Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Optimalizace agentových modelů pomocí evolučních algoritmů

Diplomová práce

Autor: Bc. Roman Zajíček
Studijní obor: Aplikovaná informatika
Vedoucí práce: Doc. RNDr. Kamila Štekerová, Ph.D.

Hradec Králové listopad 2015

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

podpis

V Hradci Králové dne 12. 11. 2015 ……

Anotace

Diplomová práce se zabývá základním teoretickým popisem agentových modelů a genetických algoritmů. V teoretické části je dále nastíněn chod primitivního genetického algoritmu s ruletovou selekcí. Následuje základní popis NetLoga a navazujícího nástroje BehaviorSearch.

V praktické části nejprve probíhá výběr agentových modelů pro optimalizaci. Vybranými modely jsou: Muscle Development, Wolf Sheep Predation a Altruism. Ke každému vybranému modelu je navržena hypotéza a jsou provedeny experimenty v nástroji BehaviorSearch. Závěrem se hodnotí výsledky, z hlediska porovnání všech algoritmů nástroje BehaviorSearch.

Annotation

Title: Optimization of agent-based models using evolutionary algorithms

Diploma thesis deals with the basic theoretical description of the agent based models and genetic algorithms. The theoretical part also outlines the operation of primitive genetic algorithm with roulette selection. Follows the basic description of NetLogo and follow-up tool BehaviorSearch.

In the practical part it takes place the agent based models choice for optimization. Select models are: Muscle Development, Wolf Sheep Predation and Altruism. Hypothesis prior for testing is made for each selected model. Then, the experiments performed in the BehaviorSearch tool. Finally, the results are evaluated in terms of comparing the algorithms in the BehaviorSearch tool.

Poděkování

Tímto bych chtěl poděkovat paní docentce RNDr. Kamila Štekerová, Ph.D. za cenné rady a odborné vedení během mé diplomové práce.

OBSAH

1. ÚVOD

Agentové modelování je silná simulační technika. V modelech mnoho agentů interaguje podle jednoduchých pravidel a vznikají tak komplexní systémy napodobující ty reálné. Tato technika začíná být populární v mnoha vědních oborech [1] a to díky síle namodelovat mnoho různých přírodních nebo umělých procesů. Důležitým krokem v modelování těchto procesů je analýza, jak se chovají systémy ovlivněné různými parametry. Ovšem počet a rozsah hodnot těchto ovládajících parametrů je někdy příliš vysoký, výpočetní náročnost může být značná a agentové modely zahrnují stochastické elementy. Z této skutečnosti vyplývá, že musí být vykonáno velké množství pokusů pro vyhodnocení chování modelu. Prohledávací prostor je pak obrovský. Malé změny v jednom parametru mohou vyvolat radikální změnu v chování systému, oproti tomu vznik nějakého jevu se objeví jenom ve velmi specifických podmínkách. Potom oblast řešení problému může být velmi malá. V důsledku toho vývoj a nastavení parametrů agentového modelu může trvat dlouho, pokud nemáme přesnou, automatickou a systematickou strategii k nalezení správného nastavení parametrů.

Možným přístupem k řešení tohoto problému je vývoj a validace agentových modelů jako optimalizační problém. Tuto širokou škálu problémů dokážou řešit optimalizační techniky, jako například evoluční algoritmy, které můžou být využity pro zkoumání parametrového prostoru a nalézt nejlepší kombinaci parametrů vzhledem k optimalizační funkci.

Použití genetických algoritmů v tomto kontextu nemusí být vždy snadné. Prvním problém spočívá ve volbě fitness funkce. Agentové modely jsou dynamické a stochastické, což komplikuje měření fitness funkce. Co a kdy má být měřeno, jsou otázky, které silně ovlivní chod modelu, a tím i kvalitu výsledků.

S ohledem na výše zmíněné skutečnosti bude představen poměrně nový nástroj BehaviorSearch, který byl stvořen pro účely vyhledávání nebo objevování parametrového prostoru. Použitím BehaviorSearch se využívá genetických algoritmů a jiných heuristických algoritmů, pro optimalizaci modelů z programu NetLogo. Rovněž bude nabídnuta optimalizace třech různých vybraných modelů NetLoga i porovnání heuristických algoritmů, které nabízí nástroj BehaviorSearch.

2. TEORETICKÁ ČÁST

2.1. AGENTOVÉ MODELY

Pohled na agentové modelování je převzat podle Macala a Northa [2]. Agentově orientované modelování je relativně nový přístup k vytváření modelů komplexních systémů složených z autonomních účastníků (jedinců, entit, agentů). Agenti mají své chování často zadané jednoduchými pravidly, vstupují do vzájemných interakcí a navzájem se ovlivňují. Rozdílnosti mezi chováním a atributy jedinců může mít za následek kvantitativní projevy systému jako celku. Různé vzory, chování a struktury vznikají právě z interakcí mezi agenty, aniž by byly naprogramovány. Agentové modelování nabízí možnosti pro modelování sociálních systémů, na které pohlížíme jako na seskupení ovlivňujících se mezi sebou, učících a adaptujících se.

Využitelnost agentových modelů je široká, lze namodelovat chování kapitálových trhů, šíření epidemií, chování zákazníků nebo pomocí modelů odkrývat důvody zániku starodávných civilizací. Některé modely jsou malé, ale velmi elegantní, zahrnují jen základní detaily systému a poskytují tak náhled do sociálního procesu a chování. Další agentové modely většího rozměru, ve kterých je zohledněno mnoho detailních informací, jsou přesně validovány a jejich výsledky jsou pak použity na důležitá rozhodnutí nebo informování například policie. Všechny modely bylo možné vytvořit v důsledku vývoje v oblasti agentového modelování, nové přístupy ve vývoji agentových modelů, zvýšením jejich autonomnosti a pokroku ve výkonech počítačů.

2.1.1. AGENTOVÉ MODELOVÁNÍ A JEHO KOMPLEXNOST

Komplexnost agentového modelování je inspirována podle [3]. Agentový model může reprezentovat komplexní systém, případně adaptivní systém. Komplexně adaptivní systémy dávají možnost agentům se adaptovat na individuální nebo populační úrovni. Takovéto komplexní systémy pomáhají zkoumat univerzální principy pro zakládání samo-organizace a původy adaptací v přírodě. Agentové modelování začalo převážně jako soubor myšlenek, technik a

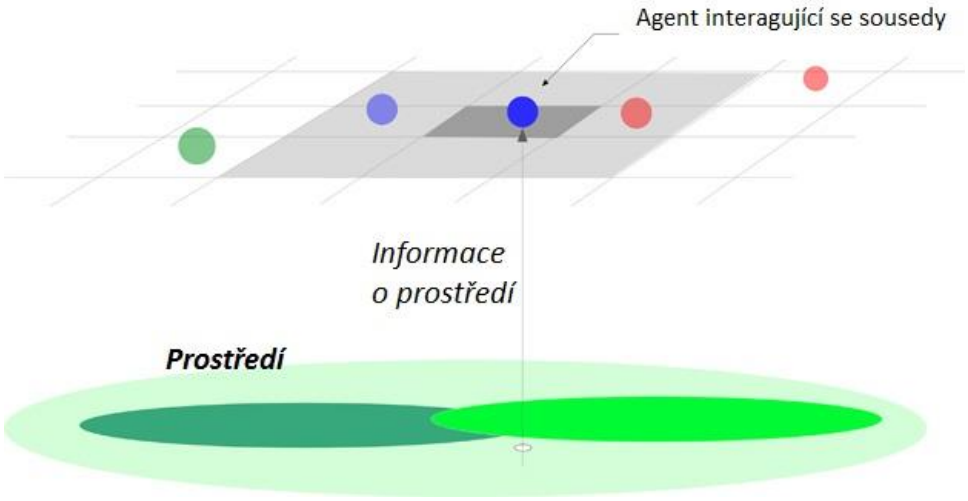
nástrojů pro vývoj výpočetních modelů komplexně adaptivních systémů. Chování agentů bylo dříve modelováno pomocí velmi jednoduchých pravidel, z kterých vznikali čím dál složitější a komplexní modely chování.

2.1.2. STRUKUTURA AGENTOVÝCH MODELŮ

Klasický agentový model se skládá ze dvou hlavních elementů, jimiž jsou:

- Agenti, jejich atributy a chování.
- Prostředí obklopující agenty.

Typická struktura agentového modelu je uvedena na obrázku 2.1. Všechny komponenty na obrázku 2.1 budou popsány v této podkapitole. Pro simulaci chování agentů a jejich interakce je potřeba programovacích jazyků nebo různých nástrojů pro modelování agentů. Agentový model se rozběhne poté, co se spustí opakovaní procesů chování a interakcí agentů. [4]



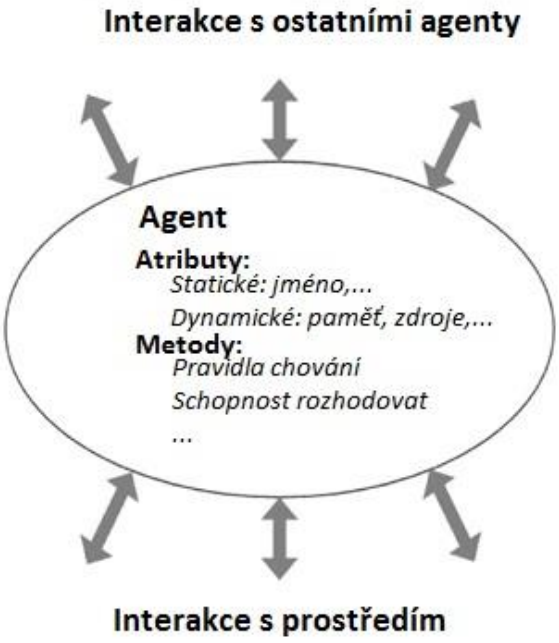
Obrázek 2.1: Struktura agentového modelu přeložena z [4]

2.1.3. AUTONOMNOST AGENTŮ

Velmi důležitou součástí agenta je, že má schopnost jednat a rozhodovat se autonomně, to znamená, že jeho rozhodování v situacích, které nastanou, není závislé na vnějším nasměrování. Agenti jsou vybaveni schopností dělat nezávislá rozhodnutí, jsou aktivní a jejich akce vedou k dosáhnutí interního cíle. Pro splnění cílů spolupracují společně s dalšími agenty a prostředím.

V literatuře nelze nalézt univerzální definici pro agenta, shodují se pouze ve vlastnostech autonomnosti. Přesto z různých implementací modelů lze vypozorovat pár základních charakteristik, jako nabízí například[5].

- Agent je **soběstačný** a jednoznačně identifikovaný jedinec. Agent má jasně vymezené hranice, lze snadno odhalit co je součástí agenta, co není a co je společnou vlastností. Díky těmto vlastnostem je pak možné ho bez problému rozeznat od dalších agentů.
- Agent je **autonomní** a samořízený, může jednat nezávisle na jeho prostředí a na ostatních agentech. Agent má samostatné chování podle jeho rozhodnutí a akcí. Toto chování pochází z interakcí s ostatními agenty a prostředím.
- Agent má **stav**, který se mění v průběhu času. Stejně jako systém má stav sestávající se z jeho stavových proměnných, tak také agent má stav, který representuje základní proměnné podle jeho aktuální situace. Agentův stav závisí na souboru jeho vlastností. Stav agentového modelu je souhrnem stavů všech agentů. Čím více stavů může agent nabývat, tím větší možnosti jeho chování může mít.
- Agent je **sociální,** má dynamické interakce s dalším agenty a tím ovlivňuje jejich chování. Má souhrn společenských pravidel pro interakci s ostatními agenty jako je komunikace, pohyb a sdílení prostoru, možnost odezvy vůči prostředí atd. Agenti mají schopnost rozpoznat a rozlišit zvláštnosti dalších agentů.
- Agenti můžou být **adaptivní,** mít schopnost naučit se a adaptovat jejich chování založené na předchozích zkušenostech. Toto učení vyžaduje mít nějaký způsob paměti.
- Agent může být **cílený**, schopnost dosahovat určitých cílů s důrazem na jeho chování. Toto umožňuje agentům porovnat výsledek chování vůči stanoveným cílům a přizpůsobit tomu budoucí interakce.



