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).

[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] 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.

[28] 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).

[29] 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.

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

[31] 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.

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

[33] 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.

[34] 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.

[35] 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.

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