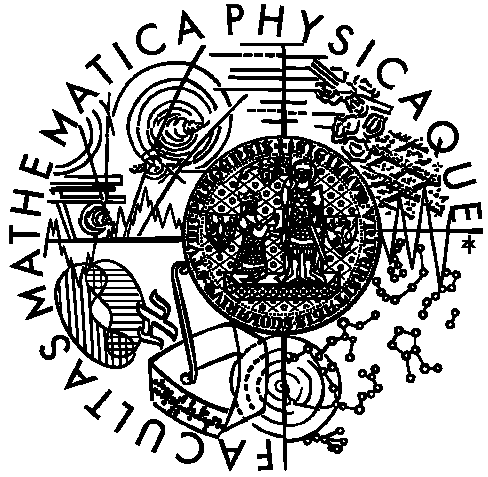
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Rudolf Kadlec

Adaptivní agenti a emoce

Katedra softwarového inženýrství

Vedoucí bakalářské práce: Mgr. Cyril Brom
Studijní program: Informatika, Obecná informatika

2006

Rád bych na tomto místě poděkoval Mgr. Cyrilu Bromovi za trpělivé vedení mé práce. Tato práce byla částečně podpořena grantem GA UK 351/2006/A-INF/MFF.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 23. května 2006 Rudolf Kadlec

Obsah

Název práce: Adaptivní agenti a emoce
Autor: Rudolf Kadlec
Katedra: Katedra softwarového inženýrství
Vedoucí bakalářské práce: Mgr. Cyril Brom
e-mail vedoucího: brom@ksvi.mff.cuni.cz

Abstrakt: Tato práce zkoumá možný přínos emocí pro autonomní adaptivní agenty pracující v prostředích podobných skutečnému světu. Emoce u živých organismů vznikly jako jeden z mechanismů adaptace na okolní prostředí a je proto na místě klást si otázku zda by se jim podobné mechanismy nedaly přenést i do modelů autonomních agentů. V rámci práce byl implementován model etologicky inspirovaného agenta používajícího algoritmus zpětnovazebního učení. Emoce v něm ovlivňují vyvážení mezi průzkumem nových strategií a využíváním strategií již známých (tzv. „explore/exploit problem“). Negativní emocionální hodnocení současného počínání vede ke změně strategie výběru akcí. V některých typech prostředí dosahovala emocionální varianta agenta lepších výsledků než neemocionální, v jiných se ale očekávání nepotvrdila. Nestabilita výsledků je pravděpodobně zapříčiněna neoptimální parametrizací celého modelu.

Klíčová slova: adaptivní agenti, zpětnovazební učení, emoce, ALife

Title: Adaptive agents and emotions
Author: Rudolf Kadlec
Department: Department of Software Engineering
Supervisor: Mgr. Cyril Brom
Supervisor's e-mail address: brom@ksvi.mff.cuni.cz

Abstrakt: This thesis investigates possible assets of emotions for autonomous adaptive agents working in environments similar to the real world. In living organisms, emotions have developed as a mechanism of adaptation to the surrounding environment. Therefore it is worth asking whether mechanisms similar to emotions can be implemented in models of autonomous agents. In this thesis a model of ethology inspired agent using reinforcement learning was implemented. This model suggests that emotions influence the balance between exploring new strategies and exploiting the strategies already known (the so-called explore/exploit problem). Negative emotional evaluation leads to changes in action selection strategy. The emotional version proved to be better than the non-emotional one in some environments. In other types of environments, the expectations have not been fulfilled. The instability of the received results is probably caused by non-optimal parameterization of the whole model.

Keywords: adaptive agents, reinforcement learning, emotions, ALife

Kapitola 1

1. Úvod

Ve sci-fi knihách se často setkáváme se stroji obdařenými „lidskou“ inteligencí. HAL 9000 z Clarkovy Vesmírné odysey 2001 ovládal lidskou řeč, měl svoji osobnost a dokonce i pud sebezáchovy. Rok 2001 již minul a lidstvo nejenom, že nelétá k Jupiteru, ale není ani o moc blíže k sestrojení skutečné umělé inteligence, než bylo v době napsání Vesmírné odysey.

Optimismus poloviny minulého století je zapomenut. Žádná revoluce se nekonala a vývoj postupuje spíše opatrnými krůčky evoluce. Za uplynulá desetiletí se sice podařilo vyřešit řadu problémů, ale ještě více se jich mezi tím objevilo. Například šachy se již staly doménou počítačů, stroj DeepBlue je schopen sehrát minimálně vyrovnanou partii i s velmistry [14]. Ve hře GO se naproti tomu počítače chovají jako začátečníci.

Podobné hry jsou však pro stroje rozhodně přívětivějším prostředím, než skutečný svět. Chůze do schodů nebo řízení automobilu jsou úkoly, které hravě zvládne každý člověk, ale pro robotické systémy je to maximum současných možností.

V čem může být příčina toho, že počítače mají takové problémy s úkoly, které jsou pro nás běžné? Především lidský mozek sice neprovede sto milionů operací v plovoucí řádové čárce za sekundu, ale svojí složitostí je pro současné a na dlouhou dobu i pro budoucí počítače naprosto nedostižný. Několik miliard let evoluce jej dokonale přizpůsobilo našemu životnímu prostředí a problémům, kterým musíme čelit. „Evoluce“ počítačů je z tohoto pohledu na úplném začátku svojí cesty. Teprve se hledají vhodné algoritmy pro řešení problémů a datové struktury pro jejich reprezentaci.

Umělá inteligence se v této oblasti již několikrát s úspěchem přírodou inspirovala a vypůjčila si některé mechanismy, které evolucí vznikaly tak dlouho. Umělé neuronové sítě nebo i samotná evoluce v podobě genetických algoritmů se staly řešením pro řadu problémů. Podobným nástrojem by se mohly v budoucnu stát i mechanismy podobné emocím. Emoce pomáhají živým organismům orientovat se v jejich proměnlivém světě a mohly by si proto najít cestu i do architektur autonomních agentů.

V rámci této práce byl navržen model agenta používajícího emoce pro zlepšení výkonu zpětnovazebního učení. Zároveň bylo vytvořeno simulované prostředí, ve kterém byl model agenta otestován. V příloze naleznete CD s oběma programy a jejich dokumentací.

1.1 Vlastní přínos

Při konstrukci adaptivních agentů je jedním ze základních problémů vyvážení mezi využíváním již nalezených strategií a průzkumem nových strategií („explore/exploit problem“). Hlavním cílem agenta není bezezbytku odhalit

mechanismus fungování prostředí, agentovi stačí znát jen tolik, aby mohl úspěšně plnit zadané úkoly. Učení je pro agenta prostředkem, nikoli cílem.

Cílem této práce je prozkoumat možnou úlohu emocí jako mechanismu, který rozhoduje, kdy se vyplatí prozkoumávat nové a kdy využívat již nalezené strategie. Užitečnost této role emocí byla testována na modelu umělé myši s názvem Algernon[1]. Algernon žije v simulovaném prostředí, které poznává pomocí algoritmu zpětnovazebního učení. Emoce hodnotí úspěšnost jejího počínání a přímo ovlivňují mechanismus výběru akcí.

V rámci práce jsem:

- Vytvořil simulované prostředí a jeho uživatelské rozhraní (viz kapitola 2).
- Navrhl architekturu agenta Algernon (viz kapitola 3). Především jsem vytvořil vlastní algoritmus učení a na něj navazující mechanismus výběru nejvhodnější akce zohledňující současný emocionální stav agenta.
- Otestoval užitečnost emocí. Hodnocenou veličinou byla doba dožití agenta. Testováno bylo několik neemocionálních verzí a jedna emocionální verze (viz kapitola 4).

S explore/exploit problémem se setkáváme i u algoritmů strojového učení. Úkolem algoritmů strojového učení je pochopit celý svět, projít všechny jeho stavy a vytvořit jeho úplný model. K tomu se často používají různé pravděpodobnostní metody, jako například simulované žíhání u neuronových sítí [16] nebo greedy algoritmus pro q-learning [12]. Jejich společným rysem je, že na začátku více prozkoumávají a ke konci už jen využívají.

Tento přístup však nemusí být pro agenty žijící v nehostinných prostředích vhodný. Fáze průzkumu může být příliš dlouhá. Agent musí brzy po začátku simulace začít uspokojovat své potřeby a nemá čas na podrobný průzkum. Idea použití emocí pro řešení explore/exploit problému je takováto: agent se pokusí uspokojit svoje potřeby s tou znalostí, kterou již má a uvidí, zdali byl skutečně úspěšný – jeho úspěšnost hodnotí (podobně jako u živých organismů) právě emoce. Pokud bude neúspěšný, prožívané emoce budou negativní a to agenta přiměje k hledání nových strategií. Rozdíl mezi navrhovaným přístupem k explore/exploit problému a přístupem běžným u strojového učení naznačuje obrázek Fig. 1.

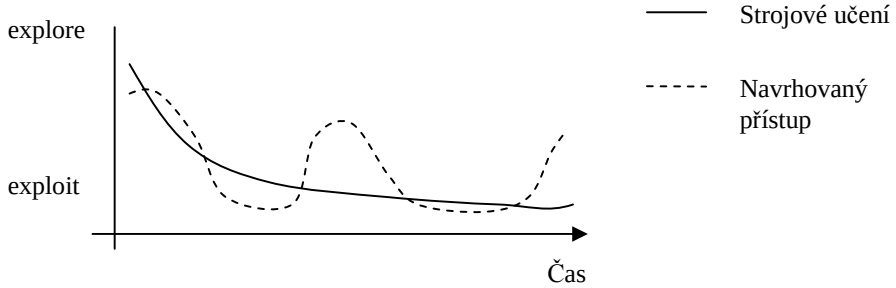


Fig. 1 Rozdíl v přístupu k explore/exploit problému

[1] Jméno Algernon jsem si pro svého agenta vypůjčil z divadelní hry D. Keyese: Růže pro Algernon. V ní se objevuje laboratorní myš Algernon, které se po podání pokusného séra zázračně zvýší inteligence.

1.2 Inspirace

V minulosti vznikla řada počítačových modelů organismů inspirovaných poznatky z etologie. Můj model vychází zejména z prací Tobyho Tyrrella a Bruce Blumberga a navíc přidává emoce jako nástroj pro zvýšení efektivity agentova chování.

Tyrrell ve své práci [13] provedl srovnávací studii několika modelů z oblasti etologie i robotiky. Měřítkem pro výkonnost agentů byl počet spáření za dobu jejich života. Pokusy se odehrávaly v nehostinném prostředí, které na agenty kladlo mnoho protichůdných požadavků. Hlavními úkoly bylo vyhledávání si jídla, pití, vyhýbání se nebezpečí, udržování čistoty, tělesné teploty a hledání partnera. Tyrrell na základě zkušeností s implementovanými modely navrhl svůj vlastní model, který nakonec ve sledovaném kriteriu všechny ostatní předčil.

Blumberg se nezabýval pouze etologií. Jeho práce [2] má přesah i do oblasti animace, počítačových her a storytellingu. Blumberg navrhl framework pro vytváření umělých postaviček a s jeho pomocí jich několik implementoval. Blumbergův model je na rozdíl od Tyrrellových modelů adaptivní. Agent používá formu zpětnovazebního učení a umí se naučit provádět stará chování v nových kontextech. Simulovaný pes Silas je například schopen naučit se, že dostane odměnu pokaždé, když se na uživatelův příkaz překulí na záda.

Tyrrellova práce mě ovlivnila při návrhu architektury Algernon. Blumbergův framework zase inspiroval Algernonin učící algoritmus.