Obrázek 2.2: Struktura typického agenta inspirována podle [6]

Na obrázku 2.2 je vidět struktura typického agenta. Následující postoj k problematice je možné vysledovat podle [6]. V agentových modelech všechno, co je asociováno s agenty, je buď atributem agenta nebo metodou agenta. Atribut agenta může být statický, to znamená neměnný v průběhu simulace, nebo dynamický, to znamená, že se v průběhu simulace mění. Jak je uvedeno v obrázku 2.2, statickým atributem může být například jméno, kdežto dynamickým atributem například paměť z minulých interakcí.

2.1.4. INTERAKCE AGENTŮ

Podle [7] dva základní problémy v agentovém modelování jsou: kdo je anebo by mohl být spojen s kým, a způsob jejich interakce. Oba tyto aspekty musí být zohledněny při vývoji agentového modelu.

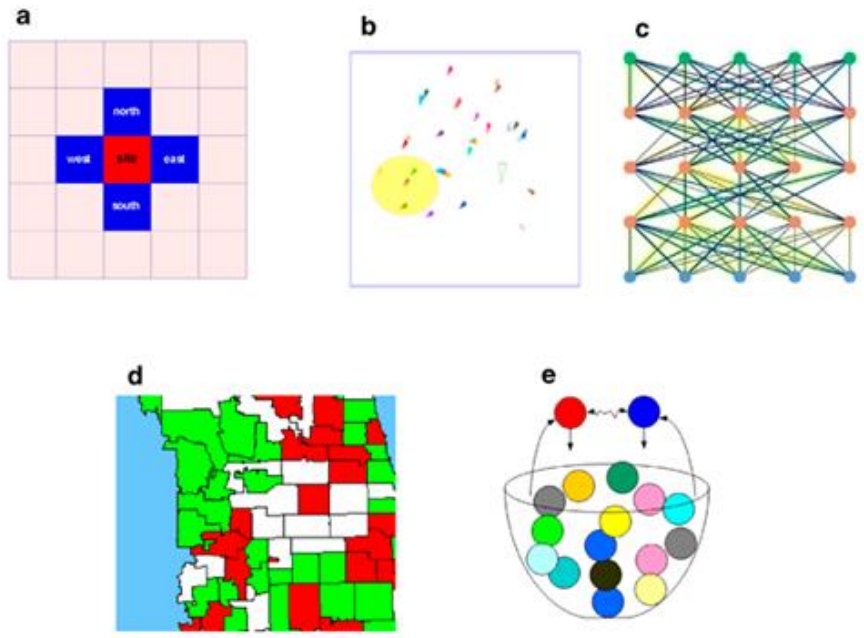
Jedním z principů komplexního systému a agentového modelování je, že agent má k dispozici pouze lokální informace. Agentové systémy jsou decentralizované. Není zde žádný centrální prvek, který by agentům dodával globálně dostupné informace, ani který by kontroloval jejich chování ve snaze optimalizovat systém. Agenti interagují s ostatními agenty, ale všichni agenti neinteragují přímo se všemi agenty celý čas. Agenti typicky interagují s podmnožinou jiných agentů, v této terminologii se jim říká sousedé. Lokální

informace bývají získávány právě v interakci se sousedy a z jejich lokálního prostředí.

Způsob propojení agentů se nazývá topologie modelu. Typickou topologií je mřížka nebo síť uzlů (agentů) a spojů (vztahů). Topologie popisuje kdo a s kým si přenáší informace. V některých systémech agenti interagují podle více topologií. Agenti interagují se svými sousedy, kteří se fyzicky nebo geograficky nacházejí blízko. Interagují také se sousedy nacházející se ve stejném sociálním prostoru specifikovaném v agentově sociální síti.

Původ v agentovém modelování se nalézá v celulárních automatech. Celulární automat je dvoudimenzionální mřížka definovaná buňkami. Buňky v okolí agenta jsou jeho sousedství. Každá buňka je identifikována jako agent a interaguje s předdefinovaným počtem sousedících buněk. Buňka je ve stavu buď „zapnuto“ nebo „vypnuto“. Většina prvních agentových modelů byla vytvořená právě touto formou.

Nyní jsou běžně používané jiné topologie agentových interakcí, jako je na obrázku 2.3. V celulárních automatech se agenti pohybují mezi buňkami na mřížce, zároveň pouze jeden agent může být v jedné chvíli v jedné buňce.



Obrázek 2.3: Topologie pro vztahy mezi agenty upravena podle [8]

Von Neumannův model pěti sousedů, který je uveden na obrázku č. 2.3a. Je běžný i model devíti sousedů. V modelu Euklidovského prostoru, agenti kolují v dvou, tří nebo vícedimenzionálním prostoru (obrázek 2.3b). Síťová topologie může být statická nebo dynamická (obrázek 2.3c). Ve statické síti jsou spoje předdefinovány a nemění se, kdežto v dynamické síti, spoje a někdy i uzly jsou děděny podle naprogramovaného modelu. V geograficky informativním systému (GIS) se agenti pohybují po realistických geografických prostorech (obrázek 2.3d). V „polévkovém“ modelu nemají agenti žádnou polohu, protože to zde není důležité (obrázek 2.3e). Páry agentů jsou náhodně vybrány pro interakci a pak vráceny do „polévky“ jako kandidáti na budoucí výběr. Mnoho agentových modelů využívá více topologií pro interakci jejich agentů. [8]

2.1.5. AGENTOVÉ PROSTŘEDÍ

Agenti interagují s jejich prostředím a s ostatními agenty, tak jak popisuje [9]. Prostředí může být využito pro poskytnutí informací o agentově pozici ve vztahu k ostatním agentům nebo může poskytnout geografickou informaci jako v GIS. Agentova lokace je brána jako dynamický atribut, sleduje pohyby napříč modelem, souboje o prostor, získávání zdrojů a dalších situací. Například hydrologické a atmosférické modely mohou poskytnout náhled na to, kde se nachází podzemní voda respektive látky znečišťující ovzduší. Prostředí může tedy vynutit akce agenta.

2.1.6. PŘÍKLADY AGENTOVÝCH MODELŮ

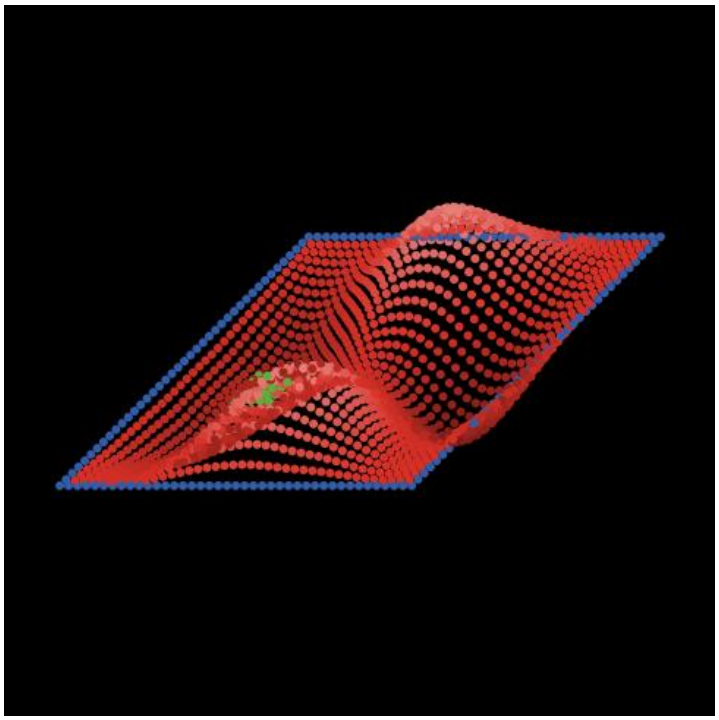
Jak uvádí zdroj [10]. Agentové modelování je využíváno v několika vědních oborech, například fyzikálních, biologických, sociologických a v oborech managementu. Níže bude uvedeno pár příkladů těchto aplikací. Existuje celá škála modelů, od minimalistických, založených na zachycení pouze základních charakteristických rysů systému až po systémy, které pomáhají při důležitých rozhodnutích v reálném světě obsahující reálná data.

Z několika vědních oborů bude uveden jeden příklad agentového modelu:

- **FYZIKÁLNÍ**

V tomto oboru jsou modely reprezentující reálné fyzikální jevy. Vznikají modely na krystalizace kovů, postup teploty při vaření vody nebo vznikání vln

apod. Jako příklad v tomto oboru byl vybrán model pohyb vln „Wave Machine“. (Obrázek 2.4)

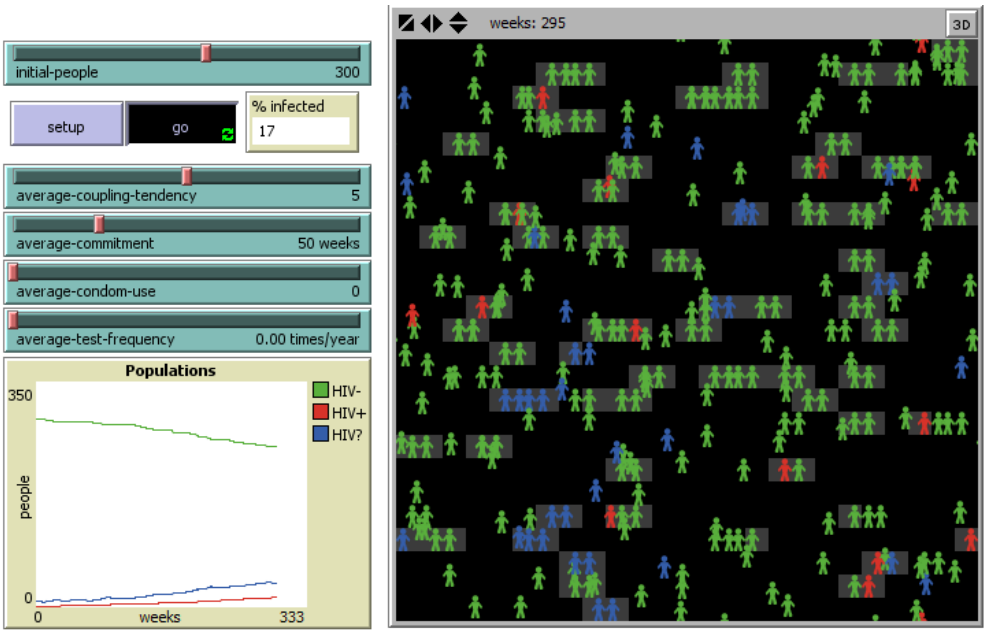


Obrázek 2.4: Screenshot modelu „Wave Machine“ z NetLoga [11]

