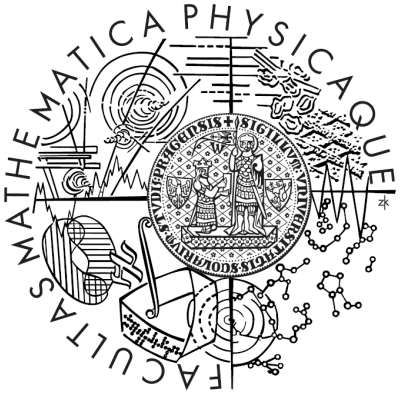
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Otakar Trunda

Zavedení kast do evoluce umělých bytostí

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: RNDr. Jan Hric
Studijní program: Informatika
Studijní obor: Obecná informatika

Praha 2011

Chtěl bych poděkovat vedoucímu své bakalářské práce RNDr. Janu Hricovi za příjemnou spolupráci a cenné rady a připomínky, které mi při psaní práce poskytl.

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

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

V ..... dne ..... Podpis autora

Název práce: Zavedení kast do evoluce umělých bytostí

Autor: Otakar Trunda

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: RNDr. Jan Hric

Abstrakt: V této práci se zabýváme problematikou evoluce umělých bytostí rozdělených do kast. Představujeme návrh prostředí umožňujícího uživateli definovat různé kasty včetně jejich vzájemných vztahů a simulovat život příslušníků těchto kast ve virtuálním světě. Popisujeme vlastnosti námi navrženého prostředí a jeho aplikovatelnost demonstrujeme na několika typických scénářích s využitím kast. V takto navrženém prostředí zkoumáme modelové situace, kde bytosti ve virtuálním světě zastávají různé role. Pozorujeme výsledky simulací a zkoumáme, jaký vliv na výsledek mají počáteční podmínky, hodnoty parametrů agentů (např. maximální věk) a hodnoty parametrů prostředí (např. pravděpodobnost mutace).

Klíčová slova: Umělý život, kasty, evoluce, agentové systémy

Title: Introduction of Castes in Evolution of Artificial Beings

Author: Otakar Trunda

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. Jan Hric

Abstract: In present work we study questions about introduction of castes in evolution of artificial beings. We present the concept that allows defining various castes and its mutual relations and simulating life of members of these castes in virtual environment. We describe features of our concept and illustrate its usability on several typical scenarios using castes. In this environment we study model situations where the beings in virtual world play different roles. We observe the simulations and its results trying to learn what impact the initial conditions, agents parameters (such as Life expectancy) and environment parameters (e.g. mutation probability) have on the outcome of the simulation.

Keywords: Artificial life, castes, evolution, agent based systems

Obsah

Kapitola 1

Úvod

1.1 Motivace

Při simulování života umělých bytostí sledujeme vždy nějaký cíl, fenomén motivovaný biologicky, sociologicky nebo jinak. V některých experimentech jsou všichni agenti účastnící se simulace stejného druhu (například simulace pohybu lidí v davu, mravenčí kolonie, šíření nákazy v populaci a podobně). Existují však scénáře, kde potřebujeme více druhů agentů. Může se jednat např. o modelování vztahu predátor - kořist, potravních řetězců nebo různého pohlaví agentů. Rozdělení agentů do kast nám umožní právě takové situace modelovat.

1.2 Cíle práce

Cílem této práce je prozkoumat možnosti jak navrhnout model, který by umožňoval uživateli provádět experimenty se simulováním umělého života, ve kterých se vyskytuje více druhů agentů. Důraz je zde kladen na to, aby systém byl flexibilní, tedy aby uživatel mohl sám popsat, jaké druhy agentů se budou v prostředí vyskytovat a jaké budou jejich vzájemné vztahy.

Systém bude simulovat přirozenou evoluci agentů pomocí genetického algoritmu a měl by umožňovat napodobit některé aspekty života skutečných organismů.

Našim dalším cílem je navržený systém implementovat a jeho použitelnost předvést na několika typických scénářích využívajících kasty. Ukážeme přitom jakým způsobem se dá s programem pracovat, jaké experimenty s ním lze provádět a jak tyto experimenty navrhovat.

Zhodnotíme také jaká jsou omezení našeho návrhu, které scénáře se v něm nedají dobře realizovat a proč tomu tak je.

1.3 Obsah práce

V první kapitole stručně popisujeme motivaci a stanovujeme cíle této práce. V kapitole Analýza představujeme různé přístupy k řešení a hodnotíme, jak jsou vhodné pro naše účely. Ve třetí kapitole popisujeme konkrétně návrh, pro který jsme se rozhodli a objasňujeme důvody, které vedly k volbě tohoto řešení.

Čtvrtá kapitola je věnovaná experimentům. Popisujeme zde sadu experimentů a hodnotíme jejich výsledky. Dále se zde věnujeme také otázkám navrhování experimentů a vlivu různých parametrů na výsledky simulací.

Na závěr shrnujeme přínos této práce a uvažujeme i o možných rozšířeních. Poté následuje seznam použité literatury a přílohy k práci.

Kapitola 2

Analýza

Shrnutí: V této kapitole budeme analyzovat otázky týkající se návrhu programu. Představíme různá řešení, o kterých jsme uvažovali a popíšeme jejich výhody a nevýhody. Konečný návrh představíme v následující kapitole.

2.1 Vymezení pojmů

V dalších částech textu budeme používat pojmy z oblasti agentových systémů. Některé z nich zde připomeneme:

Agent: entita, která dokáže přijímat podněty z okolí a svou činností okolí také ovlivňovat.

Prostředí: prvek, se kterým agenti komunikují. Každý agent je spjatý s prostředím, ve kterém pracuje. Bez prostředí nemá smysl o agentech hovořit.

Senzory: prostředky, pomocí kterých agent přijímá podněty z okolí. Souhrn všech informací, které senzory poskytují v jeden okamžik nazýváme vjem.

Akční členy: prostředky, pomocí kterých agent ovlivňuje své okolí. Atomické činnosti, které agent může vykonávat, nazýváme akce.

Agentova funkce: zobrazení, které posloupnosti vjemů přiřazuje akci. Další akce agenta je určená hodnotou agentovy funkce jejímž parametrem je

posloupnost všech vjemů, které agent zatím přijal. Agentova funkce jednoznačně popisuje veškeré agentovo chování. Jedná se o zobrazení, tedy abstraktní matematický popis.

Agentův program: konkrétní implementace agentovy funkce. Také bývá nazýván "rozhodovací mechanismus". Vstupem je současný vjem a stav agenta, výstupem pak agentova akce (případně posloupnost atomických akcí, pokud se rozhodování provádí na vyšší úrovni). Zde má smysl klást otázky typu: jak dlouho trvá než agent dospěje k rozhodnutí, kolik paměti potřebuje agentův program, na jakém principu se provádí rozhodování a podobně.

Interakce Jakákoliv výměna informací mezi agentem a prostředím nebo mezi dvěma (případně i více) agenty.

2.2 Prostředí a agenti

Prostředí i agenty chceme navrhnout tak, abychom mohli sledovat zajímavé modelové situace využívající kasty. Zároveň se snažíme, aby navržené řešení bylo co nejjednodušší. Návrh můžeme rozdělit na 3 základní části:

1. Interakce mezi agenty a prostředím a mezi agenty navzájem
2. Vlastnosti prostředí
3. Senzory a akční členy agentů

Toto rozdělení je pouze formální, protože jednotlivé části navzájem úzce souvisejí a není možné je striktně oddělit. Návrh interakcí mezi agenty ovlivňuje volbu akčních členů i další části návrhu, proto se této otázce budeme věnovat nejdříve.

2.2.1 Interakce mezi agenty

Jak jsme již naznačili v úvodu, zabýváme se umělým životem a budeme se snažit, aby náš program umožňoval do jisté míry napodobit život skutečných organismů. V tomto duchu tedy navrhneme interakce mezi agenty. K základním vlastnostem živých organismů patří metabolismus, neboli příjem a výdej látek a energie, a schopnost rozmnožování. To jsou také nejdůležitější interakce agentů s prostředím. Mezi další běžné interakce s prostředím patří pohyb; interakce mezi agenty pak mohou být například sežrání jednoho agenta druhým nebo komunikace.

Pokud mají mít interakce nějaký smysl, pak musejí ovlivňovat agentovo okolí tedy prostředí a ostatní agenty. Je proto třeba, aby si agenti udržovali nějaký vnitřní stav, který se bude při interakcích měnit. Přirozenou volbou jak takový vnitřní stav zavést, je přiřadit každému agentovi číslo vyjadřující zásobu jeho energie. Pomocí zásoby energie je pak možné simulovat metabolismus agenta tak, že některé interakce budou agentovu zásobu energie zvyšovat a jiné snižovat.

Tento přístup je možné rozšířit i na více parametrů. Agentův stav může být tvořen několika různými hodnotami vyjadřujícími například agentovu potřebu jíst, pít, spát, míru agentovy spokojenosti a podobně. Interakce s okolím by pak příslušným způsobem ovlivňovaly hodnoty těchto parametrů. V dalších částech textu budeme pro zjednodušení popisu uvažovat pouze jeden parametr.

Návrh interakcí mezi agenty inspirovaný skutečným životem by vypadal takto:

- Interakce mezi agentem a prostředím
  - Získávání informací o okolí
  - Pohyb
- Interakce mezi agenty navzájem
  - Sežrání jednoho agenta druhým
  - Rozmnožování

Tento způsob interakce mezi agenty však není jediný možný. Zmíníme se nyní ještě o jiném, obecnějším přístupu: na interakci mezi dvěma agenty se můžeme dívat jako na hraní hry. Agenti, kteří se potkají hrají proti sobě hru, jejíž pravidla jsou definovaná v rámci prostředí. Výsledek hry pak může ovlivnit vnitřní stav obou agentů.

V prvním případě inspirovaném skutečným životem jsou typy interakcí poměrně omezené: požírání agentů způsobí smrt jednoho agenta a přenese jeho energii k predátorovi. Při návrhu založeném na hraní hry jsou možnosti širší: lze definovat pravidla tak, že výsledkem hry bude například smrt obou agentů, ztráta energie, nebo naopak zisk energie obou hráčů.

Pomocí hraní hry můžeme simulovat některé scénáře, na které by zjednodušený model nestačil. Jeden z nich zde popíšeme: (Tento modelový příklad je převzatý z [1].) V prostředí se nachází potrava, o kterou agenti soupeří. Agenti jsou rozděleni do dvou druhů podle strategie, kterou používají.

Strategie Jestřáb: bojuje, dokud nezvítězí, nebo není vážně zraněn;

Strategie Hrdlička: pouze rituální souboje, při napadení prchá.

Pravidla hry jsou definovaná takto:

- vítězství (zisk potravy): +50
- prohra: 0
- zranění: -100
- ztráta času: -10

Výsledek hry se určí podle této tabulky: (první číslo určuje zisk vítěze, druhé zisk poraženého)

- Jestřáb x Jestřáb: 50/-100
- Jestřáb x Hrdlička: 50/0
- Hrdlička x Hrdlička: 40/-10

Jestřáb vždy poráží Hrdličku, při souboji Jestřáb x Jestřáb nebo Hrdlička x Hrdlička je vítěz určen náhodně.

V takto definovaném prostředí můžeme zkoumat například jak se bude vyvíjet poměr počtu Jestřábů a Hrdliček a další otázky týkající se tzv. evolučně stabilních strategií. Více informací o tomto tématu lze nalézt v [1].

Rozdíl mezi těmito dvěma přístupy je zřejmý: v prvním případě inspirovaném skutečným životem je možné získat energii jen sežráním jiného agenta, což způsobí smrt kořisti. Pokud budou interakce definované jako hraní hry, pak toto není nutné a navíc lze pravidla nastavit i tak, že souboj skončí ziskem obou hráčů. (Typickým příkladem může být tzv. vězňovo dilema).

Otázkou však zůstává, jak do takového modelu zavést evoluci agentů. V těchto scénářích se často omezujeme pouze na sledování vývoje počtu agentů jednotlivých kast bez jakýchkoli změn jejich parametrů. Dále je třeba rozlišit, zda výsledek hry bude záviset pouze na kastách účastníků (tak je tomu v předchozím příkladě), nebo zda agenti budou hru skutečně hrát a každý jednotlivý agent může hrát jiným způsobem. Ve druhém případě už by použití evoluce bylo možné. Zmíněný příklad můžeme do této podoby upravit následovně: každý agent bude mít parametr určující pravděpodobnost, že bude hrát jako Jestřáb nebo Hrdlička, v souboji si podle této pravděpodobnosti vždy zvolí svou strategii. Tento parametr může podléhat evoluci (při rozmnožování přechází z rodičů na potomky a prochází mutací).

V této práci se zaměříme na přístup inspirovaný skutečným životem, bez hraní hry. Modely, kde jsou interakce zavedené jako hra, lze zkoumat například pomocí programu NetLogo.

2.2.2 Vlastnosti prostředí

V této části rozebereme, jaké vlastnosti bude mít prostředí, ve kterém se agenti pohybují. V praxi nastává spíše situace, kdy vlastnosti prostředí jsou dané (například kvalitou senzorů a akčních členů) a je možné je pouze studovat a přizpůsobit se jim. V našem případě však máme volné ruce a můžeme prostředí navrhnout tak, jak se nám hodí. Na prostředí se můžeme dívat z několika různých hledisek.