1.3 Plán práce

V následující podkapitole zavedeme některé pojmy často používané ve zbytku práce. V další podkapitole se zamyslíme nad možným významem emocí pro autonomní agenty obecně.

V kapitole Simulované prostředí představíme architekturu prostředí použitého pro simulaci agenta. Následovat bude kapitola s podrobným popisem Algernon, jejího učícího algoritmu, mechanismu výběru akcí, emocionálního modelu a vlivu emocí na agentovo chování. Předposlední kapitola bude pojednávat o testech Algernon v několika různých nastaveních. Poslední kapitola přinese postřehy pro budoucí práci a shrne dosažené výsledky.

1.4 Terminologie

Nyní následuje seznam pojmů, které jsou v této práci často zmiňovány.

- **Autonomní agent** – je počítačový systém, fungující v nějakém prostředí, schopný samostatně provádět akce tak, aby splnil úkoly, ke kterým byl zkonstruován. Tuto definici přejímám z [17]. Takovýmto agentem může být

robot, postava z počítačové hry nebo simulovaná myš. V textu budou často autonomní agenti často označování jen jako agenti.

- **Interní stimuly** – poskytují agentovi informaci o jeho vnitřním stavu. Příkladem interního stimulu může být žízeň, tedy nízká hladina vody v organismu.
- **Externí stimuly** – vyjadřují možnosti, které agentovi skýtá prostředí. Příkladem externího stimulu je blízkost potravy, zdroje vody nebo predátora.
- **Akce** – pomocí akcí agent interaguje se svým prostředím. Každá akce přinese agentovi nějakou odměnu (např. energii). Akce můžeme dělit na ty, které odměnu přinesou okamžitě („consummatory actions“) a ty, které slibují odměnu v budoucnu, ale samy ji přímo nepřinášejí („appetitive actions“). Příkladem prvního typu může být akce „sněz jablko“, druhého pak „jdi k jablku“.
- **Mechanismus výběru akcí** („Action selection mechanism“ – ASM) – je postup, podle kterého agent na základě svého vnitřního stavu a stavu okolního prostředí vybere nejvhodnější akci.
- **Emoce** – vymezení pojmu emoce bývá často ovlivněno předmětem zkoumání a občas vede k nejasnostem. Pro nás budou emoce vyjadřovat subjektivní vztah k předmětům a jevům objektivní reality i k sobě samému, který je prožíván v široké škále pocitů od výrazně kladných, přes neutrální, až k výrazně záporným pocitům. Tuto definici přejímám z [7].

1.5 Další možné použití emocí pro agenty

Tato kapitola se zamýšlí nad dalšími možnostmi použití emocí pro autonomní agenty. Učení není jediným problémem, který by emoce mohly pomoci řešit. Abychom mohli posoudit užitečnost emocí i v jiných oblastech zavedeme si zjednodušený a velmi obecný model agenta a podíváme se z jakých funkčních bloků se skládá. Poté se inspirujeme funkcí emocí v živých organismech a navrhneme jejich analogické uplatnění v našem modelu.

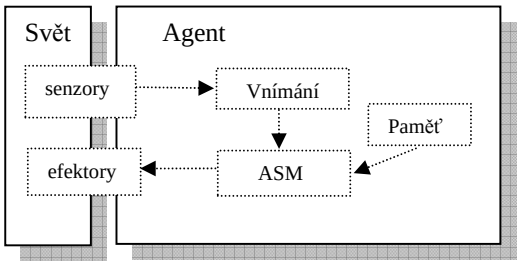


Fig. 2 Typická architektura agenta

Obecnou architekturu agenta a jeho pracovní cyklus ilustruje Fig. 2. Agent nejdříve zjistí stav prostředí pomocí *senzorů*. Proces *vnímání* přeloží údaje ze senzorů do reprezentace, která je vhodnější pro další rozhodování (např. nalezení určitých objektů na obrázku z kamery). Na základě těchto údajů a faktů z paměti pak *ASM* vybere

nejvhodnější akci, kterou agent provede svými *efektory*. Učící se agent navíc vyhodnotí efekt provedených akcí a podle toho upraví paměťové záznamy nebo další parametry svého modelu.

Nyní probereme, v čem a jak by mohly emoce na jednotlivých úrovních (vnímání, ASM, paměť a učení) výše popsaného modelu agenta pomoci. Jde spíše o nástin možností, nikoli o kompletní výčet.

Usměrnění vnímání a uvažování. U lidí je usměrnění vnímání a uvažování patrné zejména na vlivu strachu a hněvu. Tyto emoce způsobují, že některé podněty z okolí nevědomě přehlížíme, zatímco jiným věnujeme veškerou naši pozornost. Tento stav bývá označován jako tzv. „tunelové vidění“. Zároveň se zužuje i náš akční repertoár.

Pro agenta mohou emoce plnit funkci filtru, který například v situaci vyvolávající strach potlačí nepodstatné podněty a umožní tím věnovat více výpočetní kapacity na hledání únikových cest nebo zdrojů potenciálního nebezpečí. Jednoduchý emoční filtr, který mění důležitost objektů podle nálady, se objevuje například v práci [5].

Rychlost ASM. U lidí je emoční zhodnocení nastalé situace velmi rychlé a předchází vyšším kognitivním procesům. To umožňuje téměř okamžitou reakci na základě obecných rysů situace. Pozdější vědomé zhodnocení pak většinou naše jednání určitým způsobem koriguje.

Pro agenty může být tento mechanismus užitečný v prostředích, kde je důležitá rychlost reakce. Agent si v situaci ohrožení nemůže dovolit dopodrobna rozpracovávat plán úniku; musí zareagovat okamžitě. Pokud se bude cítit ohrožen, může okamžitě začít provádět obecný a tudíž jen přibližný plán úniku (pohybuj se ve směru od nebezpečí) a teprve potom jej začít zpřesňovat (najdi dveře a ukryj se za ně).

Vztah emocí a paměti. Experimenty potvrzují, že si s větší pravděpodobností vybavíme událost, kterou jsme si zapamatovali ve stavu odpovídajícím našemu současnému rozpoložení. Tento jev bývá označován jako *efekt kongruence s náladou*. O jeho vysvětlení se pokusil například Bower (dle Berkowitze [1]). Bower vidí paměť jako síť různých uzlů. Některé uzly odpovídají předmětům reálného světa (např. hrnek), jiné jsou svázány s emocemi (strach, štěstí atd.). Spoje mezi uzly vyjadřují, s jakou pravděpodobností si vybavíme předmět nebo emoci uzlu, pokud bude aktivní sousední uzel. Při souhlasné aktivaci dvou uzlů se spojení mezi nimi posílí. Jde vlastně o asociativní síť, ve které vystupují i emoce.

S ohledem na agenty vede existence tohoto jevu k myšlence využití emocí jako kontextu pro vyhledávání v paměti. Uzel pro hrnek může být aktivován tím, že agent právě vidí hrnek stát na stole. Pokud bude mít zároveň strach, spojení mezi uzly „hrnek“ a „strach“ se posílí. Aktivace se spojením může přelévat v obou směrech. Příště emoce strachu zvýší šanci, že si agent vzpomene na hrnek a naopak vizuální vjem hrnku u něj může vyvolat strach. V prvním případě emoce pomohou vybavit si vzpomínky získané v situaci, která byla alespoň v nějakém aspektu (zde stejném emocionálním stavu) podobná té současné. Agent si pod vlivem strachu začne vybavovat co všechno zažil, když měl strach minule a třeba mu to pomůže nalézt řešení jeho situace. V druhém případě vnější situace ovlivní emoce agenta a tím i jeho chování.

Učení a propojování znalostí. Mimo řešení explore/exploit problému se emocionální zhodnocení dá použít i jako zdroj odměny při zpětnovazebním učení (viz projekt CMatie [8]). To platí zejména pro prostředí, kde efekt akcí není známý.

Vztah emocí k učení může být i složitější. Pozitivní emoce pomáhají propojit nově získanou zkušenost se starou zkušeností, přispívají k nalézání souvislostí. Negativní emoce tuto vlastnost nemají, zkušenost pak často není zasazena do širšího kontextu. Tím se stane nedostupnou pro uvažování a bude zapomenuta [10] (str. 123).

Ve světě agentů by tedy emoce mohly plnit funkci heuristiky, která nám řekne, kolik času se vyplatí věnovat zhodnocení nové zkušenosti.

Výše popsané přístupy předpokládaly přítomnost nějakého emočního generátoru, modulu, který na základě vnitřního stavu agenta a informací o stavu prostředí určí emocionální zhodnocení situace. Oprávněnost existence takovéhoto centra je podpořena neurologickými nálezy, které potvrzující důležitou roli amygdaly[2] v emočním prožívání.

V informatice se objevuje ale i opačný přístup, který emoce chápe jako emergentní jevy složitého systému [4]. Například Minsky ve své koncepci mysli jako společenství specializovaných výpočetních jednotek („cloud of resources“) chápe emoce jako vyjádření stavu těchto jednotek (rádců) [9]. Emoce podle něj odpovídají aktivaci různých rádců. Pokud o agentovi řekneme, že je rozezlený, není to podle Minského proto, že by někde v jeho modelu byl nastaven přepínač hněvu na „zapnuto“, ale proto, že je aktivní určitá podmnožina rádců a zbylí rádci jsou potlačeni.

Výše popsané možnosti začlenění emocí do architektur autonomních agentů jsou v současné době spíše ve stadiu teoretických úvah a implementací pokusných modelů. Jejich praktické aplikace se zatím zdají být vzdáleny.

[2] Amygdala – dvě mandlovitá ganglia ve spodní části předního mozku.

Kapitola 2

2. Simulované prostředí

Cílem této práce je vytvoření adaptivního, etologicky motivovaného agenta využívajícího emocí. Prostředí by proto mělo postihovat základní charakteristiky skutečného světa. Rozlišit zásadní vlastnosti prostředí od těch méně podstatných je však nesnadný úkol často ovlivněný subjektivním rozhodováním. Mnohé vlastnosti skutečného světa budeme muset opomenout a je otázkou, nakolik to naši simulaci znehodnotí.

Pro potřeby této práce bylo implementováno velmi obecné prostředí, jehož hlavním úkolem je zprostředkování interakce mezi agenty a objekty. Vlastnosti prostředí jako je přívětivost nebo naopak hostilita, determinismus nebo nedeterminismus jsou určeny objekty přítomnými v mapě. Chování objektů použitých pro testování agenta je deterministické. Objekty se dají snadno vytvářet jako samostatné moduly a následně mohou být přidány do mapy prostředí.

Grafickým rozhraním prostředí je aplikace EnvCentre. EnvCentre umožňuje prohlížení mapy, její editaci a sledování činnosti agentů. Tato kapitola pojednává o vlastnostech prostředí implementovaného v aplikaci EnvCentre.