Model simuluje pohyb vln na pružné tenké desce. Modré hrany jsou přichyceny k rámu. Zelený čtverec představuje ovladač desky, který se pohybuje nahoru a dolů a způsobuje sinusoidní pohyb. [11]

- **BIOLOGICKÝ**

V biologické vědě je agentové modelování použito při modelech chování buněk a interakcí, fungování imunitního systému, růst tkání, průběh nemocí a další. Autoři těchto modelů tvrdí, že agentové elektronické modelování poskytuje přídavek ke klasickým laboratorním výsledkům. Jedním z těchto modelů je například model „AIDS“. (Obrázek 2.5)

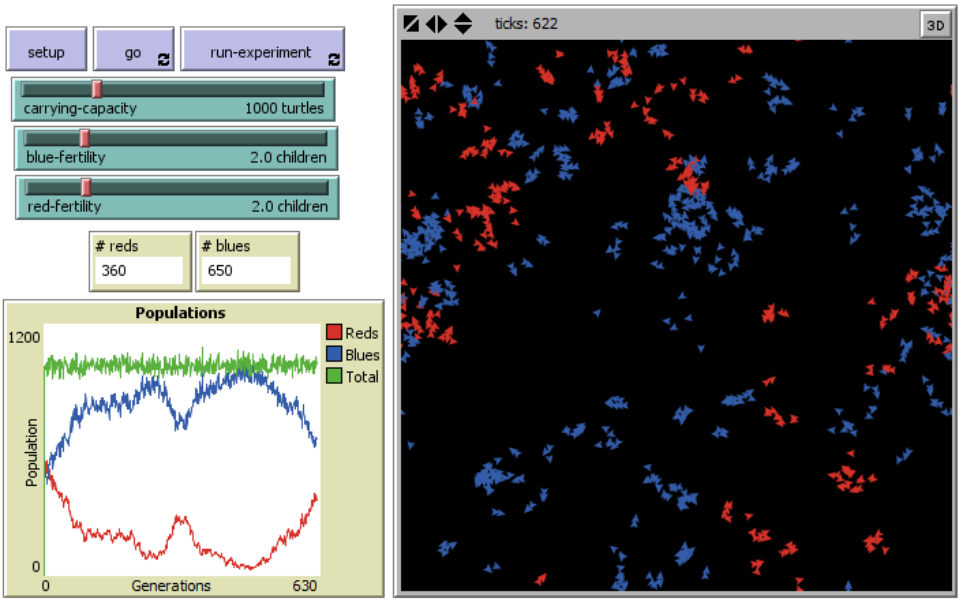


Obrázek 2.5: Screenshot modelu „AIDS“ z NetLoga [12]

Model popisuje šíření viru lidské imunitní nedostatečnosti (HIV) pohlavním přenosem skrz malou isolovanou lidskou populaci. Model odhaluje dopady sexuálního chování v závislosti na několika proměnných. [12]

- **SOCIOLOGICKÝ**

Výpočetní sociální věda je rozvíjející se obor, který spojuje modelování a simulace se sociálními vědními obory. Několik sociálních jevů je zkoumáno použitím agentových modelů. Jedním z těchto modelů je „Simple Birth Rates“ na obrázku 2.6. Model jednoduše simuluje generace populace. V modelu jsou dvě populace, jedna je červená a druhá je modrá, obě dvě mají nastavitelnou porodnost. Když nějaký jedinec překročí „hranice“ modelu, nějaký agent umírá (při stejné pravděpodobnosti u všech agentů) pro udržování relativně konstantní populace. Model představuje, jak velký rozdíl může způsobit porodnost populace. [13]



Obrázek 2.6: Screenshot modelu „Simple Birth Rates“ z NetLoga [13]

Agentové modelování lze využít v mnoha dalších odvětvích jako například v umění, chemii, počítačových vědách, hrách, matematice, sítích a podobně. Je nepochybné, že agentové modelování má své místo v oblasti výpočetní techniky.

2.2. GENETICKÉ ALGORITMY

2.2.1. ZÁKLADNÍ POJMY

Genetický algoritmus pracuje s několika pojmy z oblasti genetiky, pro popis datových a programových struktur algoritmu. Podle [14] jsou to tyto pojmy:

- **Populace (P) –** Množina jedinců v jednom okamžiku. Jedná se o soubor hypotéz, které jsou v genetickém algoritmu k dispozici. V každé iteraci genetického algoritmu jsou vybrány nejvhodnější řešení. Genetický algoritmus pracuje buď s jednou, nebo i více populacemi současně, které se pak rozvíjejí do různých struktur, jako jsou sítě a stromy. Mezi strukturami pak dochází k výměně informací o vhodnosti řešení problému, tím nemusí hned konvergovat k lokálnímu optimu.
- **Jedinec –** Jedno možné řešení problému. Tělo jedince se skládá z genotypu a fenotypu. Fenotyp je řešení problému a genotyp je potřebný pro vytvoření fenotypu. Genotyp obsahuje veškerou genetickou informaci. V genetických algoritmech se využívá pouze jednoho chromozomu.
- **Fitness (F) –** Je ohodnocení jedince, tedy každého řešení, které je algoritmem vytvořeno. Je přiřazeno na základě kriteriální funkce, kterou je nutné znát, protože určuje, které řešení je lepší a horší. Vyjadřuje vzdálenost od optimálního řešení. Jedinec s vyšším hodnocením má větší šanci „přežít“ a vytvářet tak další generace.
- **Gen –** Jednotka genetické výbavy jedince. Význam závisí na řešeném problému a na použitém kódování. Genem může být jeden bit, znak, celé číslo apod.
    - **Chromozom –** Vektor genů. Část genetické výbavy jedince, přenášená na jeho potomky.
        - **Alela –** Konkrétní hodnota na jednotlivých pozicích v genu.
    - **Genotyp –** Kompletní genetická výbava jedince. Soubor chromozomů se nazývá genom. V genetických algoritmech je genetická výbava obsažena jen v chromozomech. Většina

genetických algoritmů pak využívá jen jeden chromozom. Tedy v genetických algoritmech genotyp = genom = chromozom.

- **Fenotyp –** Množina vedlejších vlastností a znaků jedince. V genetických algoritmech je právě to řešení problému. Fenotyp může mít jakoukoliv podobu odpovídající řešené úloze.

2.2.2. DEFINICE GENETICKÉHO ALGORITMU

Genetický algoritmus je heuristická metoda založená na principech evoluční biologie. Pro efektivní nasazení genetického algoritmu je vyžadováno maximálních znalostí o struktuře, charakteru problému a případném využití tradičních metod a algoritmů, čemuž se říká „hybridizace“.

John Holland, stojící u základů v oblasti genetických algoritmů, studoval generační procesy v populacích. Výsledkem jeho bádání byl návrh genetického algoritmu, který tyto biologické procesy napodoboval. [15]

Soupeření v genetických algoritmech podle [16].

„*V přírodě obvykle individua v rámci dané populace soupeří o zdroje potřebné k přežití, chrání se před predátory a usilují o možnost vlastní reprodukce. Nejúspěšnější jedinci, kteří obstojí v tomto náročném zápasu, mají potom příležitost žít déle, nalézt a přilákat vhodného partnera a následně společně zanechat relativně větší počet potomků než méně úspěšní jedinci, kteří mají omezenější možnosti se reprodukce účastnit, a počet jejich potomků je potom menší, nebo dokonce zemřou bez potomstva. Použitím tohoto principu je možné dosáhnout toho, že vlastnosti schopnějších či lépe přizpůsobivých jedinců zakódované v relevantní genové výbavě se předávají rostoucímu počtu potomků v dalších generacích. Navíc je zřejmé, že vhodnou kombinací vlastností rodičovského páru lze docílit toho, že jejich potomek (nebo potomci) bude mít lepší vlastnosti než kterýkoliv z obou rodičů a bude tedy lépe přizpůsoben životu v daném prostředí.*„

Hollandův genetický algoritmus je založen na právě zmíněných Darwinových objevů v přírodě. Genetický algoritmus pracuje s populací jedinců a každý tento jedinec má v sobě zabudováno nějaké řešení problému. Všem jedincům je přiřazené ohodnocení. Působením genetických operátorů mutace, křížení a inverze mezi vybranými jedinci v procesu přirozeného výběru vzniká

možnost reprodukce. Opakováním procesu reprodukce vznikají nové populace s čím dál zdatnějšími jedinci, tedy přijatelnému nebo dokonce optimálnímu řešení problému. [17]

Genetický algoritmus je algoritmem slepým, nemá žádné předchozí znalosti o konkrétním problému nebo prohledávacím prostoru. Tím je velmi univerzální, ale nemůže konkurovat speciálně navrženým algoritmům pro daný problém. Obecně se dá říci, že genetický algoritmus se hlavně využívá pro řešení problému, kde není žádný předem známý speciální algoritmus. Nebo se využívá právě v kombinaci se speciálními algoritmy, tedy jako „hybridní algoritmy".

Definovat algoritmus lze také pomocí obecného schéma jak uvádí [16]:

1. Vynuluj hodnotu počítadla generací *t = 0*.
2. Náhodně vygeneruj počáteční populaci *P(0).*
3. Vypočítej ohodnocení (fitness) každého jedince v počáteční populaci *P(0).*
4. Vyber dvojice jedinců z populace *P(t)* a vytvoř jejich potomky *P'(t).*
5. Ohodnoť nově vytvořená jedince v *P'(t).*
6. Vytvoř novou populaci *P(t+1)* z původní populace *P(t)* a množiny potomků *P'(t).*
7. Zvětši hodnotu počítadla generací o jedničku *(t:= t + 1).*
8. Vypočítej ohodnocení (fitness) každého jedince v populaci *P(t).*
9. Pokud *t* je rovno maximálnímu počtu generací nebo je splněno jiné ukončovací kritérium, vrať jako výsledek populaci *P(t);* jinak pokračuj krokem číslo 4.

2.2.3. PRINCIP GENETICKÉHO ALGORITMU

Principem genetických algoritmů je napodobování evolučního procesu. Začne se s vytvořením nové populace. Tento proces probíhá většinou náhodně, ale pokud je k dispozici nějaká heuristika, může se jí využít v tomto kroku. Každý jedinec v populaci je vyjádřen svým chromozomem s pevnou délkou a pevně uspořádanými geny. Je velmi důležité pro úspěch nebo neúspěch genetického algoritmu mít správně reprezentovány (zakódovány) jedince.

Jako dostatečné se obecně jeví kódování ve formě řetězce či pole hodnot. Kódování má svůj obraz také v genetice, kdy řetězce reprezentují chromozómy, pozice v těchto řetězcích jednotlivým genům a hodnoty na těchto pozicích alely. V praxi se velmi často využívá binárního kódování. Toto kódování bude také využito pro účely vysvětlení jednotlivých kroků genetického algoritmu.

Binární kódování představuje gen, který nabývá pouze dvou stavů a to: 0 nebo 1. V příkladu bude vygenerována náhodná populace čtyř jedinců s délkou chromozomu 8. Pro vygenerování populace se využije funkce EXCELu „=RANDBETWEEN(0;1)“ pro 8 sloupců a 4 řádky (tabulka 2.1). Ukázka fungování genetického algoritmu je inspirována podle [16].