1. Z hlediska pozorovatelnosti
    - Plně pozorovatelné prostředí - agenti jsou schopni získat informace o úplném stavu celého prostředí
    - Částečně pozorovatelné prostředí - některé informace o stavu prostředí jsou pro agenty skryté
2. Podle rozdělení prostoru
    - Diskrétní - prostředí se skládá políček, agenti se pohybují mezi sousedními políčky
    - Spojité - prostředí není rozdělené na části, agenti se mohou pohybovat volně
3. Podle rozdělení času
    - Diskrétní v čase - děje v prostředí probíhají po krocích

- Spojité v čase - stav prostředí se mění plynule

4. Podle původu změn prostředí
   - Statické - stav prostředí se mění pouze činností agentů, prostředí samo o sobě je neměnné
   - Dynamické - prostředí může samovolně měnit svůj stav

Nyní popíšeme jednotlivé vlastnosti prostředí podrobněji včetně jejich důsledků a zhodnotíme, jak jsou vhodné pro naše potřeby.

Úplná x částečná pozorovatelnost Plně pozorovatelné prostředí je z biologického hlediska nerealistické. Navíc předpokládáme, že prostředí bude rozsáhlé, kdežto interakce budou pouze lokální, takže nebude třeba, aby agenti pozorovali odlehlé části prostředí. Ani u blízkých agentů nebude třeba, aby o nich agent získal úplnou informaci, například jejich vnitřní stav by měl být skrytý. Plně pozorovatelné prostředí by komplikovalo návrh rozhodování agentů, které chceme mít co nejjednodušší. Z těchto důvodů se částečná pozorovatelnost jeví jako lepší volba.

Diskrétní x spojité prostředí Diskrétní prostředí má výhodu v tom, že vjemy z agentových senzorů je možné jednoduše definovat jako uspořádané n-tice, kde n představuje počet políček, která agenti vidí. Každý prvek této n-tice by pak reprezentoval obsah příslušného políčka. Spojité prostředí naproti tomu žádné zásadní výhody nepřináší a takový návrh by byl obtížnější na implementaci.

Statické x dynamické prostředí V reálném prostředí nepochybně platí, že každá změna má svou příčinu. Rozdíl mezi statickým a dynamickým prostředím spočívá v tom, zda tuto příčinu považujeme za agenta, nebo za vlastnost prostředí. Ve statickém prostředí jsou veškeré změny způsobené agenty. To nám však nemusí vždy vyhovovat. Často má simulace jeden hlavní cíl, problém, na který má pomoci odpovědět, a na základě toho jsou navrženi agenti. Ty změny prostředí, jejichž příčina není důležitá (nesouvisí se sledovaným jevem) je pak vhodné definovat jako vlastnost prostředí.

Příkladem může být simulace pohybu lidí ve městě. Pohyb lidí bude nepochybně záviset na počasí (například zda prší, nebo neprší), ale příčina změn počasí (proč prší) pro nás není důležitá. Proto je vhodné pracovat s počasím jako by to byla vlastnost prostředí.

Další výhodou dynamického prostředí je, že jevy definované jako vlastnosti prostředí, můžeme přímo ovlivňovat. Ve výše zmíněném příkladě můžeme nastavit třeba: od 15:00 do 17:00 bude pršet. Naproti tomu jevy vzniklé činností agentů můžeme ovlivňovat pouze nepřímo (motivovat agenty k tomu, aby se chovali určitým způsobem). Této možnosti dobře využili autoři v práci [3], kterou zde stručně popíšeme.

Jedná se o systém, ve kterém agenti mimo jiné hledají potravu, potrava vyrůstá v prostředí podle tzv. míry úrodnosti (vyšší míra - více potravy a naopak). Agenti se v průběhu simulace učí, později dokáží potravu hledat lépe. Na začátku jsou však nastaveni náhodně a jejich schopnosti hledat potravu jsou velmi malé. Tento problém autoři vyřešili tak, že zvolili míru úrodnosti jako vlastnost prostředí a to jim umožnilo tento parametr v průběhu simulace regulovat. Na začátku používali vysoké hodnoty, aby agenti byli schopní přežít, později, když byli agenti lépe přizpůsobení, úrodnost postupně snižovali.

2.2.3 Senzory a akční členy

Senzory hrají zásadní roli při agentově rozhodování. Musejí podávat dostatečně kvalitní informace o agentově okolí ale zároveň bychom nechtěli, aby agentovy vjemy obsahovaly nepotřebné informace. Přesněji řečeno každá dodatečná informace může být svým způsobem užitečná, ale na druhou stranu čím složitější budou vjemy, tím složitější bude i rozhodovací mechanismus agentů a ten chceme udržovat jednoduchý. Je proto potřeba najít rozumný kompromis.

Další otázkou je, zda mají být senzory a akční členy stejné pro všechny agenty, nebo se některé skupiny agentů budou v tomto ohledu lišit. K tomuto problému se vrátíme v následující sekci, která je věnovaná zavedení kast.

Senzory agentů Při návrhu senzorů je třeba si položit otázku, které informace jsou pro agenta důležité. Pokud vyjdeme opět z biologické předlohy, pak mezi nejdůležitější informace patří:

- zda je prostor před agentem volný
- zda je v okolí potrava
- zda je v okolí predátor nebo jiný zdroj nebezpečí
- zda je v okolí vhodný partner k rozmnožování

Alespoň tyto informace by tedy senzory měly v nějaké formě poskytovat (přímo či nepřímo).

Návrh senzorů, který by odpovídal požadavkům, může vypadat takto: agent dostává informace o obsahu okolních políček, ale pouze do té míry, že vidí jaká je kasta agenta, nacházejícího se na daném políčku. Nemá informaci o jeho vnitřním stavu, ani nedokáže rozlišit dva různé příslušníky stejné kasty. Takové zjednodušení je inspirované faktem, že informace o vnitřním stavu pozorovaného agenta, ani schopnost rozpoznávat různé členy stejné kasty nejsou pro agenta v našem modelu důležité.

Zjednodušení vjemů je pro nás samozřejmě vhodné, na druhou stranu při tomto návrhu agent nebude schopen odlišit například své příbuzné od ostatních příslušníků stejné kasty, což nám znemožní simulovat některé složitější chování agentů, jako je péče o potomky. Není však našim cílem takto složitého chování dosáhnout, a proto bude výše popsaný model pro naše účely dostačující.

Jiný případ, kdy je rozlišování individualit důležité, nastává při modelování chování označovaného jako "Tit for tat" [2, 1], což můžeme přeložit jako "půjčka za oplátku" nebo "oko za oko, zub za zub". Pro toto chování je typické, že agent oplácí, čili chová se k ostatním agentům stejně jako se oni zachovali k němu. Je zřejmé, že k tomu potřebuje jednotlivé agenty rozlišovat.

Tento přístup se používá i v pozměněné podobě, kdy agent oplácí nejen těm, kteří se zachovali špatně k němu, ale všem, kteří se zachovali špatně ke komukoli. A obdobně odměňuje ty, kteří se zachovali dobře (ne nutně k němu, ale ke komukoli). Pak hovoříme o tzv. altruistickém chování [4]. Zde je nutné aby si agent pamatoval nejen své interakce s ostatními, ale aby mohl pozorovat všechny interakce mezi všemi agenty. Zde se těmito přístupy dále zabývat nebudeme.

Pokud bychom se rozhodli pro model, který by umožňoval rozlišovat individuální agenty, vedlo by to k mnoha dalším změnám a celý systém by se značně zkomplikoval. Agent by musel informaci o své individualitě poskytovat, ostatní agenti by ji museli detekovat (složitější senzory, mnohonásobně větší prostor vjemů) a poté tuto informaci využít při svém rozhodování (složitější rozhodovací mechanismus, bylo by nutné, aby agenti měli paměť).

Otázky v této části rozebíráme z pohledu agentových senzorů. Je však možné se na ně dívat i jako na vlastnosti prostředí a to tak, že prostředí udává jistá pravidla, která agenti musejí dodržovat. Tato pravidla pak popisují, které informace jsou pro agenty přístupné, a které nikoliv. Jak jsme zmínili výše, návrh senzorů a prostředí není možné striktně oddělovat.

Akční členy agentů Akce, které agenti mohou vykonávat, budou odpovídat interakcím popsaným výše. Zejména je třeba zmínit akci pro rozmnožování, která je důležitá kvůli použití genetického algoritmu. Vykonávání akcí by mělo být spojeno se ztrátami energie (každá akce by měla mít určenou cenu, kterou agent zaplatí za použití této akce), což odpovídá biologické předloze. Konkrétní návrh akcí agentů je popsaný v následující kapitole.

2.3 Rozhodování agentů

Zde popíšeme několik běžně používaných způsobů implementace rozhodovacího mechanismu agentů. U každého zhodnotíme jejich výhody a nevýhody a posoudíme, jak jsou vhodné pro naše potřeby. Na rozhodovací mechanismus klademe tyto požadavky:

1. Autonomie agentů. Agenti musejí být autonomní, budou se rozhodovat pouze na základě svých vjemů a svého stavu.

2. Adaptivita agentů. Chceme, aby agenti byli schopní přizpůsobovat své chování okolnímu prostředí a reagovat i na případné změny prostředí. (Adaptivita by se měla objevit v průběhu evoluce, tedy další generace by měly být lépe přizpůsobené než předchozí. Nepožadujeme, aby se jednotliví agenti přizpůsobovali v průběhu svého života.) Tento požadavek souvisí s následujícím bodem.

3. Dostatečná vyjadřovací síla. Rozhodovací systém by měl umožňovat, aby se u agentů vyvinulo rozumné chování.

4. Podpora pro použití genetického algoritmu. Našim cílem je rozhodování agentů vylepšovat genetickým algoritmem. Z toho důvodu je třeba, abychom rozhodovací mechanismy agentů mohli mutovat a navzájem křížit.

5. Malé paměťové nároky. Předpokládáme, že ve virtuálním ekosystému se budou nacházet tisíce nebo desítky tisíc agentů. Každý z nich bude mít svůj rozhodovací program, tento program by proto neměl být příliš náročný na paměť.

6. Malé časové nároky. Ze stejného důvodu jako v předchozím bodě požadujeme, aby rozhodování agentů netrvalo příliš dlouho.

Je zřejmé, že některé požadavky jsou částečně protichůdné: čím složitějšího chování budeme chtít docílit, tím větší budou časové i paměťové nároky programů. Je třeba najít rozumný kompromis.

Z popsaných požadavků také plyne, že některé modely jsou pro nás nevhodné. Existují rozhodovací mechanismy navržené pro řízení jediného agenta, který má plnit složité úkoly. Tyto přístupy nebudou příliš úspěšné při použití v prostředí obsahujícím mnoho jednoduchých agentů, od kterých neočekáváme vysoce sofistikované chování. Takové způsoby řízení jsou totiž časově a paměťově náročné (i pokud se jedná o jediného agenta) a nevznikají obvykle pomocí genetického algoritmu.

Jako vhodnější se jeví jednodušší způsoby rozhodování, například tzv. reaktivní plánování. Při tomto způsobu plánování agent neudržuje žádnou vnitřní reprezentaci znalostí a pouze převádí aktuální vjem na akci (případně může udržovat vnitřní stav). Díky tomu, že není třeba pracovat se znalostmi, má systém menší paměťové nároky a genetické operátory mohou být jednodušší.

2.3.1 If-then pravidla

Ve své základní podobě spadá tento způsob do kategorie reaktivního plánování. Agentův program se skládá z množiny pravidel, na jejichž levé straně je podmínka (co musí platit, aby bylo možné pravidlo použít) a na pravé straně akce agenta. Tento nejjednodušší přístup lze dále rozšířit například na tzv. hierarchický model, kdy na pravé straně některých pravidel není přímo akce, ale odkaz na další, podřízenou sadu pravidel. Pravidla pak tvoří stromovou strukturu. Pro použití tohoto modelu v našich podmínkách by bylo potřeba vyřešit několik dalších otázek:

- jak zajistit, aby celý prostor agentových vjemů byl pokrytý pravidly - tedy aby agent vždy našel použitelné pravidlo.
- jak zajistit, aby akce byla v každém kroku určena jednoznačně - jak řešit případy, kdy lze použít více pravidel.
- jak navrhnout křížení a mutaci agentova programu.

Jedním z možných řešení, je navrhnout pravidla tak, aby jejich levé strany disjunktně pokrývaly celý prostor agentových vjemů. V tomto případě by tedy nikdy nenastávaly konflikty (situace, kdy lze použít více různých pravidel), protože pro každý vjem by existovalo právě jedno použitelné pravidlo. Není však jednoduché navrhnout pro tuto reprezentaci genetické operátory, které by zachovávaly požadovaný invariant (disjunktní pokrytí celého prostoru agentových vjemů) a zároveň udržovaly sadu pravidel agenta rozumně malou.

Jinou možností je použít priority u jednotlivých pravidel. Pokud by šlo v dané situaci použít více pravidel, pak se vybere to s nejvyšší prioritou. Tento přístup včetně návrhu genetických operátorů je podrobně popsán v následující kapitole.