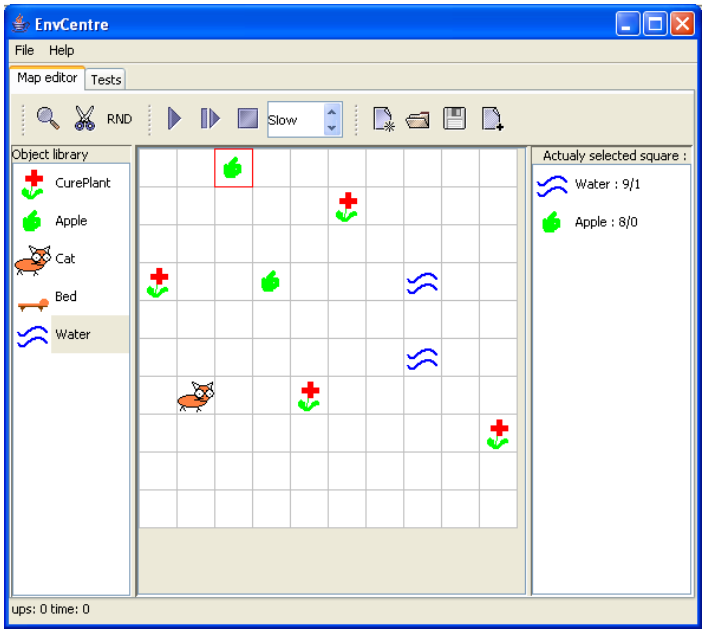


Fig. 3 EnvCenre - grafické rozhranní implementovaného prostředí

2.1 Architektura prostředí

Základní charakteristiky prostředí jsou:

- Šachovnicová mapa – objekty jsou umístěny na políčcích šachovnice.
- Diskrétní čas – všechny změny ve světě se odehrávají v pevně daných kolech. Simulace se nesnaží napodobovat spojitý čas.
- Pevně definovaná množina akcí – agenti mohou na objektech v prostředí vykonat jednu z šesti definovaných akcí.
- Jednotná forma odměny z prostředí – agent dostane od prostředí po provedení akce odměnu, ta má pro všechny akce stejnou strukturu a vyjadřuje změnu stavu agentových vnitřních proměnných. Vnitřní proměnné a jejich význam je popsán níže.

V prostředí vystupují dva základní typy entit: objekty a agenti. Agenti jsou „vylepšené“ objekty, které mohou vnímat okolí a provádět akce na objektech ve své blízkosti. Příkladem objektu může být jablko nebo rostlina, agentem může být např. myš nebo kočka.

Prostředí předpokládá, že všichni agenti mají podobný metabolismus. Výsledkem každé akce je totiž čtveřice čísel udávajících, jak se po jejím provedení změnily agentovy vnitřní proměnné. Vnitřními proměnnými agentů jsou:

- Energie – představuje hladinu cukru v krvi.
- Voda – představuje hladinu vody v organismu.
- Zdraví – odráží celkový stav organismu
- Únava – představuje bdělost organismu.

Akcemi, které mohou agenti provádět, jsou:

- *Jdi k*
- *Jdi od*
- *Sněz*
- *Napij se*
- *Zmáčkni*
- *Odpočiň si*

Akce se na objektech vyvolávají podobně jako metody na objektech z OOP jazyků. Jejich „návratovou hodnotou“ je odměna. Akce *jdi k* a *jdi od* navíc posunou agenta ve směru k resp. od objektu, na kterém byla akce vyvolána. Odměna, kterou akce vrací, pak už záleží na implementaci objektu.

Prostředí neklade žádné požadavky na agentovu logiku rozhodování. To znamená, že se tu vedle sebe mohou potkávat agenti s různou vnitřní architekturou. Mohou to být například čistě reaktivní agenti řízení sadou rozhodovacích pravidel, adaptivní agenti používající neuronové sítě nebo avataři ovládaní člověkem. Algernon je v současné době jediným složitějším implementovaným agentem.

2.2 Jedno kolo agentova života

Jednotkou simulačního času je jedno kolo. V průběhu každého kola všichni agenti obdrží aktualizovanou mapu svého okolí a v návaznosti na to provedou vybranou akci.

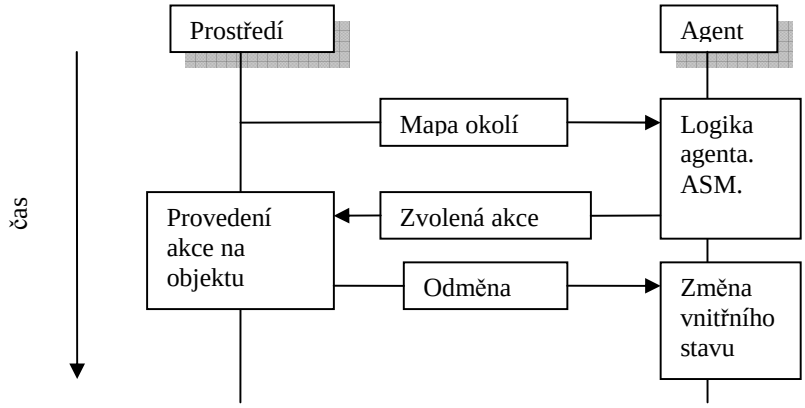


Fig. 4 Komunikace mezi agentem a prostředím

Fig. 4 ukazuje, jak vypadá jedno kolo agentova života. Pojďme si teď tento diagram popsat podrobněji.

Mapa okolí je šachovnice velikosti (2N+1)×(2N+1), kde N udává agentův dohled. Agent je přesně uprostřed této šachovnice. Na každém políčku šachovnice je seznam objektů, které se na něm momentálně nacházejí. Agent se může pokusit složit ve své vnitřní paměti celou mapu okolí (jak to dělá Algernon). To znamená, že si bude zapamatovávat všechna místa kterými prošel a postupně zvětšovat prozkoumanou plochu mapy. Na místech, která se mezitím dostala z jeho dohledu nemusí však vnitřní mapa odpovídat realitě. O dlaždicích ve svém dohledu má naproti tomu vždy přesnou informaci.

ASM („Action Selection Mechanism“) vybírá nejvhodnější akci k provedení. ASM se většinou rozhoduje podle stavu vnitřních proměnných agenta, podle faktů uložených v jeho paměti (pokud agent paměť má) a podle stavu okolního prostředí (např. blízkost jablka může zvýšit pravděpodobnost vybrání akce *sněz jablko*).

Zvolená akce je zpráva tvaru <akce, objekt>, která říká, jakou akci chce agent vyvolat na zvoleném objektu.

Odměna je čtveřice čísel udávající, o kolik se změnil agentův ukazatel energie, vody, únavy a zdraví. Když agent provede akci *sněz jablko*, odměna může vypadat například takto: energie +3, voda +1, únava +0.5, zdraví +1 (tyto hodnoty se většinou pohybují v intervalu od -3 do +3). Agent si tedy výrazně doplní zásoby energie, získá trochu vody, zlepší si zdraví a nepatrně se unaví.

Kapitola 3

3. Agent Algernon

V této kapitole bude detailně popsán výpočetní model agenta Algernon. Algernon je simulovaná myš, používající jednodimenzionální emoční model pro zvýšení efektivity svého chování. Emoce hodnotí Algernonino počínání ve spojité škále: pozitivní – negativní.

Po „narození“ do prostředí o něm Algernon neví téměř nic. Neví co má dělat pro to, by se uzdravila. Neví, které věci se mají jíst a které naopak pít. Všechny tyto informace musí získat až v průběhu svého života. Prostředí může v závislosti na přítomných objektech poskytovat různé cesty ke splnění Algernoniných cílů. Emoce ji pak pomáhají najít rovnováhu mezi používáním již objevených strategií a nalézáním nových. Pozitivní emoce ji utvrzují v současném postupu, pod vlivem negativních emocí Algernon naopak současnou strategii zavrhne a vyzkouší nějakou jinou. Toho se dosáhne změnou algoritmu výběru akcí.

Agent Algernon byl implementován ve stejnojmenné aplikaci. Její grafickou nadstavbou je program AlgernonApp (viz Fig. 5). Prostředím pro simulaci agenta je program EnvCentre (kap. 2).

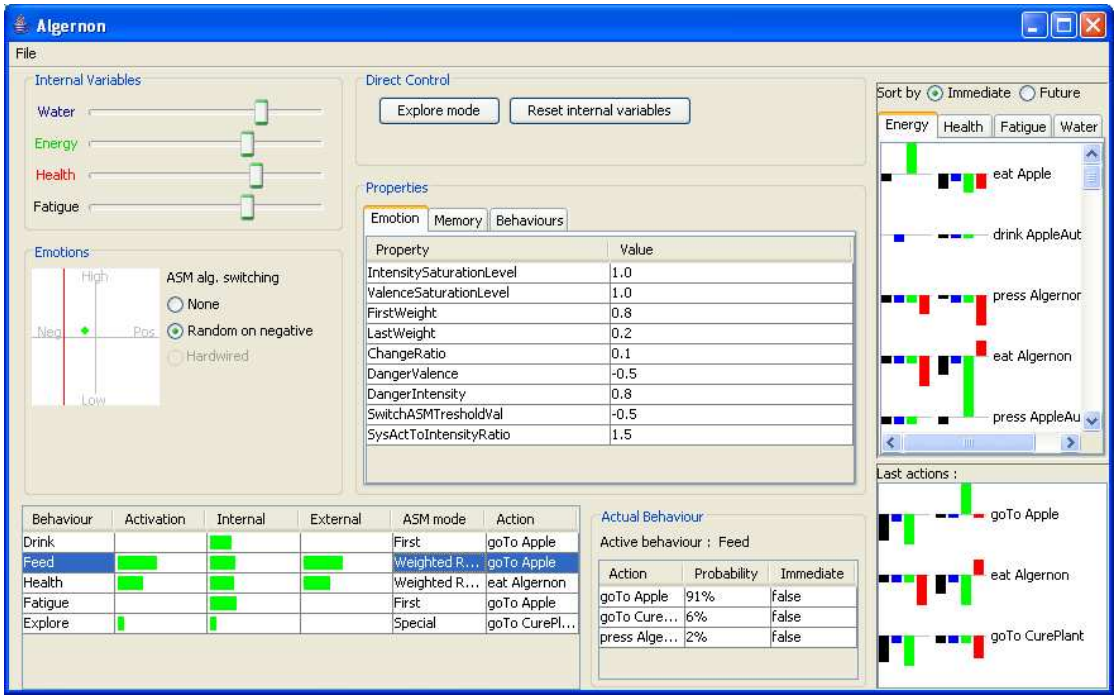


Fig. 5 AlgernonApp - prohlížeč vnitřního stavu Algernon

V této kapitole bude v hrubých rysech popsána architektura Algernon. Další kapitoly se pak budou podrobněji věnovat funkci jednotlivých modulů a jejich provázanosti.

3.1 Architektura Algernon

Architektura Algernon je naznačena v Fig. 6.

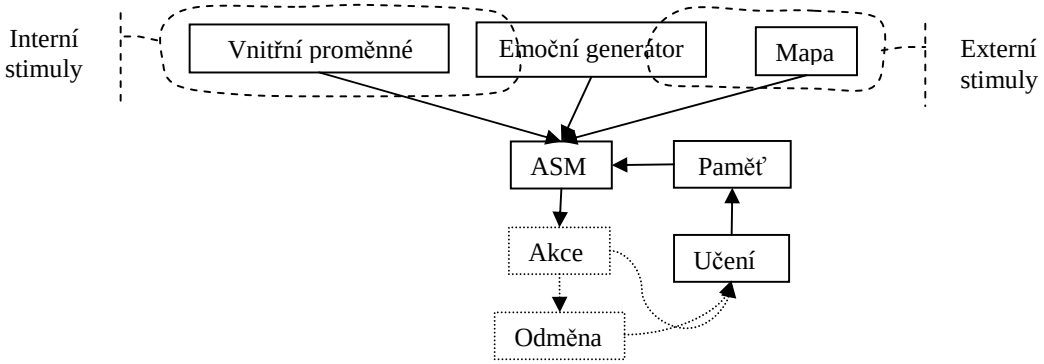


Fig. 6 Architektura Algernon

Ústředním modulem agenta je **ASM**. ASM se rozhoduje na základě informací pocházejících z různých zdrojů – stimulů. Stimuly můžeme v souladu s etologickou tradicí rozdělit na ty, které vycházejí z agentova těla – interní stimuly, a ty, které se vztahují k okolnímu prostředí – externí stimuly.