Jedinec	Chromozom							
č. 1	1	0	0	0	0	1	0	0
č. 2	1	1	1	1	1	0	1	0
č. 3	0	0	1	1	1	0	1	1
č. 4	1	1	0	1	0	0	0	0

Tabulka 2.1: Vygenerovaná populace z EXCELu

Následujícím krokem algoritmu je ohodnocení každého jedince v populaci. Každá ohodnocující funkce je dána konkrétním problémem. Pro jednoduché vysvětlení algoritmu bude používáno ohodnocení podle součtu jednotkových bitů. Tedy za zdatnější jedince nebo za optimálnější řešení je považován genotyp obsahující více jedniček. Ohodnocení populace je označeno v tabulce 2.2.

Jedinec	Chromozom								Fitness
č. 1	1	0	0	0	0	1	0	0	**2**
č. 2	1	1	1	1	1	0	1	0	**6**
č. 3	0	0	1	1	1	0	1	1	**5**
č. 4	1	1	0	1	0	0	0	0	**3**

Tabulka 2.2: Ohodnocená populace

Jedním z klíčových prvků genetického algoritmu je výběr dvojice jedinců z této populace pro další reprodukci. Výběr by měl být založen na přirozeném

výběru v Darwinově teorii o původu druhů. Zdatnější jedinci mají větší šanci „přežít“ a vytvořit tak novou generaci potomků.

Je několik forem implementace přirozeného výběru. Mezi nejpoužívanější patří ruletová a turnajová selekce.

- **RULETOVÁ SELEKCE**

Při klasické ruletě může být vybráno číslo se stejnou pravděpodobností, protože mají všechny čísla stejně velkou výseč. V případě ruletové selekce jsou s větší pravděpodobností vybráni jedinci s lepším ohodnocením, jelikož takovým jedincům je přiřazena větší výseč a tím i větší šance pro výběr.

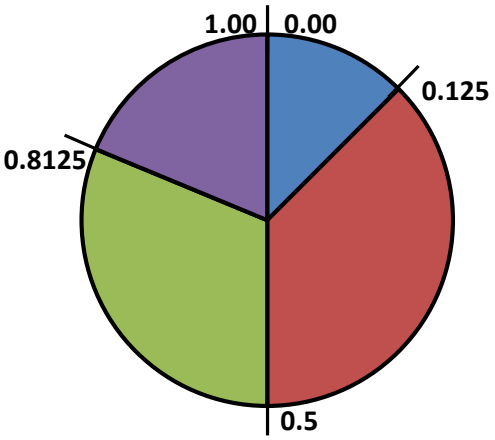
Existují různé možnosti sestrojení ruletového kola. Jednou z možností je přiřazení každé výseči relativní ohodnocení jedinců v populaci. Pro výpočet relativního ohodnocení se sečte ohodnocení všech jedinců v populaci a poté se každý jedinec tímto počtem vydělí. Relativním ohodnocením se získá pravděpodobnost (P) výběru jedince pro reprodukci. Pomocí kumulovaného ohodnocení (K) se získává podoba ruletového kola při selekci jedinců.

Jedinec	Chromozom								Fitness	P	K
č. 1	1	0	0	0	0	1	0	0	**2**	0,125	0,125
č. 2	1	1	1	1	1	0	1	0	**6**	0,375	0,5
č. 3	0	0	1	1	1	0	1	1	**5**	0,3125	0,8125
č. 4	1	1	0	1	0	0	0	0	**3**	0,1875	1
									16		

Tabulka 2.3: Populace s pravděpodobností výběru jedince

Z tabulky je zřetelně vidět, že jedinci č. 2 a č. 3 mají mnohem větší šanci být vybráni než jedinci č. 1 a č. 4. Tito jedinci však stále mohou být vybráni. I v přírodě se někdy může stát, že slabší jedinec přežije a zúčastní se reprodukce, kdežto silnější jedinec se neprosadí, nenajde vhodného protějška pro reprodukci nebo předčasně zahyne.

Ruletové kolo se sestrojí v tomto příkladu ze čtyř výsečí podle počtu jedinců. Každá výseč dostane podle kumulativního ohodnocení (K) tabulky 2.3 svojí pravděpodobnost.



Obrázek 2.7: Ruletová selekce s pravděpodobností výběru podle kumulovaného ohodnocení

Pro výběr jedinců se čtyřikrát použije ruletová selekce na obrázku 2.7. Pro udržení stejně velké populace se zvolí právě čtyři výběry. Výběr bude proveden čtyřmi pseudonáhodnými čísly mezi nulou a jedničkou. Podle toho, v jaké výseči ruletového kola skončí vygenerované číslo, se použije právě ten jedinec pro další reprodukci. Pseudonáhodně vygenerovanými čísly jsou: 0,864; 0,492; 0,243 a 0,702. Tedy vylosovaní jedinci v párech jsou v tabulce 2.4.

	Jedinec	Chromozom								Fitness
1. pár	č. 4	1	1	0	1	0	0	0	0	**3**
	č. 2	1	1	1	1	1	0	1	0	**6**
2. pár	č. 2	1	1	1	1	1	0	1	0	**6**
	č. 3	0	0	1	1	1	0	1	1	**5**

Tabulka 2.4: Rodičovské páry z ruletové selekce

Nejsilnější jedinec č. 2 byl z počáteční populace vylosován dvakrát a zúčastní se reprodukčního procesu v obou párech. Nejslabší jedinec č. 1 naproti tomu nebyl vybrán vůbec.

- **TURNAJOVÁ SELEKCE**

Tato metoda je využívána programem BehaviorSearch. Z populace se náhodně vybere *x* jedinců. Obvykle to bývá 2-3 jedinci, záleží na velikosti populace. Z těchto vybraných jedinců se pak vybere jedinec s nejlepším ohodnocením, který

dále figuruje jako rodič. Takováto selekce zaručuje, že i jedinec s menším ohodnocením má šanci stát se rodičem.

Není pravidlem, že vždy nejsilnější jedinci jsou vybráni, ale z dlouhodobého hlediska budou silnější jedinci preferováni.

Při vytváření potomků z rodičovských párů se využívají dva základní genetické operátory:

- **Křížení -** Existuje opět několik metod křížení. Všechny mají společnou charakteristiku ve výměně části chromozomů. Na tomto příkladu bude vysvětlena zřejmě nejjednodušší metoda jednobodového křížení. Náhodně se zvolí jeden gen v chromozomu a od tohoto genu se vymění zbylé části v chromozomu mezi oběma rodiči. Vznikají pak dva noví jedinci, z nichž každý má určitou část genetické výbavy po obou rodičích. V některých případech může být vhodné ponechat jedince beze změn. Z těchto důvodů se operátor křížení dělá s určitou pravděpodobností (podle [18] je to obvykle 0,6 – 0,95). V příkladu je operátor použit na oba rodičovské páry.

	Chromozom									Nový chromozom							
1. pár	***1***	***1***	***0***	***1***	***0***	***0***	***0***	***0***		***1***	***1***	***0***	***1***	***0***	0	1	0
	1	1	1	1	1	0	1	0		1	1	1	1	1	***0***	***0***	***0***
2. pár	***1***	***1***	***1***	***1***	***1***	***0***	***1***	***0***		***1***	***1***	1	1	1	0	1	1
	0	0	1	1	1	0	1	1		0	0	***1***	***1***	***1***	***0***	***1***	***0***

Tabulka 2.5: Jednobodová metoda křížení

V tabulce 2.5 je na levé straně zobrazen původní chromozom vybraných párů a na právě straně nový chromozom. V prvním páru byl náhodně zvolen gen číslo 5 a v druhém páru gen číslo 2.

- **Mutace –** Druhým genetickým operátorem je mutace. Provádí se s velmi malou pravděpodobností, obvykle od 0,001 až po 0,05. [19] Proces prochází všemi geny, a poté je podle určité pravděpodobnosti mutuje (v tomto případě změní gen z nuly na jedničku). Mutace brání algoritmus vůči předčasným konvergencím a nedostatečné divergenci.

Chromozom před mutací									Chromozom po mutaci							
1	**1**	0	1	0	0	1	0		1	**0**	0	1	0	0	1	0
1	1	1	1	1	0	0	**0**		1	1	1	1	1	0	0	**1**
1	1	1	1	1	0	1	1		1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	0		0	0	1	1	1	0	1	0

Tabulka 2.6: Mutace

Tabulka 2.6 představuje mutaci na nově vzniklé populaci po provedení operátoru křížení. Změnily se dva geny chromozomu, a tím vzniklo nové potomstvo v tabulce 2.7.

Jedinec	Chromozom								Fitness
č. 5	1	0	0	1	0	0	1	0	**3**
č. 6	1	1	1	1	1	0	0	1	**6**
č. 7	1	1	1	1	1	0	1	1	**7**
č. 8	0	0	1	1	1	0	1	0	**4**
									20

Tabulka 2.7: Nová populace

Tímto byl dokončen přechod z původní populace do populace nové. V příkladu je možné pozorovat přechod z ohodnocení populace z 16 na 20. Což znamená velmi výrazný posun vpřed vůči hledanému řešení. Původní populace už se v pokračování algoritmu nebere v úvahu a pracuje se pouze s novou populací. Tento proces se bude opakovat v cyklu do té doby, než nebude splněna ukončovací podmínka.

Ukončovací podmínkou může být například maximální počet generací, nalezení uspokojivého řešení nebo nedostatečná změna v posledních *x* generací.

2.3. NETLOGO A BEHAVIORSEARCH

2.3.1. NETLOGO

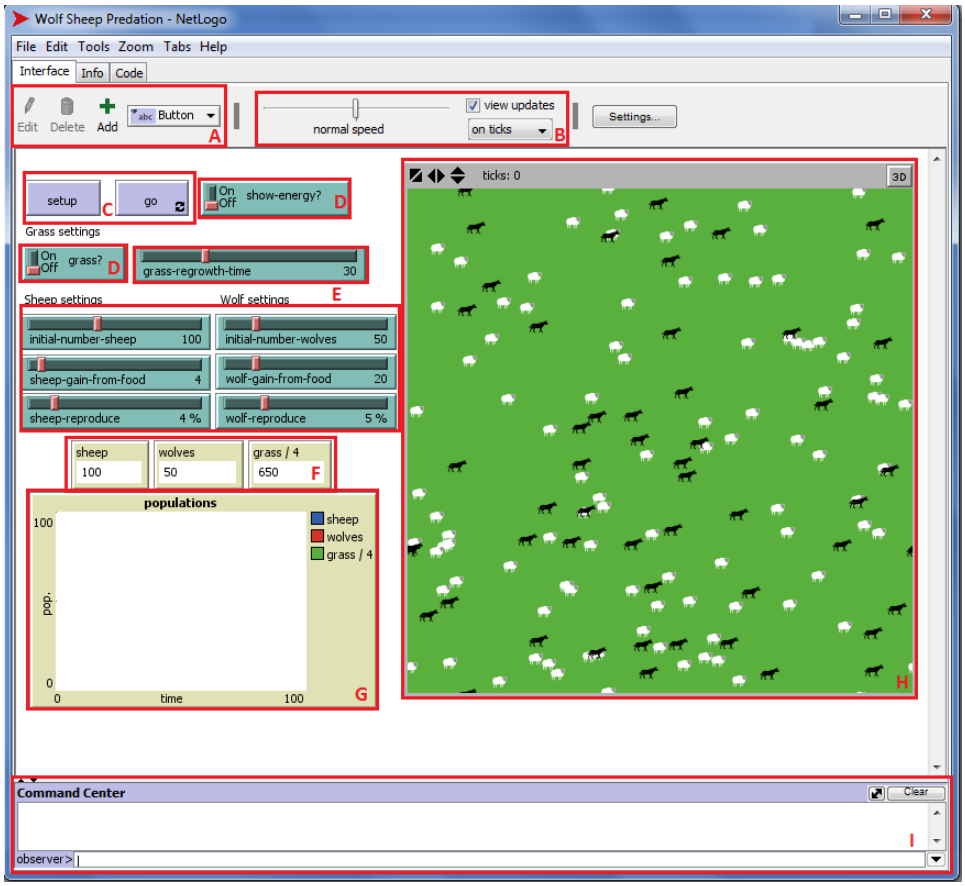
Tato podkapitola je inspirována podle [20].