2.3.2 Konečný automat

Agentova funkce je v tomto případě reprezentovaná pomocí konečného automatu. Automat obsahuje množinu stavů, z nichž vždy právě jeden je aktivní, a přechodů mezi stavy. Přechody mají tvar if-then pravidla: na levé straně je podmínka, na pravé straně odkaz na nový stav. Podmínky nejsou v tomto případě totalistické (nepokrývají všechny agentovy vjemy), pokud žádná z podmínek pro přechod mezi stavy neplatí, pak agent zůstává ve stejném stavu. S každým stavem je navíc asociován mechanismus pro určení další akce.

Zde existuje více možností: od jednoduchých, kdy každý stav obsahuje právě jednu akci, až po složitější modely, kdy ve stavu je posloupnost akcí nebo skript, který určuje další rozhodování. Provádění této posloupnosti nebo skriptu je přerušeno jakmile se změní stav automatu (uspěje některé pravidlo pro přechod mezi stavy). Používá se také model zvaný hierarchický konečný automat, kde každý stav může mít ještě několik podstavů a teprve v nich je rozhodovací mechanismus.

Pro konečné automaty je obtížné navrhnout rozumně operátor křížení. V odvětví zvaném evoluční programování, které se mimo jiné zabývá i evolucí konečných automatů, převažují postupy, které křížení vůbec nepoužívají a vystačí si pouze s mutacemi. Pokud bychom tento postup použili v našem modelu, znamenalo by to povolit pouze nepohlavní rozmnožování agentů (tedy agent vytvoří svůj klon, který následně projde mutací). Pro pohlavní rozmnožování bychom křížení potřebovali.

2.3.3 Stochastické modely

Oba předchozí modely jsou deterministické - další akce agenta je jednoznačně určena jeho stavem a současným vjemem. To má za následek, že v konkrétní situaci se agent zachová vždy stejně. Toto nám v některých případech nemusí vyhovovat, proto byly navrženy pravděpodobnostní modely, ve kterých se při výběru akce uplatňuje náhodný faktor.

If-then pravidla lze jednoduše rozšířit na stochastickou verzi například tak, že na pravé straně pravidel nebude pouze jediná akce, ale několik různých akcí a jejich pravděpodobnosti. Pokud toto pravidlo uspěje, pak agent vybere jednu z odpovídajících akcí podle příslušného pravděpodobnostního rozdělení. Náhodný faktor lze využít také při řešení konfliktů (když lze v konkrétní situaci použít více pravidel).

Model využívající konečný automat lze randomizovat přidáním pravděpodobnosti při přechodech mezi jednotlivými stavy. Pravidlo pro přechod mezi stavy neurčuje cílový stav jednznačně, ale obsahuje několik možných cílových stavů a pravděpodobnosti jejich vybrání.

V našem případě by tento postup mohl přinést jisté výhody. Pokud bychom se rozhodli pro použití reaktivního plánování, pak agentova akce bude záviset pouze na aktuálním vjemu. Agent se ve stejné situaci rozhodne vždy stejně. Toto příliš neodpovídá skutečným organismům, které mají paměť a pokud se dostanou do stejné situace znovu jejich vnitřní stav (obsah paměti) je jiný a mohou se rozhodnout jinak. Přidání náhodného faktoru by tento problém mohlo částečně odbourat.

S tím souvisí i nebezpečí zacyklení. Reaktivní agenti v částečně pozorovatelném prostředí mohou cyklit, čemuž bychom se chtěli vyhnout. (I když zde to nepředstavuje žádný zásadní problém, předpokládáme totiž, že život každého agenta bude omezený maximálním věkem, kterého se může dožít. Agenti, kteří se v průběhu života zacyklí, se nerozmnoží a zahynou.)

Je třeba zmínit i nevýhody. Jsou to zejména: delší čas potřebný pro generování náhodných čísel, složitější řídící program (což v našem případě znamená i složitější genetické operátory) a také větší náročnost na paměť (místo jedné volby je třeba uchovávat více možností a jejich pravděpodobnosti).

2.3.4 Konekcionistické modely

Pro rozhodování agentů lze využít i umělou neuronovou síť, na jejíž vstupy přivedeme agentovy vjemy (nebo pouze současný vjem pokud používáme reaktivní plánování) a stav, a jejíž výstup bude určovat agentovu akci. V našem modelu však požadujeme, aby se rozhodování agentů vylepšovalo pomocí genetického algoritmu, což by v tomto případě mohl být problém.

Pro učení neuronových sítí převládají spíše přístupy tzv. učení s učitelem, kde se síť učí na množině trénovacích dat. Potřebovali bychom zde tedy znát, které výstupy jsou žádoucí a které ne. V našem modelu však požadujeme, aby se agenti samostatně naučili v prostředí přežívat - předpokládáme tedy učení bez učitele.

Bylo by sice možné použít podobný postup jako v NEAT [5] (bude popsáno dále) s tím, že fitness nebudeme určovat na základě trénovací množiny, ale necháme ji implicitní (bude ji určovat to, zda agent dokáže v prostředí přežít a rozmnožit se, nebo ne). Některé aspekty přístupu používaného v NEAT však v našich podmínkách nejsme schopni napodobit, a proto by tento pokus pravděpodobně nepřinesl očekávané výsledky.

Jinou možností jak učit neuronovou síť je pomocí tzv. posilovaného učení, které by v našich podmínkách bylo možné realizovat. Prostředí by dávalo agentům zpětnou vazbu určující, zda bylo jejich rozhodnutí správné, nebo ne (systém odměn a trestů). Odměna by mohla následovat například za zisk energie, rozmnožení se nebo dosažení vysokého věku. Tento přístup však nepoužívá evoluční algoritmus a proto nevyhovuje našim požadavkům.

Existují i jiné, komplikovanější konekcionistické modely než klasická neuronová síť. Můžeme sem zařadit např. Tyrrellovu free-flow hierarchii [6]. Tyto složitější přístupy však slouží k simulování komplexního chování a pro naše účely nejsou vhodné z důvodů popsaných výše.

2.3.5 Plánování s cílem

Jedná se o proaktivní přístup - agent plánuje delší posloupnosti akcí, ne jen jeden krok dopředu. Umožňuje modelovat sofistikovanější chování agentů, ale za cenu vyšších časových nároků (používá prohledávání stavového prostoru).

Rozhodovací systém je v tomto případě mnohem složitější. Obsahuje komponentu pro plánování, pro generování cílů (některé cíle mohou být protichůdné - komponenta pro řešení konfliktů). Bylo by velmi obtížné navrhnout pro takový systém genetické operátory.

Dále je třeba udržovat vnitřní stav (například v jaké fázi provádění plánu se agent právě nachází) a mapu světa (případně i jiné znalosti), což má velké paměťové nároky. Tento způsob může být vhodný pro kvalitní řízení jednoho agenta, ale ne pro naše účely.

2.4 Zavedení kast

Stejně jako u rozhodování agentů se i zde nabízí několik alternativ, jak do našeho modelu zavést podporu pro rozdělení agentů do kast. Nejprve shrňme, jakých cílů chceme dosáhnout:

- Kasty budou definované uživatelem. Uživatel sám navrhne kasty a specifikuje jejich vzájemné interakce.
- Příslušnost k určité kastě bude rozhodujícím způsobem ovlivňovat agentovo postavení ve virtuálním světě.
- Systém kast bude dostatečně silný a flexibilní, aby uživateli umožňoval snadno navrhnout vlastní experimenty, ve kterých agenti zastávají různé role.

2.4.1 Zavedení kast přímým způsobem

Kasty můžeme zavést přirozeným způsobem tak, že určíme vlastnosti, kterými se jednotlivé kasty mohou navzájem odlišovat, a tyto vlastnosti nastaví uživatel. Odlišnosti mezi jednotlivými kastami můžeme rozdělit na 3 kategorie: (vycházíme opět z biologické předlohy)

1. rozdíly v agentově vnitřním fungování
    - délka života
    - množství energie, které agent může uchovávat
    - způsob rozmnožování (pohlavní x nepohlavní)
    - kolik své energie předává potomkům
    - další parametry například délka spánku, délka těhotenství, věk pohlavní dospělosti

2. rozdíly v interakcích s prostředím
    - akce, které mohou agenti vykonávat
    - množství energie, které agent platí za vykonávání akcí
    - jakou část prostředí může pozorovat
3. rozdíly v interakcích s ostatními agenty
    - které agenty může sežrat
    - kolik energie získá sežráním kořisti
    - se kterými agenty se může rozmnožovat

Tento seznam rozhodně není vyčerpávající, lze si představit mnoho parametrů, ve kterých se jednotlivé kasty mohou lišit.

Pro naše účely potřebujeme navrhnout pevně danou sadu parametrů, jejichž hodnoty budou definovat kastu. Konkrétní hodnoty pak nastaví uživatel. Zde nastávají problémy pokud povolíme, aby různé kasty mohly vykonávat různé akce nebo získávaly různé vjemy ze senzorů: Není možné dát uživateli obecný nástroj, pomocí kterého by sám navrhoval senzory a akční členy agentů. Je však možné pro tyto parametry (parametry určující agentovy senzory a akční členy) definovat několik přípustných hodnot a dát uživateli možnost volby, kterou z těchto hodnot vybere.

2.4.2 Zavedení kast pomocí evolučního procesu

Jiný způsob jak zavést kasty do našeho modelu je přímo pomocí evolučního procesu. Pokud povolíme, aby se v průběhu evoluce měnilo nejen rozhodování agentů, ale i parametry ovlivňující agentovy interakce s okolím (např. které agenty může sežrat a kolik energie tím získá), pak se v průběhu času budou agenti samovolně rozčleňovat do kast v závislosti na kombinaci parametrů, která se u nich vyvine. Tento způsob také více odpovídá evoluci skutečných organismů. Nastává zde však několik problémů:

Kombinace parametrů by byla u každého jednotlivého agenta jiná a tedy, přísně vzato, by každý agent tvořil samostatnou kastu. To ovšem nevyhovuje našim představám (agenti by se nemohli ani rozmnožovat, pokud povolíme rozmnožování jen v rámci stejné kasty), a proto bychom museli kasty definovat jiným způsobem - například zavést relaci ekvivalence na genomu tak, že organismy, jejichž genomy jsou si v jistém smyslu podobné, budou považovány za ekvivalentní. Kasty pak budou tvořeny třídami ekvivalence. Toto

opět částečně odpovídá biologické taxonomii - žádné dva organismy nemají přesně stejnou DNA, přesto je můžeme na základě podobnosti a vývojové příbuznosti rozdělit do druhů.

Druhým, závažnějším problémem je fakt, že v našem simulovaném světě neexistují fyzikální zákony, tedy mechanismus, který by zabraňoval, aby se u agentů vyvinuly nereálné vlastnosti. Například velkou výhodu by jistě získali agenti, u kterých by ceny za akce byly záporné - agenti by vykonáváním akcí získávali energii, nebo agenti, kteří by byli schopní sežrat jakýkoliv organismus a získat tím velké množství energie. Stejně tak maximální věk agentů a maximální zásoba energie jsou parametry, jejichž neomezené zvyšování poskytuje výhodu.

Tomuto problému můžeme čelit nastavením limitů - minimální ceny pro každou akci (žádná nižší by se nemohla vyskytovat) a hranice pro věk a zásobu energie, což ovšem povede pouze k tomu, že u agentů se postupem času vyvinou právě tyto nejvýhodnější limitní hodnoty. Jinou možností je přidání různých vztahů mezi parametry, například aby součet cen všech akcí byl větší než nějaká dolní mez nebo aby součet cen akcí byl přímo úměrný maximálnímu věku či maximální zásobě energie. Bohužel i tyto pokusy mají své problémy a domníváme se, že obdobu fyzikálních zákonů není možné jednoduše zavést.

Samovolný vznik kast pomocí evolučního procesu však přesto může mít své opodstatnění. Uvedeme zde příklad, ve kterém podobný princip funguje velmi úspěšně. Jedná se o již zmiňovaný systém NEAT [5] určený pro evoluci neuronových sítí. Vstupem programu je množina trénovacích dat a program se poté snaží najít neuronovou síť, která by trénovací množině nejlépe odpovídala. K tomu používá genetický algoritmus, vyvíjí se současně struktura sítí (počet neuronů a jejich zapojení) i váhy.

Problémem této techniky je právě současný vývoj struktury i vah sítě. Pokud v průběhu evoluce vznikne síť s jinou strukturou (např. pomocí mutace), pak tento nový jedinec má typicky mnohem horší fitness funkci než zbytek populace. Je to způsobeno tím, že nová struktura potřebuje jiné nastavení vah, aby byla úspěšná. Nově vzniklé struktury tak často nepřežijí v konkurenci ostatních již přizpůsobených sítí, přestože při správném nastavení vah by mohly být úspěšnější.

Tento problém je v NEAT vyřešen tak, že populace je při ohodnocování rozdělena na skupiny (kasty) podle struktury sítě. Při selekci se pak vybírá z každé skupiny zvlášť, tedy proti sobě soupeří pouze sítě se stejnou topologií. Takový přístup odpovídá tomu, jako by síť s jinou strukturou byla považována za jiný živočišný druh, který má v ekosystému jiné postavení (zaujímá jinou niku) a konkurencí jsou pro něho pouze ostatní obyvatelé stejné niky, tedy sítě se stejnou topologií.