Interními stimuly jsou u Algernon hodnoty vnitřních proměnných modelu. **Vnitřní proměnné** odpovídají složkám odměny, kterou Algernon obdrží z prostředí po provedení akce. Jsou to tedy:

- Energie
- Voda
- Zdraví
- Únava

Externí stimuly vyjadřují příležitosti, které Algernon přináší současná konfigurace prostředí. Pokud Algernon uvidí objekt o kterém si myslí, že je zdrojem vydatné potravy, stane se tento pro ni významným externím stimulem. Zdrojem externích stimulů je u Algernon vnitřní **mapa**.

Funkce interních a externích stimulů a jejich konkrétní implementace je blíže popsána v kapitole 3.3.

Emoční generátor spadá částečně do obou kategorií. Emoce jsou ovlivňovány jak vnějšími okolnostmi, tak jejich subjektivním zhodnocením. Emoce ovlivňují způsob práce ASM, měly by agentovi pomoci dostávat se z lokálních maxim jeho chování a nalézat lepší strategie. Emočnímu generátoru se věnuje kapitola 3.8.

Algernon je adaptivní agent a proto v každém kole vyhodnocuje efekt svých akcí. U každé akce si pamatuje, jakou odměnu její provedení přinese bezprostředně a jakou odměnu může přinést v budoucnu. Například akce *sněz sýr* přinese okamžitou

odměnu v podobě zvýšení energie. Akce *jdi k sýru* okamžitě hladinu energie nezvýší (naopak povede k jejímu mírnému snížení, protože při pohybu se energie spotřebovává), ale její provedení přenese Algernon do stavu, kde už se možná hladina energie zvýší. Ke zvýšení hladiny energie dojde, pokud se Algernon dostane k sýru a sní ho.

Paměť obsahuje veškerou znalost, kterou má Algernon o prostředí. Základní jednotkou informace je jeden paměťový záznam, ten je tvaru <akce, objekt, okamžitá odměna, možná budoucí odměna>. Více je o paměti uvedeno v kapitole 3.4.

Stimuly pomáhají **ASM** určit chování, které je pro agenta momentálně nejdůležitější (které bude nejvýhodnější v současném stavu uspokojit), konkrétní akce se pak vybere podle paměťových záznamů. Výběr nejdůležitějšího chování je popsán v kapitole 3.3, kostře algoritmu pro výběr nejvýhodnější akce se věnuje kapitola 3.5, o modifikacích algoritmu pak pojednává kapitola 3.9.

Učení se stará o distribuci odměny na předešlé akce. Odměna, kterou Algernon získá provedením akce, se proporcionálně distribuuje až na pět posledně provedených akcí v podobě možné budoucí odměny. Nejstarší akce dostane nejmenší díl odměny, zatímco předposlední největší. To pak v součinnosti s algoritmem ASM umožňuje Algernon naučit se sekvence akcí typu „dojdi k pasti na myši“, „zneškodni ji“ a „sněz sýr“. Implementovanému algoritmu zpětnovazebního učení se věnuje kapitola 3.6.

3.2 Mechanismus výběru akcí

Celý problém vytváření autonomních agentů by se vlastně dal redukovat na problém ASM. Máme vstupy definované architekturou prostředí a ty chceme co nejlépe namapovat na množinu akcí, které umí agent provést. Výběr nejvhodnější akce je ale dosti netriviální záležitostí. Agent chce většinou uspokojit více cílů zároveň a ty mohou často klást i protichůdné nároky.

Základní poučkou programování je, že pokud se zdá být úloha příliš rozsáhlá dekomponujeme ji na řadu menších a ty vyřešíme samostatně. Podobný postup volí při studiu chování zvířat i etologie. Chování zvířete jako celek rozdělíme na řadu menších chování s jasně vymezeným účelem a zavedeme arbitrační mechanismus, který bude mezi nově zavedenými chováními přepínat. Pokud je chování aktivní, má jakoby kontrolu nad zvířetem a vybírá akce až do chvíle, než je mu řízení odebráno a aktivním se stane jiné chování. Neaktivní chování vyčkávají a monitorují situaci. Ve chvíli, kdy se situace zdá příhodná pro uspokojení jejich cílů, přihlásí se arbitrovi a počkají, zda jim ten přidělí řízení.

Algernon má pět základních chování, jsou to chování pro:

- Jídlo (Feed) – snaží se maximalizovat zisk energie
- Pití (Drink) – snaží se maximalizovat zisk vody
- Udržování zdraví (Health) – snaží se o uzdravení
- Odpočinek (Rest) – snaží se minimalizovat únavu
- Průzkum (Explore) – slouží pro průzkum prostředí

Nejdůležitější chování je vybráno na základě aktivace. Chování s největší aktivací vyhrává a může si zvolit akci o které se domnívá, že ho nejlépe uspokojí.

Aktivace je funkcí interních a externích stimulů příslušejících danému chování. Pro první čtyři chování (Feed, Drink, Health, Rest) jsou interními stimuly hodnoty příslušejících vnitřních proměnných (Energie, Voda, Zdraví, Únava). Externím stimulem je přítomnost objektu, který vystupuje v paměťovém záznamu majícím vysokou hodnotu příslušné složky odměny. Pokud bude mít Algernon v paměti záznam tvaru <sněz, jablko, +3 energie … , 0 energie …> a zároveň uvidí jablko nablízku, tak se toto jablko stane externím stimulem pro chování Feed.

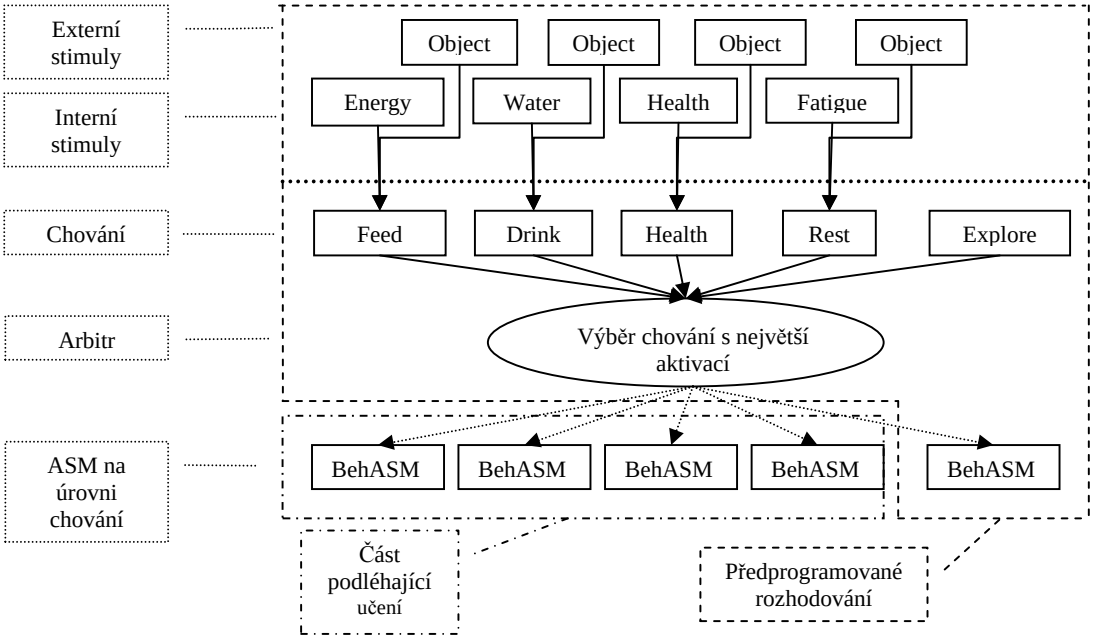


Fig. 7 Detailnější pohled na ASM

Výpočet aktivace chování a následný výběr nejdůležitějšího chování je předem naprogramován. Učení probíhá až na úrovni výběru akcí pro jednotlivá chování (BehASM – Behaviour ASM). Výjimkou je chování průzkumu, které je také předprogramované.

3.3 Výběr nejdůležitějšího chování

Nejdůležitější chování se vybírá pouze na základě aktivace. Výpočet aktivace jednotlivých chování je proto stěžejní pro správné fungování celého ASM. Model Algernon se inspiruje chováním skutečných zvířat. Výběr nejdůležitějšího chování by tedy měl připomínat rozhodování, jaké vidíme i u zvířat.

Představme si, že skutečná myš má zároveň žízeň i hlad, žízeň je však přece jen o něco akutnější a tak se vydá ke známému zdroji vody, aby se tam mohla napít. Cestou k vodě ale narazí na kousek potravy, v tu chvíli převáží potřeba hladu a tak se myš zastaví a sní nalezenou potravu. Teprve poté se znovu vydá za vodou.

Tento příklad ilustruje potřebu začlenění externích stimulů do procesu rozhodování o nejdůležitějším chování. Pokud by model neuvažoval externí stimuly, tak by Algernon jídlo minula a bez zastávky by pokračovala v cestě za vodou. K jídlu by se možná vrátila až poté, co by se napila. Aktivace chování pro pití by klesla a jezení by se nejspíš stalo nejdůležitějším chováním. Algernon by vykonala jednu cestu navíc. Úspora energie, kterou zavedení externích stimulů přináší, je zřejmá.

Funkce výpočtu aktivace není však prostou sumací interních a externích stimulů, zavádí i další členy, jejichž význam nyní bude rozebrán. Aktivace se spočte předpisem:

$$external = possibleGain * ExtWeight \qquad \textbf{Eq. 1}$$

$$internal = |varOpt - var| * VarWeight + var' * DerivWeight \qquad \textbf{Eq. 2}$$

$$energy = internal + external \qquad \textbf{Eq. 3}$$

$$activation = \max(0, energy - StopTreshold) \quad \text{pokud je chování preaktivní}$$

$$activation = 0 \quad \text{pokud není preaktivní} \qquad \textbf{Eq. 4}$$

Chování je preaktivní pokud někdy v minulosti jeho *energie* byla nad prahem *StartTreshold* a od té doby ani jednou neklesla pod *StopTreshold*. To, že chování je pracující ještě neznamená, že je vítězem a vybírá další akci. Znamená to jen, že může mít nenulovou aktivaci a tudíž může soutěž vyhrát. Energie vyjádřená rovnicí Eq. 3 je součtem interních (Eq. 2) a externích (Eq. 1) stimulů. Hodnota *internal* je vážený součet vzdálenosti vnitřní proměnné příslušející tomuto chování (var) od její optimální hodnoty (*varOpt*) a derivace té samé vnitřní proměnné v čase (*var'*). Hodnotou externích stimulů (*external*) je součin váhy (*ExtWeight*) a čísla udávajícího očekávanou změnu vnitřní proměnné (*possibleGain*), pokud bude toto chování moci provést svoji akci. Tato hodnota závisí na akci, kterou chce chování provést. Její přesný význam bude popsán v kapitole o algoritmu BehASM.

Chování bude mít po většinu času nulovou aktivaci, avšak ve chvíli, kdy začne být potřeba jeho uspokojení akutní (energie přeroste mez *StartTreshold*), vstoupí chování do soutěže o kontrolu nad Algernon. Počáteční hodnota aktivace bude navíc poměrně veliká, to dá chování možnost vyhrát v soutěži vícekrát za sebou a tak lépe uspokojit své potřeby. Tím se model vyhýbá stavům, kdy je zároveň aktivních více chování a každé z nich vyhraje třeba jen na jedno nebo dvě kola. Akce, kterou si pak zvolí, nevede většinou k okamžitému získání odměny (jsou to akce typu „jdi k …“) a proto by Algernon mohla vyhladovět přešlapováním na místě. Model se dvěma prahy (*Start* a *Stop*) tímto nedostatkem netrpí. Hodnoty *StopTreshold, StartTreshold, VarWeight, DerivWeight* a *ExtWeight* se dají pro každé chování zvlášť nastavit v aplikaci AlgernonApp.