NetLogo je multiplatformní programovatelné modelovací prostředí napsané v Javě, které vytvořil Uri Wilensky v roce 1999 a je určené pro simulaci přírodních a sociálních jevů. Podporuje všechny populární operační systémy (Mac OS, Windows a Linux). NetLogo je volně dostupné a je možné si ho stáhnout z jeho webových stránek. Zdrojový kód NetLoga je open source. Programuje se v něm jazykem, který vychází z programovacího jazyka Logo, odtud také název NetLogo.

Nástroj je velmi vhodný pro modelování komplexních systémů, které probíhají v čase. Vývojář modelu má možnost rozdat instrukce pro stovky až tisíce agentů, kteří se chovají autonomně. Umožňuje to objevit propojení mezi individuálním chováním a vzorem chování modelu jako celku ze vzájemných interakcí agentů. Jedná se o nejrozšířenější nástroj pro agentové modelování, jak ve vzdělání, tak i ve výzkumné oblasti.

NetLogo má velmi rozšířenou dokumentaci a mnoho tutoriálů. Má v sobě zabudovanou knihovnu modelů („Models Library“), ve které je velmi široká škála různých modelů z různých odvětví a vědních oborů včetně umění, biologie, chemie, fyziky, geografie, matematiky, sociálních věd apod. Tyto modely můžou být hned využívány nebo upravovány podle přání uživatele.

Základními programovacími kameny v NetLogu jsou agenti, zde reprezentovány jako želvy (anglicky „turtles“), například na obrázku 2.8 v podobě bílých ovcí a černých vlků. Tyto želvy jsou v prostředí mobilní a pohybují se po stacionární mřížce agentů (anglicky „patches“), například na obrázku 2.8 zelenými políčky. Agenti mohou být propojeni spoji (anglicky „link“) a vytvářejí tak sítě, grafy a další různé agregáty. NetLogo podporuje mnoho jazykových verzí.



Obrázek 2.8: Screenshot NetLogo z modelu „Wolf Sheep Predation“[21]

V prostředí NetLoga je možné pracovat s tlačítkem (**button** viz obrázek 2.8C), posuvníkem (**slider** viz obrázek 2.8E), přepínačem (**switcher** viz obrázek 2.8D), s výběrem možností (**chooser**), s textovým polem (**input, output**), monitorem viz obrázek 2.8F, grafem (**plot** viz obrázek 2.8G) a poznámkou (**note**). Těmito nástroji se ovládá prostředí běhu modelu, které je zobrazeno na plátně viz obrázek 2.8H. NetLogo také podporuje 3D modely. Pro přepnutí na 3D stačí zmáčknout tlačítko 3D na obrázku 2.8H vpravo nahoře. Další nastavení je možné dělat z příkazové řádky na obrázku 2.8I. Nastavení z příkazové řádky lze provádět ze čtyř pohledů - z pohledu pozorovatele, z pohledu želv, z pohledu polí a z pohledu spojů (link). NetLogo umožňuje spouštět model v různých rychlostech podle posuvníku na obrázku 2.8B, lze tak pozorovat detailní chování modelu apod.

2.3.2. POPIS BEHAVIORSEARCH

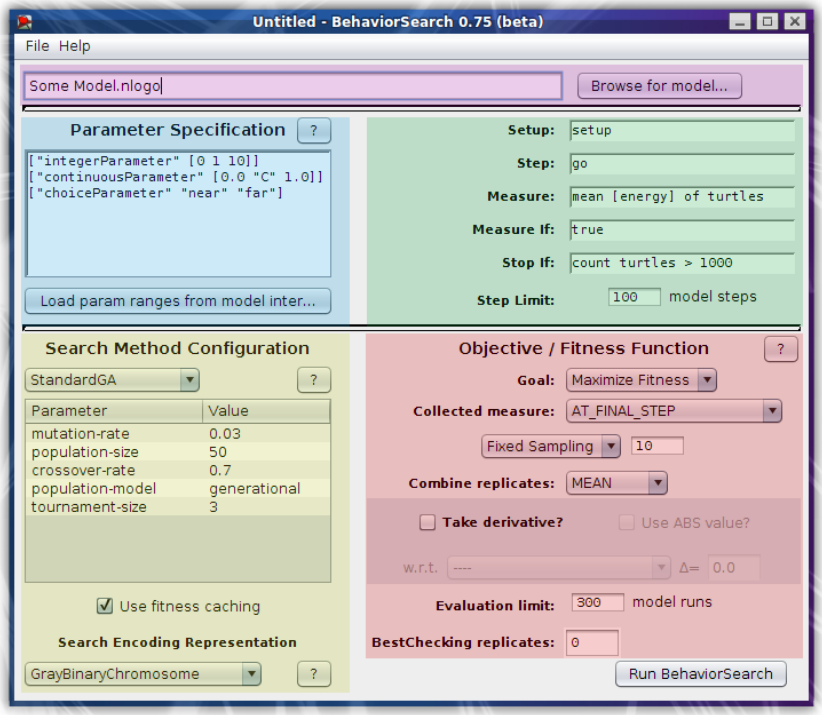
Podkapitola je inspirována podle [22].

BehaviorSearch je nástroj, který pomáhá při zkoumání agentových modelů pomocí genetických algoritmů a dalších heuristických metod. Funguje pouze ve spolupráci s NetLogem, jedná se o nadstavbu pro vyhledání ideální kombinace parametrů v modelu pro dosažení požadovaného výsledku chování.

Výzkum modelu se provádí ve čtyřech krocích:

1. Návrh měření pro požadované chování.
2. Výběr parametrů, se kterými se bude pracovat a v jakém rozsahu se mají pohybovat.
3. Výběr vyhledávacího algoritmu.
4. Zpracování výsledků.

BehaviorSearch je zaměřen na experimenty. Prostředí nástroje pomáhá vytvářet, otevírat, modifikovat a ukládat tyto experimenty (ukládány s příponou „**.bsearsh**“). Pro přehlednější rozdělení mnoha možností, které se dají v tomto nástroji nastavit, je rozčleněn do pěti logicky uspořádaných sekcí zobrazených v obrázku 2.9.



Obrázek 2.9: Úvodní obrazovka BehaviorSearch

Nástroj je rozdělen následovně podle obrázku 2.9:

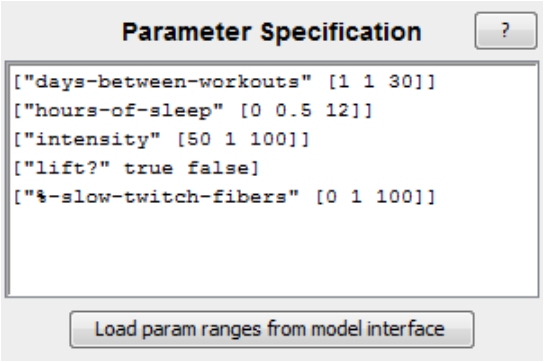
- **Fialová sekce** – Odkazuje na zpracovávaný model z NetLoga.
- **Modrá sekce** – Odkazuje na parametry z modelu a určuje jejich rozsah.
- **Zelená sekce** – Určuje jak model spustit, jakou proměnnou měřit a podmínky ukončení běhu modelu.
- **Žlutá sekce** – Určuje jaký vyhledávací algoritmus, nastavení parametrů algoritmu a kódování se použije.
- **Červená sekce** – Určuje, kolikrát se má spustit model a jakým způsobem získat data.

Vzhledem k dalšímu využití nástroje BehaviorSearch v praktické části této práce budou jednotlivé sekce popsány detailně.

2.3.3. SPECIFIKOVÁNÍ VYHLEDÁVACÍHO PROSTORU

Podkapitola inspirována podle [22].

Ve specifikování bude využíván model „**Muscle Development**“, který bude detailně popsán v kapitole 3.1. Prvním krokem spuštění genetického algoritmu na model se specifikuje vyhledávací prostor. BehaviorSearch podporuje funkci načtení parametrů z uživatelského prostředí přes tlačítko z obrázku 2.10 „**Load param ranges from model interface**“. Funkce automaticky vytáhne parametry z modelu společně s jejich rozsahem. Ve specifikaci je v uvozovkách vždy název parametru a následuje v hranatých závorkách velikost rozsahu prohledávání parametru. Číslo mezi, určuje velikost inkrementace, ta může být také znázorněna jako „C“ a znamenalo by to všechna čísla mezi daným rozsahem. Pokud má být parametr konstantní, je bez hranatých závorek, nebo může nabývat vypsaných hodnot za parametrem.



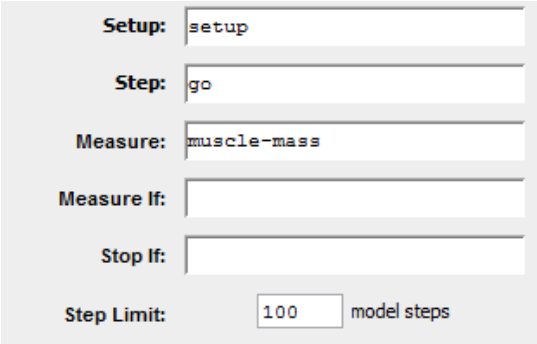
Obrázek 2.10: Příklad vyhledávacího prostoru pro model „Muscle Development“

V příkladu na obrázku 2.10 je velikost prohledávacího prostoru 30×25×50×2×101 = 7575000 různých kombinací parametrů.

2.3.4. NÁVRH MĚŘENÍ

Podkapitola inspirována podle [22].

Návrh měření spočívá v definování cílového chování nebo finálního výsledku, který má experiment přinést. V mnoha agentových modelech může být toto nadefinování složité a je základem pro smysluplný experiment. Konkrétní detaily návrhu měření modelu „Muscle Development“ budou vysvětleny v kapitole 3.2 a 3.3. V této části budou jen specifikovány jednotlivá pole formuláře z obrázku 2.11.



Obrázek 2.11: Určuje, jak má být model spuštěn, kdy a jaké data shromažďovat

- **Setup** – Příkaz v NetLogu, který nastaví běh modelu na počáteční hodnoty. Typicky to bývá procedurou „setup“.
- **Step** – Příkaz v NetLogu, který iniciuje pohyb modelu. Typicky se využívá procedura „go“.
- **Measure** – Výraz z NetLoga, který určuje, jaké parametry nebo jaké chování je hledáno v experimentu. Tento výraz může nabývat jakékoliv numerické hodnoty a měření je přímo spojené s chováním modelu.
- **Measure if** – Podmínka, která kontroluje, kdy se má provádět měření. Lze například určit, že se má počítat až po 50 krocích (step).
- **Stop if** – Ukončovací podmínka, která udává, za jakých okolností se má ukončit běh modelu. (nepovinné).
- **Step limit** – Maximální počet běhů modelu (limit provedení akce **Step**).