Takový způsob zavedení kast však nevyhovuje dostatečně požadavku, aby kasty byly definované uživatelem, proto se jím nebudeme dále zabývat.

2.4.3 Související otázky

Vraťme se ještě k problému s potravou popsanému v sekci o dynamickém prostředí. Jedná se o to, zda potravu zařadíme mezi agenty, nebo mezi vlastnosti prostředí. Výhody přístupu, kdy vyrůstání potravy je bráno jako vlastnost prostředí, jsme již popsali. Tento způsob má však i jisté nevýhody, a to zejména malou flexibilitu.

Hlavní rozdíl mezi prací [3] a touto spočívá v tom, že pro nás je potrava ve středu zájmu. V našem programu se zaměřujeme na simulace virtuálních ekosystémů, které obsahují více druhů agentů. Chtěli bychom umožnit, aby různé druhy agentů spotřebovávaly různé druhy potravy, přičemž druhy potravy by navrhoval uživatel (stejně jako druhy - kasty - agentů).

Program by měl navíc umožňovat i vytváření jednoduchých potravních řetězců, což by znamenalo, že agenti mohou být potravou pro jiné agenty. Z tohoto pohledu nemá smysl pojmy "agent" a "potrava" oddělovat. Na "potravu" tvořící základ potravního řetězce bychom se mohli dívat jako na další kastu agentů.

Je zde však několik věci, které takový přístup znesnadňují:

- Agenti mohou vykonávat akce a samostatně se rozhodovat, potrava tyto vlastnosti nemá. (Respektive má je, ale pouze ve velmi omezené míře - může se rozmnožovat.)
- Pokud budeme (prvotní) potravu považovat za agenta, odkud bude brát energii?

- Pokud povolíme, aby se na jednom políčku virtuálního světa nacházel jen jeden agent, znamenalo by to, že na políčko, kde vyrůstá potrava, nemůže žádný (jiný) agent vstoupit, ani přes ně přejít.

Konečný návrh modelu potravy, včetně řešení těchto otázek je popsán v následující kapitole.

Kapitola 3

Návrh programu

Shrnutí: V této kapitole představíme návrh programu, jeho vlastnosti a odůvodníme zvolená řešení v případech, kde se nabízelo více alternativ. Zde se soustředíme hlavně na popsání myšlenek a principů, podrobné technické detaily lze nalézt v dokumentaci.

3.1 Umělý svět

Svět, ve kterém se agenti pohybují, je diskrétní - skládá se ze čtvercových políček. Každé políčko může být obsazené nejvýše jedním aktivním a nejvýše jedním pasivním agentem (viz dále). Prostředí je tzv. statické, což znamená, že stav světa je ovlivňován pouze činností agentů (včetně pasivních), nejsou zde žádné další vnější vlivy.

Pro agenty bude svět pouze částečně pozorovatelný. Tedy agenti nebudou mít informaci o stavu celého světa, ale pouze o svém bezprostředním okolí. Simulace probíhá po krocích, v každém kroku může agent provést jednu akci.

3.2 Agenti

Virtuální svět slouží jako prostředí pro život agentů. Agenti se v prostředí pohybují, rozmnožují a navzájem požírají, čímž ovlivňují svůj vnitřní stav (zásoba energie) i své okolí. Agenti nemají žádný předem daný úkol nebo cíl, pouze vykonávají akce podle své volby a buď jsou schopní v prostředí přežít a rozmnožit se, nebo hynou. Jedná se tedy o tzv. implicitní fitness funkci, jako selekce slouží přirozený výběr prostředí.

Nejdůležitějším parametrem agenta je jeho zásoba energie. Energie tvoří agentovu životní sílu - spotřebovává se při vykonávání akcí, získává se z potravy, pokud klesne na nulu, agent zemře. V průběhu svého života agent vykonává akce na základě své sady pravidel. Tato pravidla se v průběhu života agenta nemění. Život agenta končí, pokud jeho energie klesne na nulu, věk dosáhne maximální hodnoty nebo je sežrán jiným agentem.

Problém s potravou popsaný v předchozí části, jsme vyřešili následovně: Agenti jsou rozděleni na aktivní a pasivní. Pouze aktivní agenti mohou samostatně vykonávat akce. Pasivní agenti jsou nepohybliví a slouží jako potrava aktivním agentům - tedy jako základ potravních řetězců. Při navrhování experimentu a definování kast tak může uživatel pracovat s potravou jako s kastou agentů, což mu dává široké možnosti jak nastavit například rychlost růstu potravy a její výživnost pro další kasty. Naproti tomu v průběhu simulace se pasivní agenti nechovají jako agenti v pravém slova smyslu, nemají schopnost samostatného rozhodování. Mohou se pouze rozmnožovat a to tak, že vytvoří na vedlejším políčku svoji kopii.
Dále v textu se slovem agent myslí vždy aktivní agent.

Z biologického hlediska můžeme organismy rozdělit podle způsobu získávání energie na autotrofní - sami vytvářejí organické sloučeniny z anorganických, a heterotrofní - přijímají organické sloučeniny z potravy. V našem umělém světě můžeme tyto organismy modelovat pomocí parametru EnergyLostPerTick (množství energie, které agent ztratí za jeden krok simulace bez ohledu na to, jakou akci provede). Pokud bude hodnota parametru kladná, agent bude představovat heterotrofní organismus (stále ztrácí energii, musí ji doplňovat z potravy), záporná hodnota způsobí, že agent bude postupem času samovolně získávat energii a bude představovat autotrofní organismus. Pasivní agenti by měli být vždy autotrofní, protože nemohou vykonávat akci Eat.

Jelikož agenti v průběhu života spotřebovávají energii, musejí ji doplňovat z potravy. Při sežrání jiného agenta však agent získá nejvýše tolik energie, kolik měla jeho kořist, takže tímto způsobem by celkové množství energie v systému stále klesalo a nakonec by agenti vyhynuli. Pokud chceme, aby umělý život dlouhodobě prosperoval, musíme zajistit, aby se do systému v průběhu času dostávala energie. V prostředí se tedy musí vyskytovat agenti představující autotrofní organismy. Tuto roli mohou plnit pasivní agenti.

Tento problém lze řešit i tak, že ve virtuálním světě samovolně vyrůstá potrava (která se nepočítá mezi agenty - obyvatele prostředí - ale je součástí prostředí). V našem modelu s použitím kast by však tento způsob byl příliš svazující. Chceme umožnit uživateli navrhnout své vlastní druhy potravy pro agenty - např. více druhů potravy, různá výživnost těchto druhů potravy, různá rychlost růstu atd. Při použití našeho návrhu je také možné, aby stejná potrava byla pro některé agenty prospěšná a pro jiné nestravitelná nebo dokonce jedovatá (energie získaná z potravy může být záporná).

3.3 Akce agentů

Zde vycházíme z rozboru provedeného v kapitole Analýza, kde jsme navrhli interakce mezi agenty. Tyto interakce nyní popíšeme konkrétně jako akce agentů. Agenti mohou vykonávat tyto akce:

- Move
- Turn Left
- Turn Right
- Eat
- Reproduce
- Do Nothing

Všechny akce jsou deterministické - jejich efekty jsou jasně definované, výsledek akce je vždy stejný.

Pro chování agentů je použité reaktivní plánování t.j. agent volí vždy pouze následující akci, neplánuje dopředu a neudržuje žádnou reprezentaci znalostí. Volba další akce je závislá na bezprostředním okolí agenta a na vnitřním stavu agenta, který tvoří množství energie a stáří; agenti nemají paměť. Každý agent má seznam pravidel, podle kterých vybírá další akci. Každé takové pravidlo má své předpoklady (kdy ho lze použít) a výsledek (akce, kterou agent zvolí při použití tohoto pravidla).
Předpokladem pravidla je dvojice :

1. Vzor okolí - jaký má být obsah příslušných okolních políček, aby bylo možné pravidlo použít. Může obsahovat tzv. Wildcard symboly - t.j. na obsahu příslušného pole nezáleží.

2. Vnitřní stav agenta - množství energie a věk agenta

Agentovy senzory jsou schopné zjišťovat obsah okolních políček jen do té míry, že rozeznají, k jaké kastě patří agent, nacházející se na daném políčku. Tyto vjemy se také používají v předpokladech pravidel. Agentovy senzory tedy nerozlišují mezi jednotlivými příslušníky stejné kasty, ani nepodávají informace o vnitřním stavu pozorovaného agenta (jeho energie a stáří).

Nyní popíšeme strukturu pravidel formálně. Nejprve zavedeme potřebné značení.

- Pravidla budeme označovat písmeny řecké abecedy φ, ψ, π
- Množinu všech akcí, které můžou agenti vykonávat označme A
- Kasty vyskytující se v simulaci budeme označovat symboly K_1 až K_m, kde m je počet použitých kast. Množinu všech kast vyskytujících se v simulaci označíme K.

$$K = \{K_1, ..., K_m\}$$

- Množinu všech agentových vjemů označme V.

$$V = \{(v_1, ..., v_n) | v_i \in K \cup \{K_0\}\}$$

  n označuje počet políček, která agent vidí, K_0 reprezentuje prázdné políčko.

- Každé pravidlo má požadavek na okolí. Množinu všech požadavků na okolí můžeme popsat takto

$$B = \{(b_1, ..., b_n) | b_i \in K \cup \{K_0\} \cup \{WildCard\}\}$$

  Symbol $WildCard$ na i-tém místě znamená, že na obsahu tohoto pole nezáleží.

- Požadavek na okolí ve skutečnosti popisuje podmnožinu V (všechny vjemy, které vyhovují požadavku). Přesněji každé $b \in B$ reprezentuje nadrovinu ve V. Množinu vjemů, které požadavku $b \in B$ vyhovují, označme $\rho(b)$

$$\rho(b) = \{(v_1, ..., v_n) | (v_1, ..., v_n) \in V \wedge (v_i = b_i \vee b_i = WildCard)\}$$

  pro $b = (b_1, ..., b_n) \in B$.

- Množinu možných vnitřních stavů agenta označme S.
- Pravidlo pak můžeme definovat jako trojici:

$$\varphi = (b, s, a), b \in B, s \in S, a \in A$$

- Dále zaveďme značení $Dom(\varphi)$, které u pravidla φ označuje množinu vjemů, na které toto pravidlo lze použít.

$$Dom(\varphi) = Dom(b, s, a) = \rho(b)$$

Nyní popíšeme množinu vnitřích stavů agenta. Jedním z požadavků popsaných v sekci Analýza je, aby rozhodovací mechanismus agentů byl co možná nejjednodušší a paměťově nenáročný. Nabízí se proto otázka, zde se nemůžeme obejít bez stavu a další akci vybírat pouze na základě agentova vjemu.

Tento způsob by však měl velmi negativní dopad na vyjadřovací sílu agentova programu. Například pokud by agent viděl potravu, vždy by jedl bez ohledu na to, jaká je jeho zásoba energie. To mu se chceme vyhnout, proto by rozhodování mělo záviset na hodnotě energie. Déle bychom považovali za rozumné, aby se agent, jehož věk je vysoký (blíží se maximální hodnotě), věnoval spíše rozmnožování, než hromadění energie. K tomu je třeba, aby rozhodování záviselo také na aktuálním věku.

Z těchto důvodů zavádíme alespoň jednoduchý vnitřní stav agentů. Možné stavy jsou 4:

- málo energie, nízký věk
- málo energie, vysoký věk
- hodně energie, nízký věk
- hodně energie vysoký věk

Výraz "málo" nebo "nízký" znamená, že ukazatel je menší než polovina maximální hodnoty, "hodně" a "vysoký" pak že ukazatel je větší nebo roven polovině maximální hodnoty.

Taková volba stavů, kdy pouze porovnáváme aktuální a maximální hodnotu, je pro náš model velmi vhodná. Pokud bychom použili konkrétní hodnoty (jako například: "$Energie \leq 20$"), nastávaly by problémy. Maximální hodnotu energie agenta totiž zadává uživatel, a není proto dobře možné definovat stavy dopředu pomocí konkrétních hodnot. Dalším problémem by bylo, že při experimentech s odděleným pohlavím mohou být hodnoty maximálního věku a energie pro obě pohlaví různá. Pokud by tedy stav byl postavený na konkrétních hodnotách, potomci by dostávali pravidla od obou rodičů, ale konkrétní hodnoty ve stavech by pro ně měly jiný význam (měli by jiný maximální věk a energii než jeden z rodičů). Při zavedení stavů relativně pomocí maximální hodnoty se těmto problémům vyhneme.

Pravidlo, jehož požadavek na okolí obsahuje větší množství Wildcard symbolů označujeme jako obecnější. Naopak pravidlo, jehož požadavek na okolí obsahuje menší množství Wildcard symbolů označujeme jako specializovanější. Tuto vlastnost můžeme formálně popsat pomocí relace "být obecnější", kterou budeme označovat $\succ$ a definujeme ji takto:

$$\varphi \succ \psi \Leftrightarrow |Dom(\varphi)| \geq |Dom(\psi)|$$

Při výběru další akce použije agent vždy nejspecializovanější z pravidel, která lze použít. Každý agent má jednu akci označenou jako předvolenou (default). Tuto akci zvolí, pokud žádné z pravidel v jeho seznamu nelze v dané situaci použít.