Fig. 8 ukazuje, jak zpravidla vypadá průběh aktivace a energie chování.

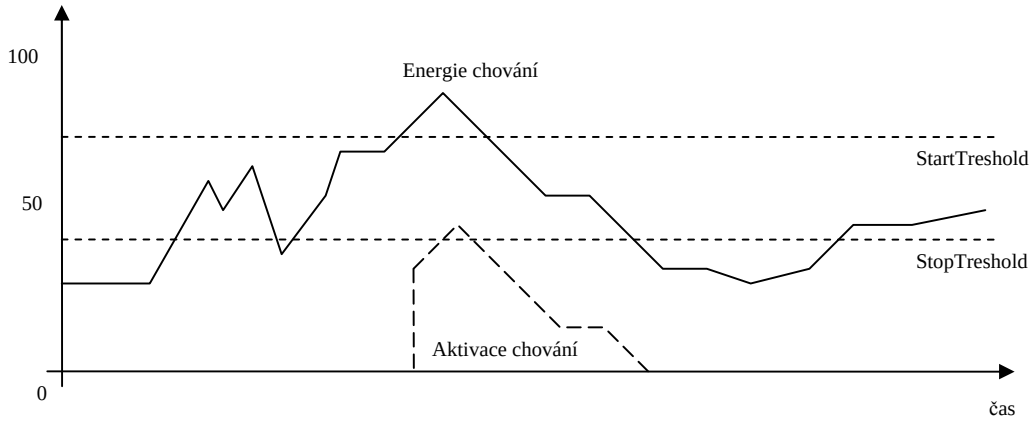


Fig. 8 Závislost energie chování a jeho aktivace

3.4 Paměť

Paměť je kolekcí paměťových záznamů. Jeden paměťový záznam je nejmenší jednotkou informace, kterou si Algernon může o prostředí uchovat. Všechny záznamy mají stejnou strukturu a jsou uzpůsobeny pro potřeby ASM. Požadavkům ASM odpovídá i způsob uložení jednotlivých záznamů, ten je optimalizován na rychlost vyhledávání.

akce	objekt	okamžitá odměna	možná budoucí odměna

Fig. 9 Šablona paměťových záznamů

jdi k	jablko	E: -1 H: 0	W: 0 F: -1	E: +3 H: +1	W: +0.5 F: -0.5
sněz	jablko	E: +4 H: +1	W: +1 F: -1	E: -0.5 H: 0	W: -0.5 F: +0.5

E – Energy
W – Water
H – Health
F – Fatigue

Fig. 10 Dva příklady paměťových záznamů

Fig. 10 ukazuje příklady dvou záznamů. První příklad se vztahuje k činnosti „jdi k jablku“, druhý k „sněz jablko“.

Položky *okamžitá odměna* a *možná budoucí* odměna jsou měněny v procesu učení.

3.5 BehASM

Tato kapitola popisuje základní variantu algoritmu pro výběr nejvhodnější akce. Modifikacemi tohoto algoritmu se zabývá kapitola 3.9.

Ve chvíli, kdy chování vyhraje, je na něm, aby zvolilo akci k provedení. Tento proces je ve Fig. 7. označen jako BehASM (Behaviour ASM). Nejvhodnější akce se

vybírá na základě paměťových záznamů. U chování Feed, Drink, Health a Rest algoritmus BehASM prochází paměťové záznamy seřazené podle jejich očekávaného užitku a vybere první, který se dá v současném stavu provést. Podrobněji to ukazuje pseudokód algoritmu:

```
1.  for (mi in itemsForBehaviourImmediate) {
2.      if (mi.okamžitá_odměna_ve_složce < 0) then break;
3.      if ((mi.akce se dá provést na malou vzdálenost and na posici agenta se
        nalézá objekt stejného typu jako je mi.objekt)
4.      or
5.      (mi.akce se dá provést na velkou vzdálenost and v mapě se nalézá objekt
        stejného typu jako je mi.objekt))
6.          then proveď akci doporučovanou v mi;
7.  }
8.  for (mi in itemsForBehaviourFuture) {
9.      if ((mi.akce se dá provést na malou vzdálenost and na posici agenta se
        nalézá objekt stejného typu jako je mi.objekt)
10.     or
11.     (mi.akce se dá provést na velkou vzdálenost and v mapě se nalézá objekt
        stejného typu jako je mi.objekt))
12.         then proveď akci doporučovanou v mi;
13. }
14. Pokud nebyla nalezena žádná vhodná akce, předej řízení chování pro průzkum;
```

Fig. 11 Kód algoritmu BehASM

Kde *itemsForBehaviourImmediate* je seznam paměťových záznamů seřazených sestupně podle složky okamžité odměny příslušné tomuto chování a *itemsForBehaviourFuture* je seznam seřazený podle možné budoucí odměny v dané složce.

Podmínka na řádku 2. říká, že za účelem získání okamžité odměny bude Algernon provádět jen akce, které současný stav ještě nezhorší. Akce, které se dají provést na malou vzdálenost jsou, „sněz“, „vypij“, „zmáčkni“ a „odpočiň si“, na velkou vzdálenost to jsou „jdi k“ a „jdi od“.

Algoritmus ASM zvýhodňuje akce s okamžitou odměnou na úkor těch, které mohou odměnu přinést v budoucnu. Pokud je nalezena proveditelná akce s okamžitým přínosem, je provedena a ostatní paměťové záznamy se už ani neprohledávají. Seřazení akcí zaručuje, že bude vybrána ta s největší odměnou.

Fungování ASM na úrovni jednoho chování osvětlí následující příklad. Dejme tomu, že má Algernon v paměti záznamy tvaru:

Akce	Objekt	Okamžitá odměna				Budoucí odměna			
		E	W	H	F	E	W	H	F
Jdi k	Automat na jablka	-1	0	0	-1	**1**	0.5	0.3	0
Zmáčkni	Automat na jablka	0	0	0	0	**1.5**	1	0.6	-0.7
Sněz	Jablko	**4**	1	1	-1	-0.9	0	-0.2	0

Fig. 12 Paměťové záznamy pro pracování ASM

Algernon si zvolila za nejdůležitější chování jídlo – chce tedy zvýšit hladinu energie. Při výběru akcí se nadále bude řídit podle energie, kterou akce přinese. Na začátku stojí Algernon tři políčka od automatu na jablka, žádné jiné objekty v mapě nejsou. Algernon tedy třikrát zvolí akci „jdi k automatu na jablka“, protože si myslí, že jí to v budoucnu zvýší energii. Žádnou jinou akci v tomto stavu ani provést nemůže. Jakmile ale dojde na políčko s automatem, bude si už vybírat ze dvou akcí: „jdi k automatu na jablka“ a „zmáčkni automat na jablka“. Algernon si vybere druhou akci, protože ta jí slibuje vyšší odměnu. V tu chvíli z automatu vypadne jablko a objeví se na stejném políčku jako je automat i Algernon. Teď si bude vybírat ze tří akcí. Ke dvěma z minulého kola přibude akce „sněz jablko“, ta se dá v současné konfiguraci provést také. Jablko je na stejném políčku jako Algernon a akce „sněz“ je proveditelná na malou vzdálenost. Příslušný paměťový záznam bude přijat již v první smyčce algoritmu BehASM. Algernon sní jablko a tím částečně uspokojí potřeby aktivního chování. Energie z interních stimulů poklesne a nejdůležitějším chováním se možná stane nějaké jiné. To, aby algoritmus skutečně proběhl tímto způsobem, závisí na hodnotách v paměťových záznamech. Pokud by například odměna za akci „zmáčkni automat na jablka“ byla menší než u „jdi k automatu na jablka“, Algernon by první akci nikdy nezvolila a stále by prováděla jen akci „jdi k automatu na jablka“. Je záležitostí algoritmu učení, aby záznamy postupem času zkonvergovaly do stavu, který umožní korektní zřetězení akcí (jdi k → zmáčkni → sněz).

Z popisu algoritmu vyplývá, že Algernon ví, zdali si akci zvolila pro její bezprostřední efekt („consummatory action“) nebo proto, že věří, že jí akce možná přinese odměnu v budoucnu („appetitive action“). Paměťové záznamy akcí prvního typu jsou přijaty v první smyčce průchodu algoritmu, paměťové záznamy druhého typu pak v druhé smyčce.

Výše popsaným algoritmem se řídí výběr akcí pro chování Feed, Drink, Health a Rest. Chování Explore má svůj vlastní algoritmus ASM:

1. vyber náhodný objekt v blízkém okolí
2. dojdi k němu (prováděj akci „jdi k objektu“ dokud nebude na stejné pozici jako ty)
3. prováděj na něm různé permutace akcí
4. po nějakém čase přejdi k bodu 1.

Fig. 13 Kód algoritmu ASM pro průzkum

Algoritmus BehASM je velmi jednoduchý. Díky tomu je však i snadno pochopitelný a kontrolovatelný člověkem. Při sledování Algernon v jejím prostředí můžeme snadno odhadnout, proč si danou akci vybrala.

Nevýhodou algoritmu BehASM je, že při výběru akce zohledňuje pouze jedno chování. To je však předurčeno již architekturou Algernon, která každému chování přisuzuje vlastní algoritmus BehASM. Takto navržený algoritmus například neumí záměrně zvolit akci uspokojující více chování – tzv. kompromisní akci. Tyrrell došel k závěru, že modely schopné volit kompromisní akce jsou výhodnější [13].

3.6 Učení

Úkolem algoritmu učení je nastavit hodnoty v paměťových záznamech tak, aby ASM mohlo provádět smysluplné řetězce akcí. Algernon za tímto účelem používá formu zpětnovazebního učení („reinforcement learning“). Zpětnovazební učení je technika inspirovaná učením zvířat [11]. Agent provede akci a za ni dostane od prostředí odměnu. Ne vždy však odměna přijde okamžitě, někdy je nutné k dosažení cíle provést celou sekvenci akcí. Aby takovéto sekvence mohly vznikat, je získaná odměna distribuována i na předešlé akce.

V této kapitole bude nejdříve popsán algoritmus učení použitý u Algernon a na závěr bude konfrontován s nejznámějším zástupcem algoritmů zpětnovazebního učení - s Q-učením („Q-learning“). Q-učení nebylo použito z důvodu své pomalé konvergence.

Hlavní funkcí učení je distribuovat odměnu získanou v aktuálním kole na akce provedené v krátké minulosti, popřípadě upravit okamžitou odměnu u paměťového záznamu posledně provedené akce pokud byla jiná než skutečně získaná odměna.

Modul učení je v každém kole informován o provedené akci a změně vnitřních proměnných Algernon (viz Fig. 6). Změna vnitřních proměnných (Δ = *new* - *old)* je prohlášena za odměnu, kterou provedení akce přineslo. To je samozřejmě zjednodušující předpoklad. Představme si, že Algernon zvolila akci „sněz jablko“ a při tom jí zároveň kousla kočka. Pokles zdraví potom bude asociován s akcí „sněz jablko“. Algernon se od tohoto okamžiku bude vyhýbat jedení jablek, i když správná interpretace situace by byla, aby se napříště vyhýbala kočkám. To je způsobeno jednoduchostí zvoleného algoritmu (viz porovnání s Q-učením).