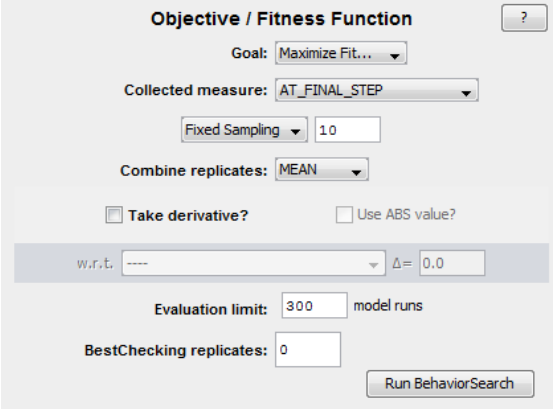
Výše je uvedeno, jaké data se vytáhnou z modelu, aby se mohlo spustit vyhledávání. Nicméně je nutné ještě specifikovat vyhledávací cíl (fitness function) viz obrázek 2.12.

- **Goal** – Určuje, zda maximalizovat nebo minimalizovat fitness.
- **Collected measure** – V průběhu běhu modelu je možné sbírat výsledky měření různými způsoby. BehaviorSearch nabízí tyto možnosti:
    - AT_FINAL_STEP: Výsledek měření po zadaném step limitu.
    - MEAN_ACROSS_STEPS: Průměr ze všech kroků modelu.
    - MEDIAN_ACROSS_STEPS: Medián ze všech kroků modelu.
    - MIN_ACROSS_STEPS: Minimum ze všech kroků modelu.
    - MAX_ACROSS_STEPS: Maximum ze všech kroků modelu.
    - VARIANCE_ACROSS_STEPS: Rozptyl mezi kroky modelu.
    - SUM_ACROSS_STEPS: Součet ze všech kroků modelu.
- **Sampling** – Určuje, kolikrát má být spuštěn model. Pouze jedno spuštění modelu, nemusí dávat reprezentativní vzorky měření, proto se využívá opakované spuštění se stejnými parametry, ale jiném nastavení náhodného vzorku.
- **Combine replicates** – Pokud se využije „Samplingu“, určuje, jakým způsobem mají být výsledky měření kombinovány. Je šest možností kombinací výsledků:
    - MEAN (průměr): Pravděpodobně nejpoužívanější funkce.
    - MEDIAN: Může být vhodnou volbou, pokud model občas vykazuje obrovsky vysokou nebo naopak nízkou hodnotu, která by pak byla ignorována.
    - MIN/MAX: Může být využita, pokud tyto obrovské výkyvy jsou hledány.
    - VARIANCE (rozptyl): V případě nestálých výsledků modelů.
    - STDEV (směrodatná odchylka): Stejné jako u rozptylu, akorát má lepší interpretaci.
- **Take derivative?** – Někdy je vhodné najít bod ve vyhledávacím prostoru, kde změna v chování měření je maximalizována

(minimalizována) s ohledem na malou změnu v nějakém parametru. Zaškrtnutí této možnosti umožňuje maximalizovat nebo minimalizovat přibližný derivát z fitness funkce s ohledem na konkrétní parametr a jeho změnu (delta).

  - Use ABS value? – Výsledný rozdíl funkce bude vždy kladný.

- **Evaluation limit** – Určuje, kolikrát se pustí vyhledávací proces modelu předtím, než skončí.
- **BestChecking replicates** – Počet dodatečného spuštění modelu, které bude provedeno v případě domnělého nalezení lepšího nastavení parametrů. Důvod této funkce je, že modely bývají stochastické a tímto se předejde neobjektivnosti fitness funkce. Výhoda této funkce oproti samplingu je, že se spustí, pouze pokud najde lepší parametry, a tím ušetří čas celému algoritmu. Pokud je tento parametr nastaven, pak je zobrazován graf pouze s hodnotou „rechecked“. Tyto dodatečné spuštění se nepočítají do celkového počtu spuštění modelu.



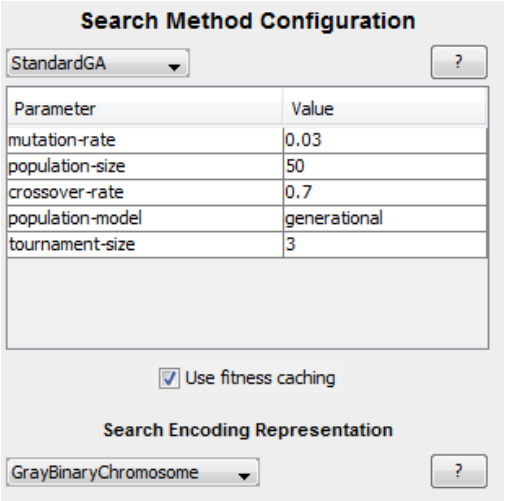
Obrázek 2.12: Určuje vyhledávací cíl

2.3.5. VÝBĚR VYHLEDÁVACÍHO ALGORITMU

Podkapitola inspirována podle [22].

Poslední volba pro běh experimentu je vyhledávací algoritmus. Tato volba obsahuje také výběr reprezentace kódování pro vyhledávací prostor a nastavení nezbytných parametrů pro vyhledávací algoritmus.

Výběr vyhledávacího algoritmu, nastavení jeho parametrů a reprezentace kódování může být velmi složitá a nesrozumitelná úloha. Není vždy zřejmé, které nastavení je vhodné pro konkrétní problém. Špatně zvolené nastavení vyhledávacího algoritmu může způsobit velmi pomalý postup nebo se úplně zastavit. Toto může být zapříčiněno tím, že vyhledávané chování není v modelu zastoupeno, nebo se musí zkusit jiné vyhledávací algoritmy.



Obrázek 2.13: Výběr a parametrizování vyhledávacího algoritmu

Vyhledávací prostor se skládá z přípustných kombinací nastavení parametrů modelu z kapitoly 2.3.3. BehaviorSearch podporuje čtyři různá kódování:

- **MixedTypeChromosome** – Toto kódování je nejvíce podobné tradičním parametrům agentových modelů. Každý parametr je uložen odděleně se svým datovým typem. Mutace se pak aplikuje na každý parametr zvlášť podle datového typu.
- **StandardBinaryChromosome** – V tomto kódování je každý parametr přetypován na string binárních číslic a tyto sekvence jsou pak přeformovány do velkého pole bitů. Mutace a křížení jsou pak prováděny na základě bitů jako na příkladu v kapitole 2.2.4.
- **GrayBinaryChromosome** – Velmi podobné chování jako u StandardBinaryChromosome. Pouze s tím rozdílem, že numerické hodnoty jsou kódovány do binárních řetězců pomocí Greyova kódu.

- **RealHypercubeChromosome** – Každý parametr (číselný nebo jiný) je representován reálnými proměnnými.

Dále je nutné zvolit vyhledávací algoritmus. BehaviorSearch poskytuje následující čtyři volby vyhledávacího algoritmu:

- **RandomSearch (náhodné vyhledávání)** - Naivní základní algoritmus, který na počátku jednoduše a náhodně vygeneruje set parametrů, pak jeden po druhém vypočítává fitness funkci pro každý krok běhu modelu a na konci vrátí nejlepší nalezené nastavení parametrů. RandomSearch pravděpodobně nebude nejefektivnější vyhledávací volba, ale je to velmi přímočará metoda.
- **MutationHillClimber (horolezcův algoritmus)** – Tento algoritmus začíná s náhodnou lokací (nastavení parametrů) ve vyhledávacím prostoru, pak opakovaně generuje lokace sousedů (použitím mutací) a pohybuje se do nové lokace jenom v případě, že je lepší než ta stará. Má dva parametry:
    - **Mutation-rate**: Určuje pravděpodobnost mutace.
    - **Restart-after-stall-count:** Určuje, po kolika neúspěšných krocích algoritmu (žádní sousedé nemají lepší lokaci) vyhledá novou náhodnou lokaci, kdekoliv ve vyhledávacím prostoru.
- **SimualtedAnnealing (simulované žíhání)** – Algoritmus je podobný předchozímu, akorát se může přesunout i do horší lokace, než je jeho aktuální, ale jen s určitou pravděpodobností založenou na „teplotě“ systému, která klesá časem. Čím větší teplota, tím větší se provádí změny. Má čtyři parametry:
    - **Mutation-rate:** Určuje pravděpodobnost mutace.
    - **Restart-after-stall-count:** Pokud se nedaří dostat do nové lokace po X pokusech, resetuje teplotu a přesune se do nové lokace.
    - **Initial-temperature:** Základní teplota systému (rozumnou volbou je průměrný očekávaný rozdíl ve fitness funkci mezi náhodnými dvěma body ve vyhledávacím prostoru).
    - **Temperature-change-factor:** Aktuální teplota systému, která je násobena tímto číslem v každém kroku. Číslo musí být menší než 1.

- **StandardGA (standardní genetický algoritmus)** – Klasický genetický algoritmus. Má pět parametrů**:**
    - **Mutation-rate:** Určuje pravděpodobnost mutace.
    - **Crossover-rate:** Určuje pravděpodobnost křížení.
    - **Population-size:** Určuje velikost populace v každé generaci.
    - **Tournament-size:** Určuje počet jedinců, kteří soutěží v turnajovém výběru. Větší hodnota způsobuje GA populaci konvergovat rychleji. Obvykle 2 nebo 3 je vhodná hodnota.
    - **Population-model:** má tři možnosti:
        - **Generational:** Celá populace se vymění najednou.
        - **Steady-state-replace-random:** Pouze jeden jedinec je náhodně vyměněn v každém kroku.
        - **Steady-state-replace-worst:** Pouze nejhorší jedinec je vyměněn v každém kroku.

Mutation-rate ovlivňuje rozdílné vyhledávací prostory rozdílně. V StandardBinary a GreyBinary kódování je mutation-rate pravděpodobnost mutace každého bitu (genu), kdežto u MixedType a RealHypercube je to pravděpodobnost mutace celého parametru. Tím pádem pro chromozomy typu MixedType a RealHypercube by měla mít pravděpodobnost mutace okolo 0,5, kdežto u binárního kódování kolem 0,02.

Ve výběru vyhledávacího algoritmu je také zaškrtávací pole „Use fitness caching“. Toto nastavení kontroluje, zda vyhledávací algoritmus uchovává fitness funkci pokaždé kdy je vypočítána. A nemusí jí tak přepočítávat pokud, se vyhledávání vrátí na stejné parametry, které už byly nastaveny.

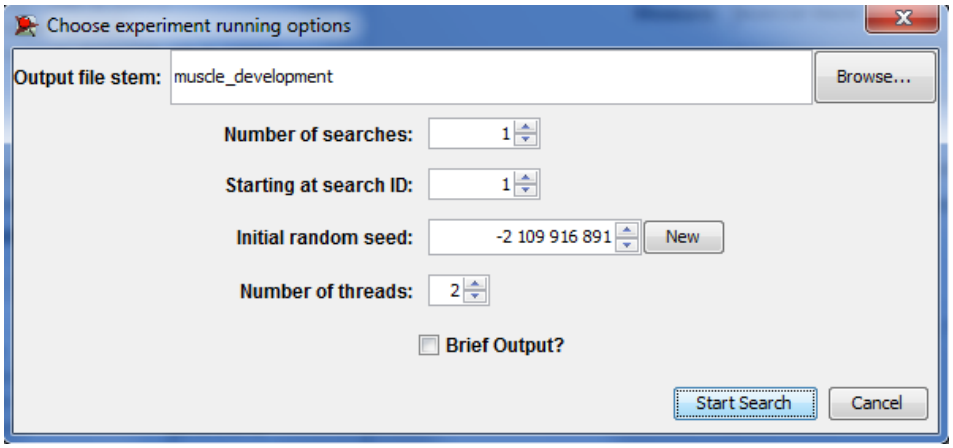
2.3.6. SPUŠTĚNÍ EXPERIMENTU

Podkapitola inspirována podle [22].