Jedná se tedy o prioritní if - then pravidla, kde pravidla s menším počtem Wildcard symbolů mají vyšší prioritu. Pravidla se vyhodnocují seriově od těch s nejvyšší prioritou, použije se první, které uspěje. Defaultní akce zajišťuje pokrytí celého prostoru agentových vjemů, uspěje vždy, ale má nejnižší prioritu.

Tento přístup umožní, aby rozhodování agentů bylo založeno na několika všeobecných pravidlech a výjimkách z těchto pravidel, které popisují konkrétní množiny situací, z nějakého důvodu důležité.

Výše popsané řešení jsme zvolili, protože se hodí pro použití v našich podmínkách, lze pro ně snadno navrhnout operace křížení a mutace, je jednoduché na implementaci a poměrně úsporné na paměť. Navíc vybrání další akce spočívá pouze v jednom průchodu sady pravidel konkrétního agenta a není tedy nijak náročné na výkon a tudíž rychlé.

3.4 Modelování evoluce

V umělém světě modelujeme evoluci agentů. Předmětem evoluce je pouze rozhodování čili sada pravidel agentů. Evoluce je založena na přenosu genomu z rodičů na potomka při rozmnožování, kde genomem je v tomto případě sada pravidel. Agenti se mohou rozmnožovat dvěma způsoby: pohlavně a nepohlavně. Při pohlavním rozmnožování se genom potomka sestaví jako kombinace genomu rodičů, v případě nepohlavního rozmnožování se genom rodiče okopíruje. V obou případech se při přenosu genomu uplatňuje mutace. Předpokládáme, že v prostředí budou přežívat vždy nejlépe přizpůsobení agenti, což povede ke zkvalitňování genomu a sofistikovanějšímu chování agentů.

Evoluci agentů je možné ovlivňovat nastavením hodnot parametrů prostředí. Jedná se zejména o:

- pravděpodobnost mutace
- pravděpodobnost křížení genomu (crossover)
- preference akcí při sestavování nových náhodných pravidel
- další upřesňující parametry mutace.

Tyto parametry nastavuje uživatel před spuštěním experimentu (respektive může změnit jejich hodnoty pokud mu nevyhovuje standardní nastavení) a jsou pro všechny kasty stejné.

Nyní popíšeme průběh evoluce podrobněji: Předpokládáme, že rozmnožovat se budou úspěšní agenti, tedy ti, jejichž sada pravidel je vhodná pro dané prostředí. Agentova rozhodovací tabulka však může obsahovat desítky pravidel, z nichž jen několik je zodpovědných za agentovo chování (ostatní pravidla obsahují takové vzory okolí, které agent ve svém životě nikdy nezaznamenal a tato pravidla tedy nijak neovlivnila jeho chování ani jeho fitness).

Právě pravidla, která úspěšný agent skutečně používá, bychom chtěli přenášet na potomky. Z toho důvodu přidáváme ke každému pravidlu také čítač počtu použití. Nepracujeme tedy pouze s úspěšnými jedinci, ale snažíme se pracovat přímo s úspěšnými pravidly.

Zde se nabízí paralela s teorií neodarwinismu, který prosazuje tzv. genocentrický pohled na evoluci. Tvrdí, že přirozený výběr neprobíhá na úrovni jednotlivců, ale přímo na úrovni genů, a že jsou to geny, kdo "usiluje" o vlastní rozšiřování, a jedince používají pouze jako nástroj své reprodukce. Dále dělí skupiny genů na kooperativní, což jsou takové skupiny, jejichž společný výskyt je prospěšný oběma stranám (zvyšuje šanci na rozšiřování genů) a kompetitivní, které navzájem soupeří o své místo v genomu jedince. V našem případě bychom za kooperativní geny mohli považovat (úspěšná) pravidla, jejichž vzory okolí jsou navzájem disjunktní, a za kompetitivní pak pravidla, jejichž požadavky na okolí se překrývají, ale každé pravidlo doporučuje jinou akci. Více o této problematice se lze dočíst v [1].

Vraťme se ještě k zavedení čítače u pravidel: v naší implementaci se čítač každého pravidla při přenosu na potomka vynuluje a počet použití pravidla se počítá pouze za života jednoho konkrétního agenta. Takový přístup se může ukázat jako nevhodný například v této situaci: pravidlo bylo uspěšné u rodiče, potomek ho zdědil, ale v průběhu svého života se nedostal do situace, kdy by ho mohl použít a tedy čítač tohoto pravidla je roven nule a pravděpodobně se do další generace nedostane, přestože by mohlo být užitečné.

Tomuto bychom mohli předejít, kdybychom u pravidel potomků ponechali čítač na hodnotě, kterou mělo pravidlo v genomu rodiče (nebo bychom hodnotu čítače u rodiče nepřenesli celou, ale například bychom při přenosu genomu na potomky všechny čítače zmenšili na polovinu). Vynulování čítačů u každé další generace znamená, že agenti rychle zapomínají, což by teoreticky mohlo přinést problémy (popsané v předchozím odstavci). Na druhou stranu tato vlastnost posiluje adaptivitu agentů a dává jim schopnost rychleji se přizpůsobovat měnícímu se prostředí.

Obecně můžeme při přenosu na potomka vynásobit hodnoty čítačů libovolným číslem z intervalu [0,1] kde 0 představuje okamžité zapomínání a 1 pak model bez zapomínání. Hodnotu je možné nastavit v konfiguračním souboru, defaultní hodnota je 0 z důvodu větší adaptivity. Zkoumáním optimální hodnoty tohoto parametru se zde zabývat nebudeme.

3.4.1 Průběh křížení

Křížení kombinuje genom dvou rodičů, výsledkem je genom potomka. V tomto programu je použité tzv. rovnoměrné informované křížení. Rovno-

měrné křížení je charakteristické tím, že prochází genomy obou rodičů současně a na každé pozici se rozhoduje, zda potomek dostane gen od otce, nebo od matky. Křížení označujeme jako informované, pokud při kombinaci genomů používá dodatečné znalosti o jedincích.

Konkrétně se křížení provádí takto: pravidla v obou genomech jsou seřazená od nejspecializovanějších po nejobecnější. V tomto pořadí se postupně procházejí. Dvojice pravidel na i-té pozici (od otce a od matky) spolu soupeří o místo na i-té pozici v genomu potomka. Genomy obecně nejsou stejně dlouhé, jakmile u jednoho genomu dojdeme na konec, zbytek druhého genomu se zkopíruje.

Při výběru lepšího z dvojice pravidel se používá ruletový selektor, který využívá informace o počtu použití jednotlivých pravidel. Konkrétně: spočítá se poměr počtu použití obou pravidel a přiřadí se jim pravděpodobnosti odpovídající tomuto poměru. Pro účely výpočtu se čítač počtu použití pravidel bere s hodnotou o jedna větší než skutečně je, aby i pravidla s nulovým počtem použití měla šanci projít výběrem.

Formálně můžeme výběr ze dvou pravidel popsat takto: Mějme pravidla φ a ψ. Funkcí $u(\varphi)$ budeme označovat hodnotu čítače počtu použití pravidla φ. Pravděpodobnost výběru pravidla φ označíme $P(\varphi)$. Její hodnota je:

$$P(\varphi) = \frac{u(\varphi) + 1}{u(\varphi) + 1 + u(\psi) + 1}$$

Nevýhodou rovnoměrného křížení je, že nezachovává souvislé úseky z genomu rodičů. Podle hypotézy o stavebních blocích bychom se měli snažit vyměňovat celé úseky genomu, abychom neporušili části, které spolu logicky souvisejí. V našem případě však pravidla, která se nacházejí v genomu vedle sebe, nemají žádnou přímou spojitost, takže v tomto nám rovnoměrné křížení nevadí.

Mezi pravidly však určité logické vazby jsou. Konkrétně mezi obecným pravidlem, které pokrývá velkou množinu situací (označme ho φ) a specializovaným pravidlem (označme ho ψ) takovým, že $Dom(\varphi) \cap Dom(\psi) \neq \emptyset$. Pak pravidlo ψ popisuje výjimku z obecného pravidla φ.

Vztah mezi těmito pravidly je zřejmý: pravidlo φ je ve skutečnosti úspěšné pouze na množině vjemů $Dom(\varphi) \setminus Dom(\psi)$, zatímco na množině $Dom(\varphi) \cap$

$Dom(\psi)$ úspěšné není a potřebuje proto pravidlo ψ ke správnému fungování. Tato pravidla bychom označili za kooperativní, nebo dokonce za vzájemně závislá. Vyřazení kteréhokoliv z nich může způsobit razantní snížení fitness agenta.

Dalším problémem může být, že některé obecné pravidlo se do genomu potomka nedostane, a část prostoru vjemů tak zůstane nepokrytá. Tato část pak připadne defaultnímu pravidlu, což nemusí být vždy vhodné. Defaultní pravidlo se sice také dědí od rodičů, ale v genomu rodičů mohlo pokrývat zcela jiné části prostoru vjemů než u potomka. Proto i když rodiče byli úspěšní, potomek s touto sadou pravidel úspěšný být nemusí.

Tento jev nemusí nastat pouze u defaultního pravidla, ale obecně u kteréhokoliv. Pravidla jsou v genomech seřazená od nejspecializovanějších a v tomto pořadí se i vyhodnocují. Pravidlo tedy pracuje na takové množině vjemů ze své domény, která není pokrytá předchozími pravidly. V genomu potomka však tato množina může být jiná.

Pokud bychom chtěli těmto problémům předejít, museli bychom při křížení hledat logické vazby mezi pravidly a snažit se je zachovávat. Účelem křížení však není to, aby všichni potomci byli lepší než rodiče, stačí, když bude existovat aspoň jeden potomek lepší než rodiče. Z tohoto důvodu skutečnost, že křížení některé vazby zpřetrhá a jiné vytvoří není na škodu, ale je naopak prospěšná. (A je to typická vlastnost křížení. Pokud bychom chtěli většinu vazeb zachovat, pak je lepší prohledávat lokálně, pouze pomocí mutací.)

Při křížení může nastat situace, kdy se do genomu potomka dostanou dvě pravidla φ a ψ, která obě byla úspěšná u rodičů a jejich požadavky na okolí se překrývají tedy $Dom(\varphi) \cap Dom(\psi) \neq \emptyset$ (nebo dokonce $Dom(\varphi) \subseteq Dom(\psi)$). Pak v závislosti na velikosti množiny $Dom(\varphi) \cap Dom(\psi)$ a četnosti výskytu vjemů se může stát, že jedno z pravidel zastíní druhé a agent toto druhé pravidlo nebude používat (a v další generaci ho zapomene přestože bylo úspěšné). To však nevadí, protože i první z pravidel bylo úspěšné a rozhodování agenta ani jeho potomků to nezhorší.

3.4.2 Průběh mutace

Mutace provede malou změnu v genomu při přenosu z rodičů na potomka. Navrhli jsme několik operací, které mohou být provedeny v rámci mutace. Každá operace má svou pravděpodobnost, jejíž hodnotu může uživatel nastavit v konfiguračním souboru. Jedná se o tyto operace:

1. přidání nového náhodného pravidla
2. smazání pravidla
3. změna předpokladů u některého pravidla
4. změna akce u některého pravidla

Při provádění mutace je často třeba vygenerovat novou náhodnou akci agenta. Nové akce se generují z pravděpodobnostního rozdělení definovaného uživatelem - pravděpodobnosti jednotlivých akcí je možné nastavit v konfiguračním souboru.

Při výběru pravidla pro mutaci se uplatňuje čítač počtu použití pravidel. Cílem je mimo jiné snižovat počet pravidel, která agent nepoužívá a tedy mu nijak nepomáhají. Děje se tak pomocí operací 2 a 3: operace 2 smaže náhodné pravidlo s malým počtem použití a operace 3 vybere pravidlo s malým počtem použití a změní jeho předpoklady. Tato změna předpokladů preferuje zobecňování pravidel t.j. přidávání tzv. wildcard symbolů. Operace 1 a 4 mají za cíl nacházet nová, zatím neznámá pravidla.

3.5 Zavedení kast

Kasty zavádíme proto, aby bylo možné agenty rozdělit na několik skupin(druhů) tak, že některé parametry agentů budou pro stejné v rámci skupiny, ale jednotlivé skupiny se budou v hodnotách těchto parametrů navzájem lišit. Jak bylo rozebráno v předchozí kapitole, bylo by možné navrhnout mnoho parametrů, ve kterých se jednotlivé kasty budou lišit. Pro naše účely jsme zvolili tyto parametry:

- Maximální zásoba energie
- Maximální věk
- Zda je agent aktivní/pasivní
- Ztráta energie za jeden krok simulace
- Množství energie předávané potomkům (absolutní nebo procentuální hodnota)
- Délka těhotenství
- Způsob rozmnožování (pohlavní/nepohlavní)

- Partner pro rozmnožování (kasta)
- Věk pohlavní dospělosti
- Kořist - kasty, jejichž příslušníky může tento agent sežrat a kolik energie tím získá
- Ceny za jednotlivé akce - ztráty energie při vykonávání akcí
- Barva při zobrazování průběhu simulace

Kasty budou definované uživatelem. Vytvoření nové kasty spočívá v nastavení hodnot těchto parametrů v konfiguraci kasty. Některé parametry rozebereme nyní podrobněji a přitom vysvětlíme detailněji i fungování programu:

Maximální zásoba energie Množství agentovy energie nemůže přesáhnout tuto hodnotu. Pokud by agent získal další energii (přes limit), energie se mu nepřičte, zůstane na původní hodnotě. Množství energie, které agent dokáže uchovávat, ovlivňuje také další agenty. Konkrétně jsou to predátoři, kteří mohou tohoto agenta sežrat. Množství energie, které tím získají, je omezené množstvím energie kořisti (která je závislá na agentově maximální energii).