Na základě informace o akci a její odměně pak učení upraví paměťové záznamy až pěti posledně provedených akcí. Algoritmus učení tedy provede tyto kroky:

1. uprav okamžitou odměnu u paměťového záznamu právě provedené akce;
2. **for** (*action* **in** *lastActions*) uprav budoucí odměnu v paměťovém záznamu pro *action*;

Fig. 14 Algoritmus učení

V prvním kroku se okamžitá odměna upraví podle vzorce:

$$r_{new} = r_{old} + (r_{old} - a) * \beta \qquad \textbf{Eq. 5}$$

Kde r_{new} a r_{old} jsou nová a stará hodnota okamžité odměny, a odměna získaná v tomto kole a β učící konstanta ($0 \prec \beta \prec 1$). Parametr β se dá nastavit v programu AlgernonApp a najde se pod názvem *ImmediateLearningRate*. Hodnota budoucí odměny se nemění.

V druhém kroku se upravuje hodnota možné budoucí odměny u nedávno provedených akcí.

V seznamu *lastActions* je každá posledně provedená akce uvedena pouze jednou. Při vícenásobném přidání se ponechá jen záznam o nejnovějším provedení akce (starý se

odebere a nový se přidá na začátek seznamu). Tím se předchází problémům s vícenásobným přidělením budoucí odměny té samé akci. Právě provedená akce v seznamu *lastActions* zaznamenána ještě není.

Budoucí odměna v paměťovém záznamu příslušejícímu akci ze seznamu *lastActions* se upraví podle rovnice:

$$r_{new} = r_{old} + (r_{old} - a) * \alpha^{age} \qquad \textbf{Eq. 6}$$

Kde r_{new} je nová hodnota budoucí odměny, r_{old} je stará hodnota budoucí odměny, a odměna získaná v tomto kole, α učící konstanta $(0 \prec \alpha \prec 1)$ a age doba, měřená v krocích simulace, před kterou byla akce provedena. Parametr α se dá nastavit v programu AlgernonApp, tam je pojmenován jako *FutureLearningRate*. Odměny r_{new} a r_{old} jsou vektory o čtyřech složkách (energy, water, health, fatigue). Hodnota okamžité odměny se u těchto akcí v paměťovém záznamu nemění.

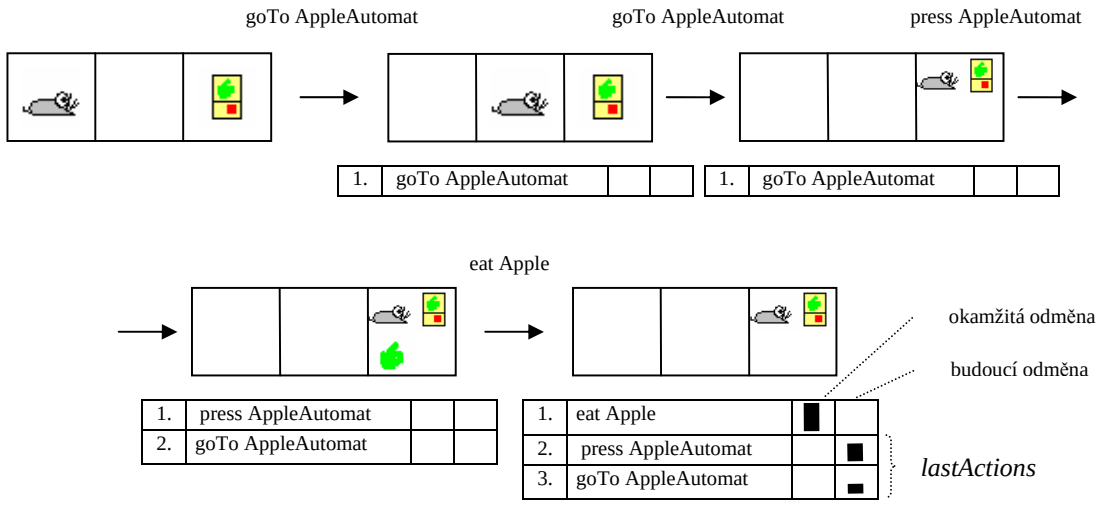


Fig. 15 Příklad práce algoritmu učení

Fig. 15 ukazuje ideální průběh učícího algoritmu. Ve skutečnosti ale i akce „jdi k“ a „zmáčkni“ přinesou okamžitou odměnu: úbytek energie a zvýšení únavy. V tomto příkladě je však ukázána jen distribuce odměny z akce „sněz jablko“ na „zmáčkni automat na jablka“ a „jdi k automatu na jablka“. Tento mechanismus distribuce odměny spolu s ASM umožní v budoucnu celý řetězec akcí zopakovat.

Výše uvedený příklad ukázal, že pokud Algernon provede smysluplnou sekvenci akcí, bude schopna si ji „zapamatovat“ a v budoucnu znovu přehrát. V tomto případě se objevení správné sekvence povedlo na první pokus.

To však není typický postup učení u Algernon. Algernon prozkoumává své okolí systémem pokus/omyl („trial/error“). Chování Explore generuje různé permutace akcí na objektech a my doufáme, že se tímto postupem podaří objevit dobrou sekvenci akcí a že

se tuto podaří zapamatovat. Algernon se sekvenci z příkladu spíše naučí tak, že se nejdřív setká s jablkem a zapamatuje si, že je dobré ho sníst. Pak v rámci průzkumu náhodně provede akci „zmáčkni automat na jablka“. V tuto chvíli je docela pravděpodobné, že aktivace chování Feed překročí práh *StartTreshold*, jablko je pro něj totiž silným externím stimulem. Pokud se Algernon skutečně rozhodne sníst jablko, odměna se přenese i na předešlé akce a sekvence bude nejspíš správně zapamatována.

Učení je iterativním procesem, pouze po mnoha opakováních paměťové záznamy pravděpodobně zkonvergují do podoby vyhovující algoritmu ASM.

Problémem tohoto přístupu je, pokud akce dostane příliš vysokou budoucí odměnu, i když ji ve skutečnosti nikdy nepřinese. Co se stane, když Algernon provede sekvenci „jdi k jablku“ → „zmáčkni jablko“ → „napij se z tůně“? Až Algernon v budoucnu dostane žízeň, tak dojde k jablku a v případě, že bude na políčku s jablkem i tůň, tak se z ní napije („napij se z tůně“ přináší okamžitou odměnu a proto dostane přednost před „zmáčkni jablko“). Pokud ale na políčku tůň nebude, Algernon by mohla provádět akci „zmáčkni jablko“ až do vyhladovění. Žádoucí je, aby po nějakém čase Algernon „přišla“ na to, že tato akce nevede k požadovanému cíli a zkusila jinou akci, třeba „napij se z tůně“. Jak ale určit hodnotu „timeoutu“ akce? Ve skutečnosti totiž timeout není konstanta ale obecně funkce. Například to, jak dlouho budeme provádět akci „jdi k“ záleží na vzdálenosti objektu, ke kterému jdeme.

Tento problém je ale pro učení příliš složitý (Blumbergův model psa [2] se například učí jen konstantní timeout – pes zjistí za jak dlouho po překulení se na záda ho pán podrbe, pokud v tomto intervalu odměna nepřijde začne vykonávat jinou činnost). Proto v modelu Algernon timeout jako parametr vůbec nevystupuje. Místo toho se od složky budoucí odměny, odpovídající právě prováděnému chování, odečte v každém kole malé kladné číslo. Takže když je akce prováděna po nějaký čas a stále nepřináší odměnu, pravděpodobně v seznamu akcí seřazených podle budoucí odměny klesne pod jinou akci a ta bude vybrána místo ní. Pokud ale skutečně povede k úspěchu, její hodnota budoucí odměny se opět zvedne. Z toho plyne, že paměťové záznamy obsahující správnou informaci se používáním neznehodnocují. Naopak ty zavádějící jsou odhaleny a příště již použity nebudou.

3.7 Srovnání s Q-učením

Algoritmus zpětnovazebního učení použitý u Algernon připomíná známý algoritmus Q-učení [15], ale ten nebyl použit hlavně kvůli své pomalé konvergenci. Na druhou stranu umožňuje odhalit mnohem složitější závislosti a je univerzálnější než Algernonin algoritmus. Q-učení se dá naučit strategie pro piškvorky, řízení automobilu v počítačové hře nebo navigace robota.

Algernonin algoritmus pracuje se záznamy tvaru <akce, objekt, odměna, odměna>. Q-učení si ke každé akci navíc pamatuje i stav, ve kterém byla provedena (ten může být rozsáhlejší než pouhý objekt, jako v prvním případě), pracuje ale jen s jedním typem odměny (i když ta může být vícerozměrná). Záznam je pak tvaru <akce, stav, odměna>. Jiná je i rovnice distribuce odměny:

$$Q(a, s_{old}) = Q(a, s_{old}) + \beta\left(q + \gamma \max\{Q(a', s_{new}) \mid a' \in actions\} - Q(a, s_{old}) \right) \qquad \textbf{Eq. 7}$$

Kde a je akce z množiny agentem proveditelných akcí *actions*, která ho převedla ze stavu s_{old} do stavu s_{new} a q odměna, kterou při tom získal. $Q(a, s)$ je funkce přiřazující každému stavu a akci očekávanou odměnu (dá se vyjádřit tabulkou), β je učící konstanta ovlivňující rychlost distribuce odměny, γ je konstanta rozptylu. Pokud bude γ nízká agent bude preferovat akce přinášející odměnu co nejdříve, zatímco vysoká hodnota favorizuje akce se vzdálenou a tudíž potencionálně nejistou odměnou. Nově spočtenou očekávanou odměnu tedy ovlivňuje: stará znalost ($Q(a, s_{old})$), získaná odměna (q) a odměna, kterou může agent získat v budoucnu, pokud zvolí nejlepší akci ($\max\{Q(a', s_{new}) \mid a' \in actions\}$).

Agent pak v každém kroku vybírá akci s největší očekávanou odměnou pro svůj současný stav.

Otázkou je, co všechno zařadit do stavu *s*. Součástí stavu může být například mapa bezprostředního okolí, hodnoty vnitřních stimulů a podobně. Tím však roste dimenze stavu a zároveň i velikost tabulky pro funkci *Q*. Velikost tabulky můžeme redukovat jejím zobecňováním genetickými algoritmy nebo můžeme zkusit funkci *Q* aproximovat neuronovou sítí.

Shrňme si hlavní rozdíly mezi oběma algoritmy:

- Q-učení uvažuje i stav agenta, ten Algernon nemá.
- Q-učení má jeden typ odměny, Algernon dva, a proto ví, jestli dělá akci pro její bezprostřední nebo budoucí užitek.
- Q-učení se při určování odměny jakoby dívá do budoucnosti (člen $\max\{Q(a', s_{new}) \mid a' \in actions\}$) zatímco Algernon uvažuje jen minulost.

Největší slabinou Algernonina algoritmu učení je, že je schopen naučit se jen prostou posloupnost akcí (např. „jdi k“ → „zmáčkni“ → „sněz“). Sekvenci akcí typu A → B → C → B, kde až druhé provedení akce B přinese odměnu, se algoritmus nenaučí. Pokud Algernon provede první tři akce v tomto pořadí, bude to znamenat, že očekávaná odměna u akce C je větší než u B, C bude proto mít před B přednost. Agent řízený Q-učením provede takovouto sekvenci za předpokladu, že se mezi dvěma provedeními akce B změní jeho stav, tedy že druhou akci B již bude provádět v jiném kontextu. To se může stát, pokud akce C nějak změní agentův vnitřní stav, například se rozsvítí světlo a agent bude vybaven senzory pro detekci světla. Druhou možností je vybavit agenta pamětí několika posledně provedených akcí a tuto začlenit do jeho vnitřního stavu. Pokud by si agent pamatoval i posledně provedenou akci, tak se již od sebe obě provedení akce B odliší (prvnímu provedení akce B předcházela akce A, druhému C). Tím se však opět zvětší velikost stavu *s* a tudíž i tabulky pro funkci *Q*.