Po stlačení tlačítka „Run BehaviorSearch“ se zobrazí dialog pro nastavení dalších možností vyhledávání (viz obrázek 2.14).

- **Output file stem** – Odkazuje, kam se mají uložit data z vyhledávání.
- **Number of searches** – Určuje, kolikrát se celý proces vyhledávání opakuje.
- **Starting at search ID** – Ovlivňuje pouze identifikační číslo, které je pak zobrazeno ve výsledcích vyhledávání.

- **Initial random seed** – Náhodné číslo pro pseudonáhodný generátor, kde se začne s vyhledáváním. Vhodné pro pozdější opakování stejného vyhledávání.
- **Number of threads** – Použitím více vláken se může razantně zvýšit výkon vyhledávacího prostoru. Ovlivňuje pouze výkon, nikoliv výsledek vyhledávání.
- **Brief Output**? – BehaviorSearch vytváří různé výstupy z modelu a některé z nich mohou být velmi velké (obsahují výsledky všech běhů modelů apod.). Při zaškrtnutí tohoto pole se dva největší soubory nevytvoří.



Obrázek 2.14: Dialog spouštění vyhledávání

Výstup z vyhledávacího experimentu se nachází v několika různých souborech ve formátu CSV (čárkou oddělené hodnoty). Dále se nachází list výstupních souborů:

- **název.searchConfig.xml** – XML soubor, který neobsahuje data z výstupu modelu, ale obsahuje pouze použité nastavení pro vyhledávání. Tento soubor pak lze nahrát do BehaviorSearch pro využití v podobných vyhledávání.
- **název.modelRunHistory.csv** – Soubor obsahuje řádek pro každý chod agentového modelu. Ukazuje nastavení parametrů, se kterými model běžel a také zobrazuje hodnoty měření.
- **název.objectiveFunction.csv** – Soubor obsahuje řádek pro každé vyhodnocení fitness funkce.

- **název.bestHistory.csv** – Soubor obsahuje pouze fitness funkce lepší, než předchozí vyhledávání.
- **název.finalBests.csv** – Soubor obsahuje pouze krátký popis nejlepších nalezených parametrů v jednom vyhledávání. To znamená, že pokud někoho zajímá výsledek z vyhledávání, nalezne ho v tomto souboru.
- **název.finalCheckedBests.csv** – Podobný „.finalBests.csv“ souboru, ale obsahuje nastavení parametrů, které byly nalezeny nezávisle na fitness funkci. Nastavení parametrů v tomto souboru bude pravděpodobně nejlepší.

K využívání výsledků se používá nastavení parametrů ze souborů „finalBests“, které se pak nastaví v původním NetLogo modelu. Potom je možné vidět, jak se model chová za těchto podmínek. Toto může pomoct odhalit neočekávané nebo zajímavé chování v modelu.

3. PRAKTICKÁ ČÁST

3.1. VÝBĚR MODELŮ PRO OPTIMALIZACI

Optimalizaci modelů bude řešit nástroj BehaviorSearch, který výhradně pracuje s modely vytvořené NetLogem. V „Models library“ NetLoga je možné nalézt mnoho funkčních a otestovaných modelů. Z těchto modelů je nutné řešit jejich vhodnost pro optimalizační účely. Takovéto vlastnosti modelů jsou:

- Široká škála chování podle přednastavených parametrů.
- Nejednoznačná reakce chování na nastavení parametrů.
- Možnost nalezení vhodného typu měření.

Modely, splňující tyto požadavky, jsou převážně v kategorii biologických a sociologických. Určitě je možné vytvořit zajímavé experimenty i v jiných vědních oborech, ale ty vyžadují vyšší nároky na znalosti z toho odvětví.

Ve velké většině modelů krajní nastavení parametrů (ať už maximální nebo minimální) způsobuje „nejextrémnější“ chování, a tedy optimalizovat takovou fitness funkci postrádá smysl. Je také důležité najít vhodné a smysluplné měření, které toto snadno předpokládané chování mít nebude, což vyžaduje hlubší znalost každého zkoumaného modelu.

Mezi takovéto modely patří:

3.1.1. MUSCLE DEVELOPMENT

Podkapitola je vytvořena podle [23].

Model popisuje posilování jedince a zobrazuje progres jeho hormonů v průběhu cvičení. Pomáhá pochopit faktory při hledání rovnováhy hormonů pro ideální růst svalových vláken při posilování. Těmito faktory jsou:

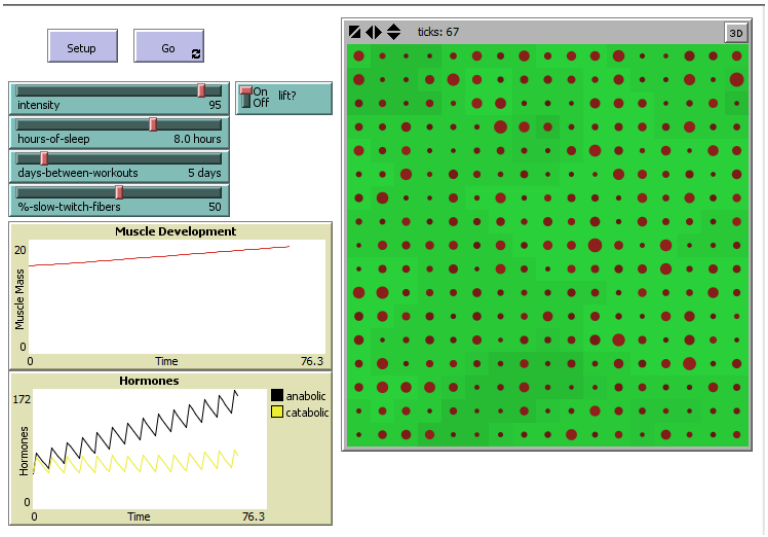
- **Frekvence** – Určuje, jak často jedinec posiluje pro co největší nárůst svalových vláken. Jestliže jedinec posiluje příliš často, tak se svaly nestačí regenerovat. Pokud ovšem jedinec neposiluje dostatečně často, nebude dostatek prostředků pro zvětšení svalů.
- **Spánek** – Tělo získává většinu energie během spánku. Jestliže tělo nemá dostatek regenerace ze spánku, bude složité nabrat svalovou hmotu.

- **Intenzita** – Určuje, jak tvrdě a efektivně jedinec pracuje v posilovně. Tento faktor pak závisí na tom, jak rychle je jedinec schopen vytvořit svalovou hmotu. Ovšem, čím větší intenzita tréninku, tím více regenerace potřebuje.
- **Genetika** – Každý jedinec má jinou genetickou výbavu. Pomalejší svalová vlákna znamenají větší vytrvalost, ale menší svalovou hmotu. Oproti tomu rychlejší svalová vlákna jsou schopná dosáhnout větší svalové hmoty, ale mají mnohem menší vytrvalost.
- **Strava** – Špatná strava může znamenat menší nebo žádný svalový růst. V tomto modelu se počítá s dokonalou stravou.

Těchto 5 faktorů musí být v souladu, aby vytvořily rovnováhu pro optimální růst svalové hmoty. Neexistuje ideální kombinace parametrů, v každém stavu se může stavba této ideální kombinace měnit.

Veškerý trénink svalstva vyvolává odpovědi od těla v podobě hormonů. Hormony, nezbytné pro vývoj svalů, jsou rozřazeny do dvou kategorií: katabolické hormony a anabolické hormony. Katabolické hormony rozkládají bílkoviny a odbourávají tak svalovou hmotu, a zároveň pomáhají znovuvytvořit silnější svaly za pomoci anabolických hormonů.

Model simuluje tyto efekty a zobrazuje detail svalu z pohledu svalového vlákna. Po spuštění modelu se rozeběhne svalové vlákno a podle přednastavených parametrů podává signály ohledně nárůstu hormonů v daném místě vlákna. Hormony pak ovlivňují vývoj svalového vlákna.



Obrázek 3.1: Model „Muscle Development“ v průběhu

Kolečka reprezentují svalové vlákno. V jejich pozadí je zobrazena „buněčná tekutina“, kterou zastupují anabolické a katabolické hormony. Světlejší zelená představuje více anabolické prostředí (růst svalů). Světlejší žlutá zase reprezentuje katabolické prostředí (odbourávání svalů). Na obrázku č. 3.1 je tedy možné vidět spíše anabolické prostředí, ve kterém rostou svalová vlákna.

V tomto modelu nelze předem jednoznačně stanovit ideální kombinaci parametrů pro zajištění co nejlepšího možného růstu svalových vláken. Vhodným měřením tak může být například kolik svalové hmoty je jedinec schopen získat za určitou dobu cvičení.

3.1.2. WOLF SHEEP PREDATION

Podkapitola je vytvořena podle [21].

Model představuje rovnováhu ekosystému, kde figuruje dravec a kořist. Takový systém je nazýván stabilním, pokud se udrží v průběhu doby, i napříč kolísáním v populaci druhů. Oproti tomu je nazýván nestabilním, pokud jeden z druhů zahyne.

Jsou zde dvě hlavní varianty modelu.

V první variantě modelu se pohybují ovce a vlci náhodně po krajině. Vlci shánějí ovce, aby se nakrmili. Každý krok stojí vlka energii, což znamená, že musí sníst ovce, aby doplnil energii. Pokud se mu to nepodaří, zahyne. Aby populace mohla pokračovat, ovce i vlci mají svoji pravděpodobnost reprodukce. Tato varianta vytváří zajímavé populační procesy, ale je velmi nestabilní.

Druhá varianta obsahuje, kromě vlků a ovcí, i trávu. Chování vlků je stejné jako v první variantě, ale ovce se musí pást trávou, aby si udržely svoji energii, jinak zahynou. Tato varianta je komplexnější a mnohem stabilnější.

Model obsahuje devět parametrů a těmi jsou:

- **INITIAL-NUMBER-SHEEP** – určuje velikost populace ovcí při prvním běhu modelu.
- **INITIAL-NUMBER-WOLVES** – určuje velikost populace vlků při prvním běhu modelu.
- **SHEEP-GAIN-FROM-FOOD** – množství získané energie ovce po požití trávy.
- **WOLF-GAIN-FROM-FOOD** – množství získané energie vlka po požití ovce.

- **SHEEP-REPRODUCE** – šance zreprodukování ovce v každém kroku modelu.
- **WOLF-REPRODUCE** – šance zreprodukování vlka v každém kroku modelu.
- **GRASS?** – určuje, zdali se použije faktoru trávy v modelu.
- **GRASS-REGROWTH-TIME** – čas, za jak dlouho doroste tráva po požití od ovce.
- **SHOW-ENERGY?** – pokud je zaškrtnuta, tak zobrazuje energii vlků a ovcí, nemá vliv na funkci modelu.