Hodnota tohoto parametru ovlivňuje také agentovo rozhodování. Jak jsme popsali výše, je volba akce agenta závislá na jeho vnitřním stavu. Vnitřní stav agenta obsahuje informaci o tom, zda je aktuální hodnota agentovy energie větší, nebo menší než polovina maximální hodnoty. Z toho důvodu tedy hodnota maximální energie ovlivňuje agentovo rozhodování.

Maximální věk Tento parametr se v mnohém podobá předchozímu. Stejně jako maximální zásoba energie i maximální věk ovlivňuje agentovo rozhodování. Součástí vnitřního stavu agenta je totiž informace, zda agentův věk překročil polovinu maximální hodnoty, nebo ne. Rozdíl mezi těmito parametry spočívá v tom, že pokud agentův věk dosáhne maximální hodnoty, agent zemře.

Ztráta energie za jeden krok simulace Vyjadřuje, kolik energie agent ztratí v každém kroku simulace (navíc k ceně za akci). Tento parametr je redundantní, mohli bychom se bez něj obejít, pokud bychom jeho hodnotu přičetli ke všem cenám za akce. Přístup, kdy tento parametr máme zvlášť, však usnadňuje uživateli navrhování kast. (Je jednodušší měnit jeden parametr, než pozměňovat ceny za všechny akce.)

Energie předávaná potomkům Při přenosu energie mezi agenty je dodržován zákon zachování energie, tedy příjemce nemůže získat víc energie, než ztratí výdejce. Zde je možné nastavit konkrétní množství energie, které rodič potomkovi předá (pokud má rodič méně, předá veškerou svou energii a sám zemře). Jinou možností je nastavit procentuální hodnotu, rodič potom předá množství energie závislé na své vlastní hodnotě.

Délka těhotenství Uplatňuje se pouze u kast, které používají pohlavní rozmnožování. Délka těhotenství vyjadřuje počet kroků simulace, kdy agent, který otěhotněl, nebude vykonávat žádné akce. Teprve po uplynutí této doby vytvoří potomka a předá mu část energie. Těhotenství lze u agentů deaktivovat tak, že tento parametr bude nastaven na nulu, v tom případě vytvoří potomka ihned po rozmnožování.

Způsob rozmnožování Hlavní rozdíl mezi pohlavním a nepohlavním rozmnožováním spočívá v tom, že při pohlavní reprodukci dochází ke křížení i mutaci, zatímco při nepohlavní se uplatňuje pouze mutace. Dalším rozdílem je, že u agentů používajících pohlavní rozmnožování jsou k přežití kasty potřeba aspoň dva jedinci. Některé další parametry se uplatňují pouze při pohlavním rozmnožování.

Partner pro rozmnožování Tento parametr má smysl pouze při pohlavním rozmnožování. Lze pomocí něj simulovat oddělené pohlaví agentů. Při vytváření potomka se jeho kasta vybere náhodně buď jako kasta otce, nebo matky. Pokud pro účely experimentu není třeba pohlaví agentů oddělovat, lze nastavit kastu partnera na stejnou hodnotu jako je kasta agenta.

Věk pohlavní dospělosti Pro každou kastu je možné nastavit věk pohlavní dospělosti, jedná se o počet kroků simulace, po který se nově vzniklý agent nemůže rozmnožovat. Pokud tento jev nechceme v experimentu použít, je možné nastavit parametr na nulu.

Kořist Zde jsou definovány interakce s ostatními agenty. Je zde seznam kast, jejichž příslušníky může tento agent sežrat a u každé z nich i množství energie, které tím získá (vždy absolutní hodnota, ale predátor může získat nejvýše tolik energie, kolik měla kořist). Je tak možné definovat potravní řetězce, více druhů potravy pro agenty, různá výživnost této potravy pro různé kasty a podobně. Energie získaná z potravy může být nastavena na zápornou hodnotu, tedy potrava může být pro některé agenty jedovatá.

Ceny za akce Je zde seznam akcí agentů a u každé akce množství energie, které agent zaplatí za vykonání akce. Energie se odečítá už při pokusu o vykonání akci bez ohledu na to, zda byla úspěšně dokončena, nebo ne. Například pokud se agent rozhodne vykonat akci Eat, bude mu okamžitě odečteno příslušné množství energie a teprve pak se zjišťuje, jestli agent může tuto akci vykonat, tedy jestli se před ním nachází kořist, kterou může sežrat. Ceny za akce mohou být i záporné, lze tedy navrhnout experimenty, kdy vykonáváním některé akce budou agenti získávat energii. Toto se dá použít při návrhu experimentů, které nemají biologický základ.

Takovýmto způsobem můžeme modelovat soužití několika různých druhů agentů ve virtuálním světě. Jednotlivé kasty mohou představovat např. různé druhy organismů (druh v biologickém smyslu) nebo příslušníky různého pohlaví. Agenti patřící do stejné kasty mohou představovat organismy, které mají stejnou stavbu těla. Ve virtuálním světě je "stavba těla" bytostí reprezentovaná výše uvedenými parametry. Hodnoty těchto parametrů jsou po celou dobu simulace neměnné (Evoluci podléhá jen rozhodování agentů). Další atributy jako zásoba energie, věk a pozice agenta jsou proměnlivé a představují agentův stav. Sada pravidel pak reprezentuje agentovu individualitu - ovlivňuje jeho chování a částečně přechází na potomky.

Je tedy možné do jisté míry napodobit život skutečných organismů, ale samozřejmě lze navrhnout i experimenty, které nejsou motivované biologicky.

Ve scénářích, kde se vyskytují kasty představující různé pohlaví, mohou nastat problémy: Při kombinaci genomu agentů, kteří mají oddělené pohlaví bude mít potomek aspekty chování obou rodičů. Lze si však představit, že rodiče se budou chovat odlišně, např. budou se živit každý jinou potravou. Potomek pak bude mít "fyzicky" jedno pohlaví, ale chování bude odpovídat částečně oběma pohlavím - např. bude se snažit jíst potravu, která pro něj není stravitelná. Je tedy třeba dbát zvýšené opatrnosti při návrhu experimentů s odděleným pohlavím, zejména snažit se, aby si obě pohlaví byla aspoň částečně podobná.

Kapitola 4

Experimenty

Shrnutí: V této kapitole představíme sadu experimentů, na kterých ukážeme možnosti použití tohoto programu. U každého experimentu bude popsáno:

- Jakou situaci chceme modelovat a proč
- Počáteční podmínky
- Průběh a výsledky simulace
- Diskuze nad očekávanými výsledky a výsledky skutečně pozorovanými
- Naše vysvětlení zajímavých jevů, které se při simulaci objevily

U každého experimentu provádíme deset pokusů, prezentujeme jeden nebo dva reprezentativní výsledky. Výsledky všech experimentů i soubory potřebné pro jejich nové provedení se nacházejí na přiloženém DVD.

4.1 Experiment 1: Schopnost učení agentů

Popis experimentu: V tomto experimentu chceme ukázat schopnost agentů přizpůsobit se podmínkám prostředí. V umělém světě budou k dispozici dva druhy potravy, z nichž jedna bude pro agenty prospěšná a druhá jedovatá. Předpokládáme, že agenti se naučí rozpoznat jedovatou potravu a budou se jí vyhýbat.

Použité kasty:

- Ovce - aktivní, bílá barva

- Tráva - pasivní, zelená barva
- Jedovatá tráva - pasivní, červená barva

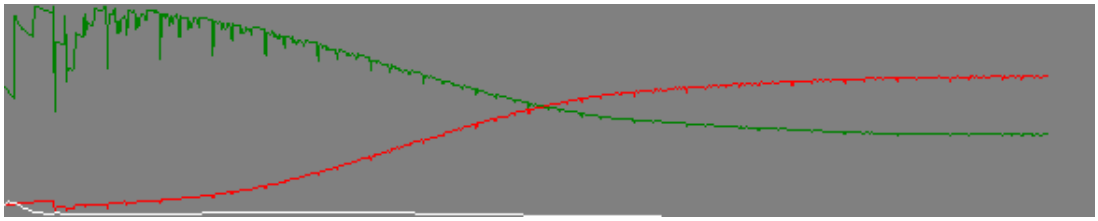
Počáteční podmínky: Velikost světa: 250 x 200 polí

- Ovce - počáteční počet: 3000
- Tráva - počáteční počet: 30000
- Jedovatá tráva - počáteční počet: 3000

Ovce mohou žrát oba druhy trávy, zelená tráva jim dodává energii, červená jim energii odebírá. Oba druhy trávy jsou co se týče ostatních parametrů naprosto totožné.

Výsledky: Provedli jsme deset pokusů. Simulaci jsme prováděli jen tak dlouho, dokud ovce přežívaly. Průměrný počet kroků byl 1047,1 nejdelší simulace trvala 1287 kroků, nejkratší 813 kroků.

Všech deset pokusů skončilo velmi podobným výsledkem, který reprezentuje obrázek 4.1. Jedovatá tráva postupně vytlačovala zelenou a jakmile množství zelené trávy kleslo pod určitou mez, ovce vyhynuly.



Obrázek 4.1: Počet agentů jednotlivých kast v čase

Zhodnocení výsledků: Agenti se v průběhu simulace skutečně naučili vyhýbat se jedovaté potravě. Tím, že spásali pouze zelenou trávu, však poskytli výhodu jedovaté trávě, která se rozšiřovala a vytlačovala tak obyčejnou trávu. Přestože na začátku bylo množství potravy 10:1 ve prospěch obyčejné trávy, postupem času jedovatá tráva převážila. Agenti tím přišli o potravu a vyhynuli. Zdá se, že takto vytvořený systém nelze dlouhodobě udržet.

4.2 Experiment 2: Hledání potravy v obtížných podmínkách

Popis experimentu: Tento experiment navazuje na předchozí. Snažíme se upravit systém z předchozího pokusu tak, abychom mohli umělý život dlouhodobě udržet. Zvýšíme zelené trávě rychlost růstu a rozmnožování, což by mělo kompenzovat její spásání ovcemi. Ostatní parametry jsou stejné jako v předchozím případě.

Použité kasty:

- Ovce - aktivní, bílá barva
- Tráva - pasivní, zelená barva
- Jedovatá tráva - pasivní, červená barva

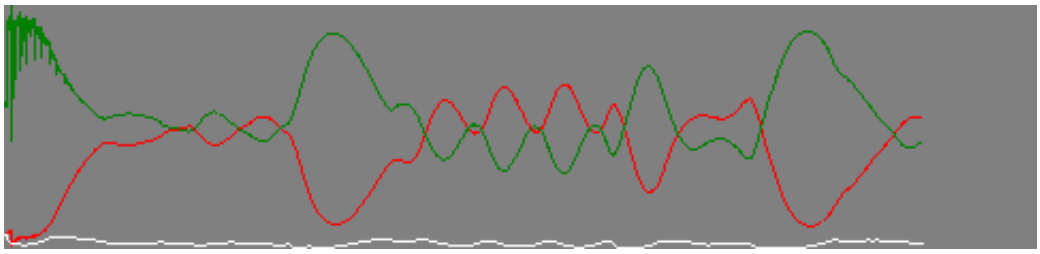
Počáteční podmínky: Velikost světa: 250 x 200 polí

- Ovce - počáteční počet: 3000
- Tráva - počáteční počet: 30000
- Jedovatá tráva - počáteční počet: 3000

Ovce mohou žrát oba druhy trávy, zelená tráva jim dodává energii, červená jim energii odebírá. Zelená tráva má oproti červené výhodu rychlejšího rozmnožování. Zvolili jsme počáteční poměr zelené a červené trávy 10:1 což by mělo dát ovcím dostatek času přizpůsobit se k hledání správné potravy. Předpokládáme, že tento poměr se bude postupně měnit ve prospěch jedovaté trávy.

Výsledky: Opět jsme provedli 10 pokusů s limitem 5000 kroků. Ve všech 10 případech dokázali příslušníci všech tří kast přežít ve virtuální prostředí po celou dobu simulace, tedy 5000 kroků.