Hlavním důvodem pro implementaci vlastního algoritmu byla potřeba velmi rychlé konvergence učení. Algernon přežívá v prostředí jen několik desítek až stovek kol a proto se musí sekvence učit co nejrychleji.

3.8 Emoce

Emocím je u živých organismů přisuzováno mnoho rozdílných funkcí a tomu odpovídá i jejich rozmanité použití v počítačových simulacích organismů jak je zmíněno v kapitole 1.5.

Algernon používá emoce jako heuristiku pro odhalení slepých uliček ve svém chování. Emocionální model odhaluje ty periody v jejím životě, kdy je frustrovaná neúspěchem v plnění svých potřeb. V takovém případě Algernon pozmění algoritmus výběru akcí a bude doufat, že jí to pomůže dostat se z nějakého „lokálního maxima“ jejího chování do lepšího maxima, nejlépe do toho „globálního“.

Implementace emočního modelu se inspiruje poznatky soudobé psychologie. V současnosti jsou patrné dvě hlavní tendence ve studiu emocí [10]. *Diskrétní přístup* popisuje vlastnosti jednotlivých funkčně oddělených (diskrétních) emocí. Emoce dělí na primární a sekundární. Primární chápe jako vrozené a většinou mezi ně řadí štěstí, hněv, smutek, strach, znechucení a zahanbení. Sekundární emoce pak vznikají směsicí primárních emocí a mohou být naučené, příklady jsou žárlivost, láska nebo závist.

Naproti tomu *dimensionální přístup* pracuje s emocionálním prostorem definovaným pomocí různých dimenzí. Nejčastějšími dimenzemi jsou intenzita (aktivovaný – neaktivovaný) a valence (příjemný – nepříjemný). Emocionální stav je pak charakterizován polohou v tomto emocionálním prostoru. Diskrétní emoce můžeme chápat jako určité oblasti v emocionálním prostoru.

Ani jeden z výše zmíněných přístupů není universálním řešením. Jsou to rozdílné pohledy na věc, které se spíše doplňují.

Algernonin model vychází z dimenzionálního přístupu. Dvěmi dimenzemi jsou výše zmíněná valence a intenzita. Model se snaží postihnout základní charakteristiky skutečných emocí. Jsou to:

- Saturace – obě dimenze mají svoje maximální a minimální hodnoty. Existuje mez příjemnosti i intensity, kterou současný emocionální stav nemůže překročit.
- Habituace – neboli zvykání si na podněty, pokud bude Algernon dlouho dostávat samé positivní impulsy zvykne si a bude to považovat za normální stav.

Výsledný emocionální stav je rozdílem mezi emočním pozadím (vážený součet podnětů za poslední období) a aktuálními podněty. Parametry modelu jsou váha nejnovějšího podnětu (v AlgernonApp označeno jako *FirstWeight)* a nejstaršího podnětu (*LastWeight*), váhy podnětů, které časově spadají mezi tyto dva jsou dopočítány lineárně podle jejich stáří. Všechny váhy jsou pak znormalizovány tak, aby jejich suma byla 1. Pokud tedy bude delší dobu přicházet ten samý podnět, bude mu vážený součet posledních podnětů roven (Algernon se na něj habituuje) a výsledná emoce bude neutrální (valence=0, intensita=0). Výpočet současného emocionálního stavu popisuje rovnice Eq. 8.

$$s = a_0 - \sum_{i=1}^{n} a_i w_i \qquad \textbf{Eq. 8}$$

kde s je výsledný emocionální stav (dvojdimenzionální, dimenzemi jsou valence a intenzita), a_0 současný signál (vektor), a_i minulé signály (vektory) a w_i jejich dopočtené váhy (skaláry).

Zdrojů signálů může být více. Cílem současného modelu je hodnotit úspěšnost Algernonina počínání, signály proto pocházejí od právě aktivního chování. Valencí signálu od aktivního chování je vážený rozdíl mezi odměnou, kterou chování získalo od doby, kdy se naposledy poprvé dostalo k řízení Algernon až do současnosti a očekávanou odměnou. Vyjádřeno rovnicí:

$$v = (g - t * a) * w \qquad \textbf{Eq. 9}$$

kde v je valence signálu, g odměna, kterou chování získalo od doby, kdy se naposledy poprvé dostalo k řízení, t je doba, po kterou je chování aktivní, a průměrný očekávaný zisk za jedno kolo (*AverageGainIfActive*), w je váha (*EvalFactor*).

Pokud Algernon získá více odměny než předpokládala, bude v dobré náladě, pokud ne, valence současného emocionálního stavu bude negativní. Intensita signálu je vždy 0, v průběhu vývoje se ukázalo, že jedna dimenze, svým významem odpovídající valenci, stačí. Model je tedy připraven i pro uvažování intensity, signály od aktivního chování ji ale nepoužívají.

Emoční generátor vyjadřuje, jak Algernon hodnotí svoje současné počínání. Pokud je spokojena, není důvod ke změně strategie jejího chování. Když je ale hodnocení současného stavu negativní (valence je nižší než práh *SwitchASMTresholdVal* – v prohlížeči je znázorněn červenou čarou, viz Fig. 5), tak to Algernon bude považovat za známku neúspěchu současného přístupu ve výběru nejvhodnější akce a pokusí se jej změnit. To znamená, že náhodně vybere jednu ze tří modifikací algoritmu BehASM popsaného v kapitole 3.5. Jednotlivé modifikace popisuje následující kapitola.

3.9 Modifikace ASM

Základní algoritmus BehASM popsaný v kapitole 3.5 vždy vybere tu nejlepší proveditelnou akci (v programu je označen jako *First*). Hodí se pro stav, kdy už Algernon objevila optimální strategii přežití pro dané prostředí a teď už ji jen využívá. Nehodí se však pro fázi, kdy se již Algernon snaží plnit potřeby aktivního chování, ale nejlepší strategii ještě nezná nebo stará strategie v současném stavu nefunguje. Pro tyto účely byly implementovány ještě další dvě modifikace algoritmu popsaného ve Fig. 11.

První modifikací je ruleta (v programu označeno jako *Roulette*). V tomto algoritmu se vybírá ze tří nejlepších akcí, tak jak by je vybral původní algoritmus. Každá

akce má třetinovou šanci, že bude zvolena. Odpovídá to stavu, kdy Algernon chce vyzkoušet úplně nové řešení.

Druhou je vážená ruleta (*Weighted Roulette*). Vybírá také mezi třemi nejlepšími akcemi, ale jejich pravděpodobnost je úměrná očekávanému užitku.

$$p_i = \frac{u_i + \varepsilon - \min_{k \in A}(u_k)}{\sum_{j \in A} u_j + |A|\left(\varepsilon - \min_{k \in A}(u_k)\right)} \qquad \textbf{Eq. 10}$$

Kde p_i je pravděpodobnost akce *i*, u_i její očekávaný užitek (okamžitý nebo budoucí, to záleží na zamýšleném použití akce – neboli na tom, v jaké smyčce algoritmu BehASM byla akce přijata), *A* množina tří zvažovaných akcí ($|A| = 3$) a ε je konstanta zajišťující nenulovou pravděpodobnost třetí akci (*ProbEpsilon*).

Tento mechanismus je kompromisem mezi ruletou a výběrem nejlepší možné akce. Nejlepší akce většinou vyhraje, ale i druhá a třetí mají šanci.

Kapitola 4

4. Testy

Pro zhodnocení přínosu emocí byly provedeny testy jedné emocionální a tří neemocionálních verzí Algernon. Emocionální verze (*Emotional*) používala algoritmus náhodného přepínaní algoritmů BehASM popsaný v kapitole 3.9. Neemocionální verze používaly jednu pevnou verzi algoritmu BehASM po celou dobu života (verze *First*, *Weighted* a *Roulette*).

Úspěšnost byla hodnocena počtem kol, které Algernon v prostředí přežila. Všechna prostředí, ve kterých byla Algernon testována, obsahovala tyto objekty:

- Jablko – snězení zvýší energii
- Vodu – vypití přidá vodu
- Uzdravující květina – snězení Algernon uzdraví
- Postel – odpočinek na posteli sníží únavu
- Automat na jablka – po zmáčknutí z něj vypadne jablko

Byly provedeny dvě série testů, v první byly testovány nenaučené instance Algernon, ve druhé instance s předběžnou znalostí o prostředí. V rámci každé série byla Algernon testována ve dvou rozdílně inicializovaných prostředích. To znamená, že prostředí obsahovala stejný počet objektů jednotlivých typů, ale posice objektů byly inicializovány náhodně a velikost mapy byla vždy 10×10. Jednotlivé varianty byly v každém prostředí spuštěny 25krát. Testovanou hypotézou je, že emoce zlepšují Algernoninu výkonnost, tedy dobu dožití.

V pracovních testech byla v prostředí přítomna i kočka. Ta se náhodně pohybovala po mapě a pokud uviděla Algernon, tak k ní došla a začala na ní provádět akci „sněz“, což vedlo k poklesu Algernonina zdraví a k její následné smrti. Algernon se měla naučit, že akce „jdi od kočky“ ji uchrání od úbytku zdraví. To se jí ale nikdy nepodařilo a Algernon byla vždy snědena kočkou. V testech prezentovaných v této práci tedy prostředí neobsahovalo ani jednu kočku, a Algernon měla zakázáno používat akci „jdi od kočky“, ta odstraněním kočky ztratila smysl.

4.1 Nenaučená Algernon

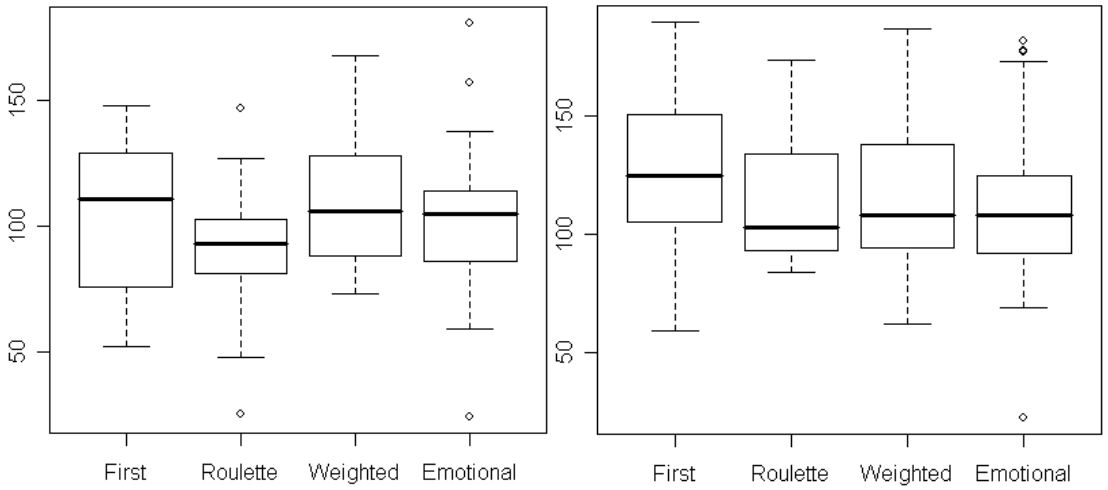


Fig. 16 Výsledky nenaučené Algernon

V prvním prostředí se jedné instanci Algernon s algoritmem *First* podařilo dosáhnout nesmrtelnosti. Naučila se používat všechny objekty a skončila v nekonečné smyčce.