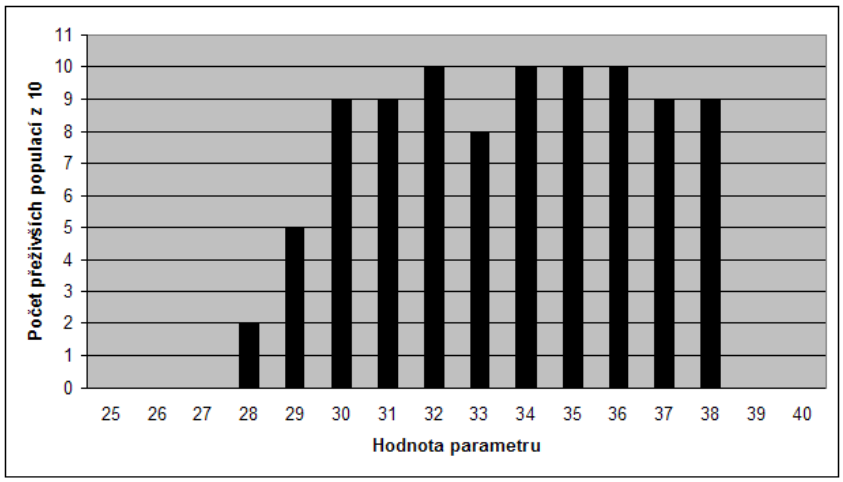
Vývoj počtu agentů v prostředí zobrazuje obrázek 4.2.



Obrázek 4.2: Počet agentů jednotlivých kast v čase

Zhodnocení výsledků: Po počátečním přizpůsobení přešel systém do stavu, který se poté periodicky opakoval. Při malém počtu ovcí se zelená tráva rozšiřovala díky svému rychlejšímu růstu oproti jedovaté trávě. Větší množství potravy pak způsobilo rozmnožení ovcí, větší spásání zelené trávy a tím i její úbytek. Převážila opět jedovatá tráva, ovce přišly o potravu a jejich počet se zmenšil, zelená tráva se začala znovu rozšiřovat… atd. Systém dospěl do dynamické rovnováhy.

Náš cíl, tedy trvalé udržení života, se podařilo splnit. Bylo k tomu potřeba přesné nastavení parametrů určujících rychlost růstu jedovaté trávy. Pokud byla rychlost růstu obou druhů trávy stejná nebo velmi podobná, život zanikl jako v experimentu 1. Pokud výhoda zelené trávy byla naopak příliš velká, jedovatá tráva nedokázala konkurovat a vyhynula dříve, než se ovce stačily rozmnožit. Pouze malý interval hodnot parametrů zajišťuje přežití všech tří kast. O tomto jevu se ještě zmíníme v části věnované navrhování experimentů.



Obrázek 4.3: Počet přežívajících populací v závislosti na hodnotě parametru.

Jedovaté trávě jsme upravili parametr určující maximální věk. Snížením jeho hodnoty jsme jedovatou trávu znevýhodnili tak, aby jí zelená tráva mohla konkurovat i při spásání ovcemi. Zelená tráva má délku života 40 kroků, u jedovaté trávy jsme zvolili hodnotu 35 kroků. Závislost výsledku simulací na hodnotě tohoto parametru ukazuje obrázek 4.3. Je v něm zobrazeno, kolik populací z deseti dokázalo přežít aspoň 2000 kroků v závislosti na hodnotě parametru délky života jedovaté trávy. Lze si všimnout skokové změny mezi hodnotami 38 a 39. Pro všechny ostatní hodnoty, které v grafu nejsou zachycené, je počet přeživších populací nulový.

4.3 Experiment 3: Potravní řetězec

Popis experimentu: V tomto experimentu chceme vytvořit jednoduchý potravní řetězec. Také zde předvedeme některé možnosti programu, které v předchozích experimentech ještě nebyly použité.

Simulujeme ekosystém bakterií, všechny kasty se zde rozmnožují nepohlavně. Dále zde představíme aktivní agenty, kteří reprezentují autotrofní organismy - postupem času samovolně získávají energii. (Předchozí experimenty byly navrženy tak, že pouze pasivní agenti, představující potravu, mohli samovolně získávat energii, zatímco aktivní agenti energii postupem času ztráceli).

Použité kasty:

- bacteryFood - pasivní, modrá barva
- Chlorobacteria - aktivní, žlutá barva
- BacterioFag - aktivní, červená barva

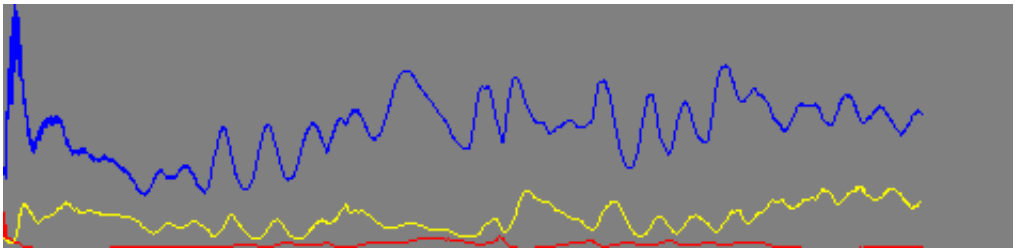
Energetická bilance je navržená tak, že příslušníci kasty Chlorobacteria získávají samovolně energii a mohou tedy přežívat bez potravy, ovšem cena za rozmnožování je příliš velká na to, aby se mohli bez potravy rozmnožovat. K rozmnožování tedy potřebují přijímat potravu, kterou zde představuje kasta bacteryFood.

Příslušníci kasty BacterioFag mohou požírat obě zbylé kasty. Hodnoty energie jsou nastavené tak, že požíráním bacteryFood mohou nějakou dobu přežívat, ale cena za rozmnožování je opět velká, takže pokud se chtějí rozmnožovat, musejí požírat kastu Chlorobacteria, která je pro ně výživnější. Tím by měli vytvářet potravní řetězec.

Počáteční podmínky: Velikost světa: 250 x 200 polí

- bacteryFood - počáteční počet: 8000
- Chlorobacteria - počáteční počet: 3300
- BacterioFag - počáteční počet: 1700

Výsledky: Provedli jsme 10 pokusů, s limitem 5000 kroků a předčasným ukončením, pokud některá z kast vyhyne. Tři pokusy skončily předčasně vyhynutím kasty BacterioFag, ve zbylých sedmi případech se život udržel po celou dobu simulace. Jeden z těchto úspěšných případů zobrazuje obrázek 4.4.

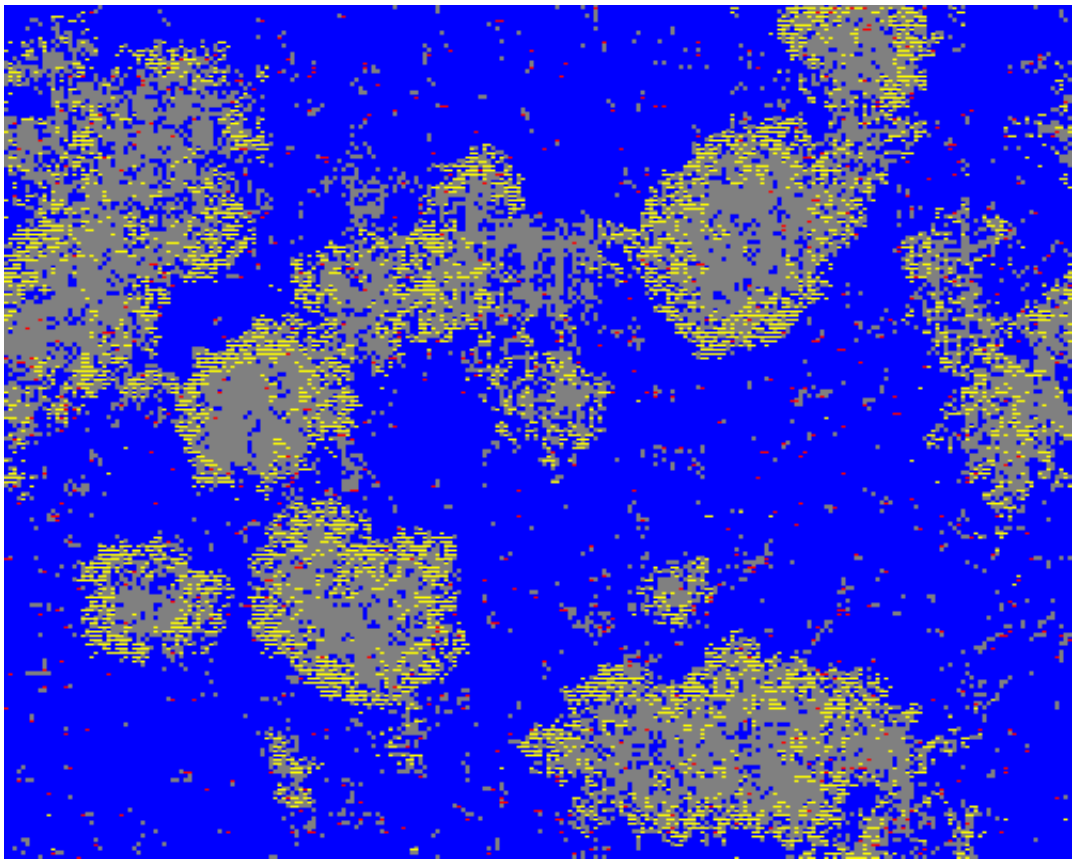


Obrázek 4.4: Počet agentů jednotlivých kast v čase

Zhodnocení výsledků: Podařilo se splnit vytyčený cíl a vytvořit jednoduchý potravní řetězec. Při experimentu s agenty, kteří používají nepohlavní rozmnožování se objevilo několik zatím nepozorovaných jevů. Patří k nim zejména velká rychlost rozmnožování agentů, pokud narazí na potravu, která připomíná exponenciální množení bakterií v příznivých podmínkách. Tento jev je patrný na obrázku 4.5. (Obrázek zachycuje stav prostředí v jednom okamžiku simulace. Celý průběh simulace je možné prohlížet ze souboru na přiloženém DVD.)

4.4 Experiment 4: Oddělené pohlaví

Popis experimentu: V tomto experimentu vytvoříme situaci, kdy rozmnožování probíhá mezi dvěma různými kastami. Tyto kasty budou představovat příslušníky odlišného pohlaví a mohou se v některých parametrech lišit. Dále zde představíme parametr určující délku těhotenství, který jsme v předchozích experimentech zatím nepoužili.



Obrázek 4.5: Agenti (žlutá barva) pojídající potravu (modrá barva), za sebou nechávají prázdné místo (šedá barva), při kontaktu s potravou se rychle množí (na okraji šedých ploch).

Použité kasty:

- Tráva - pasivní, zelená barva
- Jedovatá tráva - pasivní, červená barva
- Ovce samec - aktivní, bílá barva
- Ovce samice - aktivní, žlutá barva
- Vlk - aktivní, černá barva

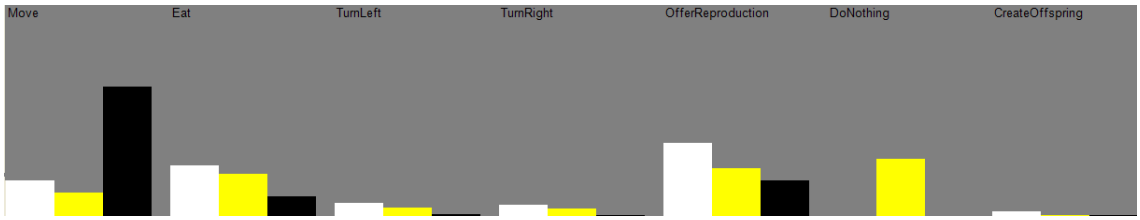
Počáteční podmínky: Velikost světa: 250 x 200 polí

- Tráva - počáteční počet: 20000
- Jedovatá tráva - počáteční počet: 1000
- Ovce samec - počáteční počet: 2000
- Ovce samice - počáteční počet: 2000
- Vlk - počáteční počet 700

Vztah trávy a ovcí je stejný jako v experimentu 1, vlk může žrát ovce (samce i samice). Rozdíl mezi oběma pohlavími je v tom, že samice ovcí používají parametr pro délku těhotenství. Konkrétní hodnota je 30 kroků.

Výsledky: Opět jsme provedli deset pokusů na maximální počet 5000 kroků s přerušením pokud některá kasta vyhyne. V šesti pokusech se život udržel po celých 5000 kroků, čtyři pokusy skončily předčasně vyhynutím kasty vlků.

Zhodnocení výsledků: V tomto experimentu se podařilo udržet umělý život pouze v šesti případech. Souvisí to s počtem kast, které se v experimentu vyskytují. Čím větší je počet zúčastněných kast, tím těžší je najít přesné nastavení parametrů, které zajišťuje jejich přežití (a také počet parametrů je větší).



Obrázek 4.6: Akce, které agenti preferují.

Zajímavé jevy můžeme v tomto experimentu sledovat zejména při pohledu na histogram akcí, které agenti vykonávali, zobrazeném na obrázku 4.6. Obrázek ukazuje procentuální podíl akcí agentů jednotlivých kast (samozřejmě zobrazujeme pouze aktivní kasty). Na první pohled je patrné, že vlci vykonávali akci "Move" mnohem častěji než ostatní akce. Je to způsobené tím, že pro vlky je poměrně obtížné najít potravu a musejí ji hledat intenzivněji. V tom se tento experiment liší od předchozího, kde kasta predátorů představovala všežravce, kteří se mohli živit jakoukoli potravou. Naproti tomu vlk může pojídat pouze ovce, jejichž výskyt v prostředí je méně častý.

Tomu odpovídá i menší zastoupení akce "Eat" u vlků než u ostatních kast, pro které je hledání potravy poměrně snadné. Jelikož u vlků je příjem energie z potravy méně častý, musejí tomu odpovídat i ceny akcí a energie získaná z potravy je v případě vlků větší.

Dále je z obrázku patrné, že samice ovcí vykonávají často akci "Do Nothing". To je způsobené nastavením délky těhotenství u této kasty na poměrně vysoké číslo 30 (po dobu těhotenství agent vykonává pouze akci "Do Nothing"). Ostatní kasty tuto akci nepreferují, protože jim nepřináší žádný užitek.

Můžeme si všimnout také různého podílu akce "Offer Reproduction" u jednotlivých kast. Samci ovce vykonavají tuto akci poměrně často, protože jim v tom nic nebrání a tato akce jim přináší užitek. Samice mají podíl této akce nižší, protože tráví mnoho času ve stavu těhotenství, kde nemohou vykonávat akce. Ještě nižší hodnota je u vlků, kde je příčina v tom, že pro vlka je těžší získat dostatek energie k rozmnožování. Přesněji řečeno každému zisku energie předchází větší počet akcí "Move" (hledání potravy) a tedy procentuální zastoupení akce "Offer Reproduction" je menší.

4.5 Navrhování experimentů

V této části se budeme věnovat navrhování simulací, konkrétně tomu, jak správně nastavit parametry kast a počty agentů tak, aby simulace splnila svůj účel. Vysvětlíme, jak k tomu lze využít informace, které program poskytuje.

Jak jsme již zmiňovali u experimentu 2, je výsledek simulace silně závislý na počátečních podmínkách. Počátečními podmínkami myslíme jak nastavení parametrů kast, tak počáteční počet agentů. Příčina tohoto jevu spočívá ve fungování evolučního algoritmu, kdy úspěšní jedinci zvětšují svůj počet geometrickou řadou, zatímco neúspěšní jedinci stejně rychle vymírají. (Samozřejmě to, kteří jedinci jsou úspěšní, se v průběhu času mění.)

Počáteční podmínky (zejména hodnoty parametrů kast) určují, jakou roli bude daná kasta zastávat v prostředí a ovlivňují i to, jak úspěšně bude konkurovat ostatním agentům. Každá změna těchto parametrů ovlivňuje nejen příslušnou kastu, ale i všechny ostatní kasty vyskytující se v prostředí. I když jiná hodnota parametrů změní konkurenceschopnost a vztahy mezi kastami jen minimálně, tyto rozdíly se v průběhu simulace budou zvětšovat exponenciálně a drobná změna parametrů tak může vyústit ve zcela odlišný výsledek simulace.

V tomo směru jsou citlivé zejména parametry určující energetické bilance t.j. zisk energie z potravy a ceny akcí. Nastavení těchto hodnot je proto třeba věnovat zvýšenou pozornost. Jak bylo popsáno výše, při přenosech energie se uplatňuje zákon zachování energie (příjemce získá nejvýše tolik energie, kolik ztratí výdejce). Toho je možné využít a spočítat údaje, která při výběru parametrů mohou pomoci.

Uvažme tento příklad: v prostředí se vyskytují 3 kasty, energetické bilance jsou následující

- Tráva
  - energyLostPerTick = -8 (získává energii)
  - maxEnergy = 200
- Ovce
  - energyLostPerTick = 5 (ztrácí energii)

  - maxEnergy = 500
  - sežrání trávy: +200 (ale nejvýše tolik, kolik měla kořist)
  - Akce Move: 6
  - Akce Eat: 5
  - Akce Turn Left: 3
  - Akce Turn Right: 3
  - Akce Offer Reproduction: 5
  - Akce Create Offspring: 15
  - Akce Do Nothing: 1

- Vlk
  - energyLostPerTick = 3 (ztrácí energii)
  - maxEnergy = 1000
  - sežrání ovce: +500 (ale nejvýše tolik, kolik měla kořist)
  - Akce Move: 3
  - Akce Eat: 3
  - Akce Turn Left: 1
  - Akce Turn Right: 1
  - Akce Offer Reproduction: 5
  - Akce Create Offspring: 15
  - Akce Do Nothing: 1

Z těchto hodnot můžeme určit například poměr počtu agentů trávy a ovcí. Nejprve zaveďme značení:

- x - současná energie agenta
- $ztraty_n$ - energie, kterou agent vydá za n kroků simulace
- $zisk_n$ - energie, kterou agent přijme za n kroků simulace

Nyní spočteme, jaký by měl být minimální energetický zisk agenta představujícího ovci za n kroků simulace. Pokud má agent přežít n kroků, pak musí být $x > ztraty_n$. Pokud mají agenti přežívat, musejí se rozmnožovat. Pro energii z toho plyne vztah

$$x - ztraty_n + zisk_n = 2x$$

Což odpovídá tomu, že za n kroků agent zdvojnásobí hodnotu své energie a poté se rozmnoží.
$ztraty_n$ můžeme odhadnout na základě cen akcí, které předpokládáme, že agent bude vykonávat, a hodnoty parametru energyLostPerTick. $ztraty_n \approx (5+6)n+15$ kde 5 představuje energyLostPerTick, 6 cenu za akci (v každém kroku) a 15 cenu akce Create Offspring (nakonec). Z uvedených vztahů plyne

$$zisk_n \geq 22n + 15$$

Jelikož Tráva získává energii rychlostí 8 za 1 krok (tedy $8n$ za n kroků), z vypočteného vztahu plyne, že každá ovce potřebuje ke svému přežití nejméně 3 agenty představující Trávu, tedy poměr počtu trávy a ovcí bude nejméně $3 : 1$.

Podobným způsobem bychom mohli určit vztah pro počet vlků a ovcí nebo vlků a trávy (protože tráva je jediným zdrojem energie v tomto systému). Lze popsat i vztahy pro další parametry například maximální věk by měl být větší než n (kde n je počet kroků za který agent zdvojnásobí hodnotu své energie).

Dále je možné zjistit maximální počet agentů, kteří jsou schopní se v prostředí uživit. Toto číslo můžeme odhadnout, když porovnáme celkové výdaje všech agentů a celkové množství energie, které do prostředí přichází. Takové výpočty však budou fungovat pouze v jednoduchých modelech, kdy máme jen jeden zdroj energie. Pokud bude v simulaci více způsobů jak se do prostředí může dostávat energie (například některé ceny za akce budou záporné), pak se podobné výpočty nedají jednoduše použít.

Stejný problém nastává, když je v experimentu použito větší množství kast a jejich vztahy jsou komplikovanější (například více druhů potravy). V takových případech jsou odhady složité na výpočet a poměrně nepřesné.

Zde je možné určit hodnoty parametrů experimentálně. Program při prohlížení výsledků simulace poskytuje informace, které mohou při úpravě hodnot parametrů pomoci. Jedná se o tyto výstupy:

Graf počtu agentů jednotlivých kast v čase Může sloužit k ověření vypočtené hodnoty předpokládaného počtu agentů a také ukazuje jak se kterým agentům daří v prostředí přežívat.

Graf celkové energie jednotlivých kast Pro každou kastu zobrazuje součet energie všech jejích žijících příslušníků. Může podat uživateli informace o celkových energetických příjmech a výdajích v prostředí a také o tom, jak rychle jednotlivé kasty získávají nebo ztrácejí energii.

Graf průměrné energie agentů jednotlivých kast Tento graf je užitečný zejména pro určení skutečného množství energie, které predátor získá od kořisti. V definici kasty je uvedená maximální hodnota, kterou agent při konzumaci potravy může získat, avšak konkrétní hodnota není větší než množství energie kořisti. Právě tuto hodnotu můžeme z grafu vyčíst.
Déle je možné z grafu zjistit při jaké hodnotě energie se agenti rozmnožují.

Histogram preferovaných akcí Podává přesný obraz o tom, kolik kterých akcí agenti vykonávají. Tuto informaci lze použít pro přesnější odhad hodnoty $ztraty_n$. Ve výše uvedeném příkladě jsme odhadli, jaké akce bude agent používat. S informací navíc můžeme průměrnou ztrátu za jeden krok vyjádřit jako energyLostPerTick + vážený průměr cen akcí, kde váhy zjistíme z histogramu.

Z důvodu poměrně velké časové náročnosti je pro experimentální určování parametrů vhodné spouštět pomocné simulace na prostředí s menšími rozměry a teprve po vhodném nastavení parametrů spustit experiment větší velikosti. Je však třeba brát v úvahu, že sada pravidel pro rozhodování agentů se na začátku generuje náhodně a tedy při malém počtu agentů se může stát, že žádný z nich nedostane dostatečně dobrou počáteční sadu pravidel a například se ani nedokážou rozmnožit a hned v první generaci vyhynou.

Kapitola 5

Závěr

5.1 Dosažené cíle

V této práci se podařilo navrhnout použitelný nástroj pro simulovaní života umělých bytostí s podporou kast. Systém je dostatečně flexibilní, aby umožňoval uživateli navrhnout vlastní kasty agentů a popsat jejich vzájemné interakce.

Konkrétní zaměření programu, tedy to, pro které scénáře je vhodný a pro které nikoli, je ovlivněno rozhodnutími učiněnými při návrhu. Zejména se jedná o návrh senzorů.

V našem modelu mohou agenti pozorovat pouze sousední políčka a také nejsou schopni rozlišovat mezi různými přislušníky stejné kasty. Následkem toho není možné simulovat některé sofistikovanější chování vyšších živočichů jako například teritoriální chování, lovení ve smečkách nebo péči o potomky.

Také prostředí, ve kterém se agenti pohybují, je do značné míry zjednodušené. Nejsou zde žádné překážky, všichni agenti mají stejnou velikost a podobně. Není proto možné skutečně realisticky simulovat život vyšších živočichů a program k tomu ani není určen.

Naopak experimenty s jednoduššími organismy, jako jsou bakterie nebo hmyz, mohou poměrně dobře odpovídat realitě.

5.2 Další možná rozšíření

Rozšiřování programu se může ubírat několika směry. Je možné do modelu přidat další akce, které agenti budou moci vykonávat. Příkladem smysluplné akce, která není v programu použitá, může být "zašlapání trávy", kdy agent zničí potravu aniž by ji pozřel (například aby zabránil šíření trávy, která je pro něj jedovatá).

Při přidávání akcí je třeba dát pozor na to, zda efekt nově přidané akce bude možné vyjádřit pomocí protokolu používaného k zaznamenání průběhu simulace do logovacího souboru. Například efekty akce "zašlapání trávy" by bylo možné takto vyjádřit. Pokud by však efekt nové akce byl složitější, bylo by třeba formát ukládání upravit, což by mohlo způsobit zpětnou nekompatibilitu s dříve uloženými simulacemi.

Jiný způsob rozšiřování programu může spočívat v přidání dalších parametrů v definici kast. Takovým parametrem by mohla být například "délka spánku" u agentů, kterou by bylo možné vcelku snadno zavést, nebo "rychlost pohybu", což už by ovšem znamenalo větší zásah do fungování systému.

Dále je možné uvažovat i o úpravách, které by učinily program uživatelsky přívětivější, například editor definičních souborů kast nebo tzv. startovacích souborů, které popisují parametry simulace (které kasty budou použité, jejich počáteční počet, délka trvání simulace, podmínky ukončení a podobně).

Literatura

[1] Dawkins Richard: *Sobecký gen*, Mladá fronta, Praha, 1998

[2] Hargreaves Heap Shaun, Varoufakis Yanis: *Game theory: a critical text*, Routledge, 2004. 191–211.

[3] Holan Tomáš: *Rodičovské investice umělých bytostí*, in MIS 2008, Matfyzpress, Praha, 2008. 11–18.

[4] Koukolík František: *Proč se Dostojevskij mýlil?*, Galén, Praha, 2007.

[5] Stanley Kenneth O., Miikkulainen Risto: *Evolving Neural Networks through Augmenting Topologies*, in Evolutionary Computation Volume 10 Number 2, The MIT Press, 2002. 99–127.

[6] Tyrrell Toby: *Computational Mechanisms for Action Selection*, Ph.D. Dissertation. Centre for Cognitive Science, University of Edinburgh, 1993

Příloha A

Obsah přiloženého DVD

- Program.zip - archiv obsahuje všechny zdrojové kódy projektu, lze otevřít přímo jako projekt ve Visual Studiu. Je zde obsažen také spustitelný soubor.
- Bakalářská práce.pdf - text této práce ve formátu pdf.
- Složka Dokumentace - obsahuje uživatelskou a programátorskou dokumentaci
  - Dokumentace\Uživatelská dokumentace.html - obsahuje uživatelskou dokumentaci včetně návodu ke spuštění programu
  - Dokumentace\VSdoc\Bakalarka.chm - programátorská dokumentace - přehled použitých tříd a metod včetně popisu jejich funkce.
- Složka Experimenty - obsahuje výsledky experimentů popsaných v této práci, soubory s definicemi kast a startovací soubory, které umožňují znovu spustit simulace se stejnými počátečními podmínkami. Výsledky experimentů je možné pomocí programu znovu prohlížet.