V obou prostředích si nejlépe vedly algoritmy *First* a *Weighted*, což není překvapivý výsledek. Oba dva totiž většinou navrhnou stejnou akci. Emocemi řízené přepínání si vedlo relativně dobře v prvním prostředí, v druhém již nikoliv. *Roulette* byl nejhorší v prvním prostředí, ve druhém si vedl podobně jako emocemi řízené přepínání algoritmů.

Subjektivní pozorování průběhu těchto testů odhalilo, že Algernon se většinou naučila uspokojovat jeden až dva typy chování, ale při pokusu o uspokojení zbylých chování selhala a nepomohlo jí v tom ani emocionální přepínání mezi algoritmy výběru akce.

I v jiných inicializacích prostředích si všechny algoritmy vedly obdobně.

Emoce v případě nenaučené Algernon svoji užitečnost neprokázaly.

4.2 Naučená Algernon

Další série testů v jiné dvojici prostředí porovnávala již naučené instance agenta. Algernon tedy měla povědomí o funkci všech objektů, ale její paměť obsahovala i některé zavádějící informace.

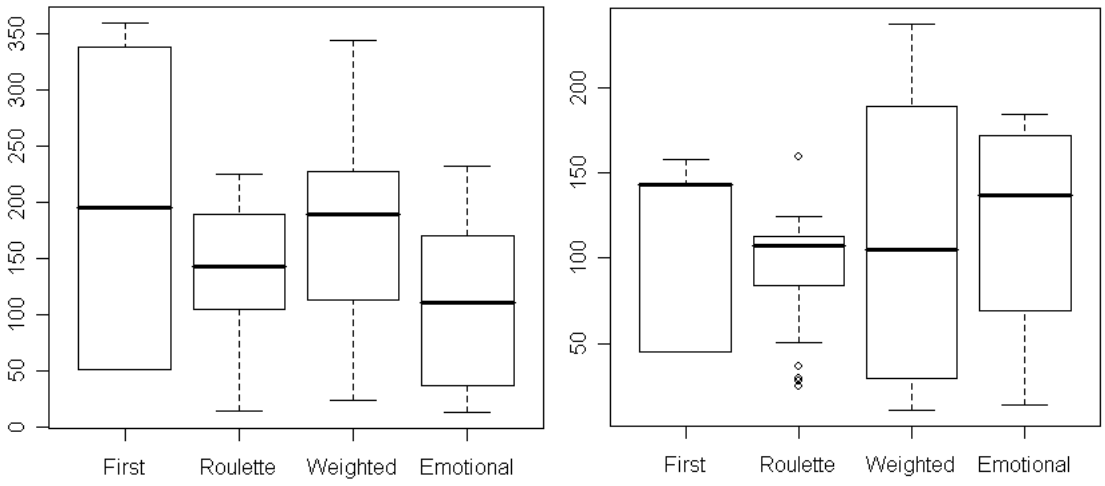


Fig. 17 Výsledky naučené Algernon

První prostředí se ukázalo být o mnoho přívětivější než druhé. V prvním prostředí mnoho jedinců žilo déle jak dvě stě kol, což se v druhém kole povedlo pouze jednomu jedinému. Rovněž je vidět, že emocemi řízené přepínání si v obou prostředích vedlo diametrálně odlišně. V prvním bylo nejhorší, zatímco v druhém dosáhlo nejvyšší průměrné délky života. Nejdéle žijící jedinci sice používali algoritmus *Weighted*, ale ten měl zároveň i největší rozptyl ze všech. Překvapivé je, že v těchto prostředích se ani jedna instance nedostala do stabilního stavu, kde by mohla přežít věčně.

Relativně dobré výsledky emocemi řízeného přepínání mohou být vysvětleny nehostinností druhého prostředí (na nehostinnost tohoto prostředí usuzuji podle špatných výsledků ostatních algoritmů). Algernon byla nucena čelit obtížím častěji než v prvním prostředí a emoce jí občas pomohly změnit způsob jejího chování k lepšímu.

I přes slibnost některých výsledků testy jednoznačně nevypovídají o přínosu současného emocionálního modelu pro zvýšení efektivity Algernonina chování.

Všechny soubory pro opětovné provedení testů jsou na přiloženém CD v adresáři s příklady. První prostředí je v souboru *rand1.env*, druhé v *rand2.env*. Jednotlivé verze naučené Algernon jsou v souborech *learnedFirst.alg*, *learnedWeighted.alg*, *learnedRoulette.alg* a *learnedEmotional.alg*.

.

Kapitola 5

5. Závěr

V rámci této práce byl implementován model adaptivního agenta s názvem Algernon. Jeho učení probíhá na úrovni koordinace akcí v rámci jednotlivých chování (low-level řízení), zatímco koordinace samotných chování je předem naprogramovaná (high-level řízení). Pro otestování agenta bylo vyvinuto prostředí s uživatelsky přívětivým rozhraním.

Algernon disponuje emocionálním modelem, který hodnotí úspěšnost v uspokojování potřeb jednotlivých chování. V případě, že je dosavadní postup vyhodnocen jako neuspokojivý, Algernon pozmění algoritmus ASM což jí může pomoci v nalezení nové, lepší strategie.

Experimenty ukázaly, že v některých prostředích emoce pomohly zvýšit efektivitu Algernonina chování (ta byla měřena délkou života), v jiných ale emocionální varianta dopadla hůře než všechny neemocionální. To poukazuje na problém nestability současného modelu způsobený zřejmě jeho nedostatečně kvalitní parametrizací.

Domnívám se, že i přes nejednoznačné výsledky současných experimentů zůstává myšlenka použití emocí jako mechanismu pomáhajícího opuštění lokálních maxim chování stále podmětnou.

5.1 Získané zkušenosti a náměty na budoucí práci

V průběhu své více jak rok trvající práce na Algernon a jejím simulovaném prostředí jsem získal mnoho neocenitelných zkušeností s programováním autonomních agentů. Narazil jsem také na několik skutečností, které tento druh práce odlišují od programování jiných aplikací.

Především fáze parametrizace agenta zabere mnohem více času než samotné jeho programování. Typický cyklus práce je: opravit kód agenta, zkompilovat, spustit jej v prostředí, dostat jej do požadované situace a sledovat jak se zachová tentokrát.

Do modelu agenta také často přibývají nové parametry, některé se po čase ukáží být jako neužitečné a zase jsou odebrány. Jelikož je ale žádoucí umět upravovat parametry modelu přímo za chodu, tak programátor musí pro každý parametr v grafickém rozhraní agenta vytvořit ovládací prvek a ten propojit s daným parametrem. Jde ale o opakující se práci, která člověka odvádí od jeho cíle – rychlého vytvoření fungujícího modelu agenta. Při řešení tohoto problému se ukázalo být užitečné Reflection API jazyka Java. To mimo jiné umožňuje za běhu získat seznam všech datových položek instance objektu a také měnit jejich hodnotu. Vytvořil jsem si tedy programovací konvenci, že všechny proměnné odpovídající parametrům modelu budou uvozeny prefixem *prop*. Za pomocí Reflection API jsem si pak vytvořil komponentu

JavaBean, která umožňuje za běhu měnit všechny takovéto proměnné zadaného objektu. To znamená, že programátorovi stačí napsat do kódu proměnnou podle příslušné konvence a ta se automaticky zobrazí v uživatelském rozhraní při příštím spuštění. Tato jednoduchá komponenta mi ušetřila mnoho monotónní práce.

Současný model Algernon má okolo padesáti parametrů, to znamená, že nalezení jejich optimálního nastavení metodou „pokus omyl“ téměř není v lidských silách. Pro nalezení jejich optimálního nastavení se nabízejí genetické algoritmy. S pomocí genetických algoritmů by mohlo být dosaženo stabilnější výkonnosti modelu a vliv emocí (ať již pozitivní nebo negativní) by mohl být prokázán mnohem přesvědčivěji.

Při vytváření agenta je také důležitá volba jeho životního prostředí, tedy simulátoru agentova světa. Na začátku své práce jsem žádné jiné prostředí neznal a proto jsem začal implementovat vlastní prostředí podle svých představ. Mezitím jsem se seznámil se zavedenými programy Framstick [6] a Breve [3], které poskytují i podporu pro vytváření agentů (neuronové sítě, genetické algoritmy). Tyto prostředí ale simulují trojrozměrný spojitý svět, což pro řízení agenta přináší řadu nových problémů.

Před případnou další prací v tomto oboru by bylo vhodné zauvažovat o návrhu univerzálního IDE pro tvorbu a testování agentů. O existenci podobného softwaru totiž doposud nevím. Mám na mysli prostředí, které by podporovalo jak vývoj agentů (mohly by zde být připravené moduly pro neuronové sítě, Kohonenovy mapy, q-učení, nebo pro integraci „behaviour oriented” jazyků), tak jejich statistické testování. IDE by mělo ideálně spolupracovat s více různými simulačními prostředími. Každé prostředí by bylo pro agenta definováno množinou poskytovaných senzorů a množinou v něm proveditelných akcí. Domnívám se, že podobné vývojové prostředí by mohlo urychlit prototypování agentů a zvýšit tak efektivitu práce vývojáře.

Literatura

[1] Berkowitz Leonard: Causes and Consequences of Feelings, Cambridge University Press, 2000

[2] Blumberg Bruce Mitchell: Old Tricks, New Dogs: Ethology and Interactive Creatures, PhD thesis, Massachusetts Institute of Technology, 1996

[3] Breve: <http://www.spiderland.org/breve/> [cit. 2006-05-22]

[4] Coutinho Eduardo, Miranda Eduardo R., Cangelosi Angelo: Towards a Model for Embodied Emotions. In: Proceedings of the Workshop on Affective Computing: Towards Affective Intelligent Systems, Covilhã, 2005

[5] Foltýn Lukáš: Realization of Intelligent Agents Architecture for Artificial Life Domain, Master thesis, Czech Technical University in Prague, 2005

[6] Framstick: <http://www.frams.alife.pl/> [cit. 2006-05-22]

[7] Gillerová Ilona: Slovník základních pojmů z psychologie, Fortuna, Praha, 2000: 16

[8] McCauley Lee, Franklin Stan: An Architecture for Emotion. In: Emotional and Intelligent: The Tangled Knot of Cognition, Menlo Park, CA: AAAI Press, 1998: 122 – 127

[9] Minsky Marvin: The Emotion Machine, draft, [online]. 2005 [cit. 2006-03-04], <http://web.media.mit.edu/~minsky/>

[10] Stuchlíková Iva: Základy psychologie emocí, Portál, Praha, 2002

[11] Sutton Richard: 1.6 History of Reinforcement Learning [online]. 1997 [cit. 2006-05-22]. <http://www.cs.ualberta.ca/~sutton/book/1/node7.html>.

[12] Sutton Richard: Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, 1038-1044, 1996

[13] Tyrrell Toby: Computational Mechanisms for Action Selection, PhD thesis, University of Edinburgh, 1993

[14] Voců Michal: Šachový šampión poražen počítačem IBM. Ikaros [online]. 1997 [cit. 2006-05-19], <http://www.ikaros.cz/node/29>

[15] Watkins Christopher: Learning from delayed rewards, PhD thesis, University of Cambridge, Psychology Department, 1989

[16] Wikipedia: Simulated annealing - Wikipedia, the free encyclopedia [online]. 2006 [cit. 2006-05-19]. <http://en.wikipedia.org/wiki/Simulated_annealing>.

[17] Wooldridge M., Jennings N. R.: Intelligent Agents – Theories, Architectures, and Languages. In: *Volume 890 of Lecture Notes in Artificial Intelligence*, Springer-Verlag, 1995

Seznam obrázků