Model je zobrazen na obrázku č. 2.8 a na obrázku č. 3.5. Byl vybrán z důvodu, že obsahuje širokou škálu chování, od ekosystému nestabilního až po stabilní ekosystém. Obsahuje nejednoznačné chování. Neví se, kdy a jaké parametry pomůžou ekosystému. Vhodné měření může být například, jak dlouho je schopen vydržet ekosystém bez zapnuté funkce trávy.

3.1.3. ALTRUISM

Popis modelu je inspirován podle [26].

Jedná se o model z kategorie evoluční biologie. Modeluje populační genetiku ovlivněnou sociálními a okolními vlivy. Model má dva typy agentů a to: altruisté a egoisté.

Základem modelu je souboj altruistů a egoistů o každé místo ve světě vstupující do genetické loterie. Tento souboj je možné si představit několik zasetých semínek a pouze nejdominantnější semínko se promění v rostlinu.

Za obvyklých okolních podmínek egoisté vyhrávají a populace altruistů vede k záhubě. Jak bude vysvětleno níže, pokud se nastaví okolní podmínky trochu ostřeji, altruistická populace má možnost se vyrovnat nebo dokonce dominovat té egoistické.

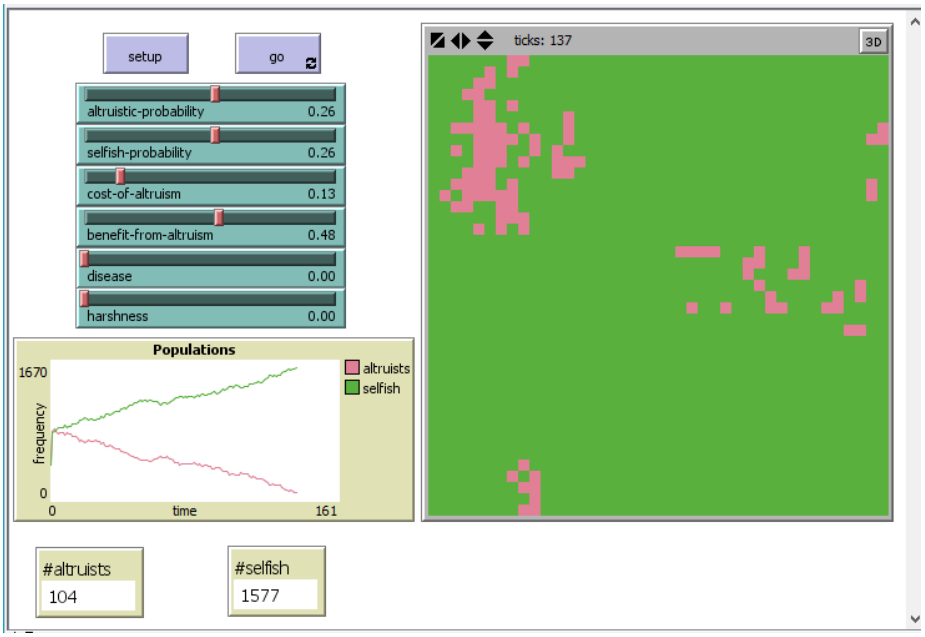
Každé pole je agent, který má svojí fitness. Každé pole bojuje s okolními čtyřmi pro získání pole na svojí stranu. Tedy altruista nebo egoista. Aby bylo pole úspěšné v loterii, musí získat ostatní pole na svojí stranu. Každé pole vypočítává svojí fitness pomocí této rovnice:

- Pokud je altruista: 1 – náklad Altruismu + (počet altruistů v sousedství / 5 * užitek z Altruismu)

- Pokud je egoista: 1 + (počet altruistů v sousedství / 5 * užitek z Altruismu).

Tedy fitness egoistických polí bude větší nežli altruistických. Například při nákladu Altruismu 0,2 a užitku z Altruismu 0,5. Bude pro pole obklopené dvěma altruisty a dvěma egoisty, fitness: 1 – 0,2 + (3/5 * 0,5) = 1,1.

Potom co se vypočítá fitness pro každé pole, tak se podívá ke čtyřem sousedům. Každý z těchto pěti polí včetně sebe, je pak předán do genetické loterie o prostřední pole. Hodnoty fitness funkcí se pro každou skupinu sečtou. Poté se určí procentuální zastoupení v sousedství. Z čehož se vypočítává pravděpodobnost pro získání pole. Vzhledem k tomu, že egoisté mají v průměru větší fitness, budou mít větší šanci pro získání polí. Jak je vidět na obrázku č. 3.2, při základním nastavení modelu, altruistů je mnohem méně jak egoistů.



Obrázek č. 3.2: Model Altruism se základním nastavením parametrů

Model má 6 nastavitelných parametrů:

- **ALTRUISTIC-PROBABILITY –** určuje, počáteční procento altruistů
- **SELFISH-PROBABILITY –** určuje, počáteční procento egoistů
- **ALTRUISM-COST –** určuje, náklady Altruismu ve výše zmíněné rovnici
- **BENEFIT-FROM-ALTRUISM –** určuje, hodnotu užitku z Altruismu

Následují dva parametry z okolního prostředí:

- **HARSHNESS –** nastavuje hodnotu odolnosti prázdného pole, proti získání ostatními agenty.
- **DISEASE –** nastavuje hodnotu pro možnost zahynutí agenta v obsazených polích. Tento parametr se promítne do genetické loterie.

Model je vybrán z důvodu širokého nastavení parametrů a nejednoznačného chování po nastavení těchto parametrů. Existuje zde pár zajímavých možností na měření. Například, při jakém nastavení je možná rovnováha mezi egoisty a altruisty.

3.2. HYPOTÉZY

Každý agentový model obsahuje spoustu parametrů různého účelu. Některé parametry jsou zahrnuty přímo v modelu a uživatel s nimi nemůže manipulovat. Další parametry jsou uživatelské, může s nimi operovat a nastavovat tak chování modelu.

Vybrané modely budou postupně testovány na výslednou fitness funkci a na porovnání vyhledávacích algoritmů z programu BehaviorSearch. Porovnávat se budou na základě časové náročnosti a kvality výsledku fitness funkce.

Zde bude představena hypotéza pro každý vyhledávací algoritmus zvlášť.

- **RandomSearch (náhodné vyhledávání)** – Algoritmus založený na náhodně nastavených parametrů v každém kroku. Tento algoritmus by měl být ze všech nejrychlejší, ale zároveň by měl být také nejméně přesný. Pokud by se vyskytl nějaký velmi dobrý výsledek, bude to hlavně díky prvku náhody.
- **MutationHillClimber (horolezcův algoritmus)** – Algoritmus, který se přesouvá pouze do lepšího nastavení parametrů, než předchozí. Předpokladem u tohoto algoritmu je menší časová náročnost, ale o něco větší, než u úplně náhodného přístupu. Zároveň řešení může konvergovat k lokálnímu maximu, a tím může být fitness funkce ohrožena.
- **SimualtedAnnealing (simulované žíhání)** – Na rozdíl od horolezeckého algoritmu s určitou pravděpodobností se může posunout do horšího nastavení parametrů. Tím částečně odpadá problém konvergence k lokálnímu maximu, ale stoupá tak i časová náročnost.
- **StandardGA (standardní genetický algoritmus)** – Klasický genetický algoritmus. Bude pravděpodobně nejnáročnější na čas, ale měl by předvést také nejlepší výsledek fitness funkce.

3.2.1. HYPOTÉZA MUSCLE DEVELOPMENT

V této části bude záležet na nejlepším možném nastavení parametrů pro model „Muscle Development“. V tomto případě bude v experimentech nejvíce zajímavé sledovat nárůst svalů za určitou časovou jednotku. V modelu lze nastavit pět parametrů a to: intenzita (intensity), hodiny spánku (hours-of-sleep), dny odpočinku (days-between-workouts), procento pomalých svalových vláken (%-slow-twitch-fibers) a zda jedinec posiluje (lift?). Za předpokladu vyhledávání největšího nárůstu svalů je jasné, že parametr, zda jedinec posiluje, bude vždy zapnut, tedy přepínač „lift?“ bude na „on“. Pro experimentální účely bude také důležité nastavit jednotné procento pomalých svalových vláken pro všechny experimenty, bude nastaveno 50 procent.

V experimentech se tedy bude pracovat pouze se třemi parametry a nalezení jejich co možná nejideálnější kombinace pro co největší nárůst svalů. V modelu se bude hledat proměnná „muscle-mass“, která určuje množství svalové hmoty. Předpoklad pro nejlepší nastavení parametrů u tohoto modelu je, že pokud bude vyšší intenzita cvičení, bude zapotřebí více dní regenerace mezi cvičeními. Oproti tomu pokud intenzita nebude tak vysoká, může jedinec posilovat častěji. Počet hodin spánku, by měl být nastaven okolo 8-9 hodin. [24,25]

3.2.2. HYPOTÉZA WOLF SHEEP PREDATION

V modelu „Wolf Sheep Predation“ se bude pracovat s první variantou modelu popsanou v kapitole 3.1.2. Tento ekosystém je velmi nestabilní a může být zajímavé sledovat, při jakém nastavení parametrů je možné ho co nejdéle udržet. Podmínkou udržení stabilního ekosystému je, že žádný ze zvířecích druhů nezahyne nebo se nepřemnoží. Přemnožení populace v tomto testu znamená překročení hranice 1000 jedinců. V takovém případě se běh modelu ukončí a pokračuje se s jiným nastavením parametrů.

Model má devět nastavitelných parametrů. Parametr ukazování energie jedinců (show-energy?) je možné ignorovat. Test to neovlivní, jde pouze o zobrazovací parametr. Důležitý parametr je použití trávy v modelu (grass?), pro aplikování první varianty musí být vypnutý. Vzhledem k vypnutému parametru trávy je zbytečný i další parametr, čas na znovuobnovení trávy (grass-regrowth-time). Dalších šest parametrů bude zkoumáno ve vyhledávacích algoritmech. Tři

stejné parametry pro každý druh. Prvním je počáteční hodnota populace jedinců (initial-number-sheep a initial-number-wolves), druhým je kolik získají energie z potravy (sheep-gain-from-food a wolf-gain-from-food) a posledním je procento další reprodukce (sheep-reproduce a wolf-reproduce).

Kombinací těchto šesti parametrů se hledá co nejideálnější rovnováha v ekosystému. Pro toto zjištění se použije doba chodu modelu, což v NetLogu představují „ticks“. Každý „tick“ představuje jednu proceduru „go“, což znamená, že čím více „ticks“, tím déle běží proces ekosystému a je tím tak nejstabilnější.

3.2.3. HYPOTÉZA ALTRUISM

U tohoto modelu se bude zkoumat vyrovnanost agentů altruistů a egoistů v určitý časový úsek. Údaje z tohoto časového úseku budou pak průměrovány. Zjištění vyrovnanosti egoistů a altruistů se provede skrz směrodatnou odchylku těchto dvou populací agentů. Předpokladem u tohoto modelu pro směrodatnou odchylku je, že pokud se bude blížit nule a druhá populace bude velmi blízká nule, tak bude fitness funkce co nejlepší. Pro zamezení tohoto problému, bude lehce pozměněn model a hodnota měření (viz. experiment Altruism).

Model Altruism má 6 přednastavitelných parametrů, všech 6 parametrů se bude zkoumat v experimentech. Vyrovnanost v určitém časovém úseku se dá předpokládat, pokud obě populace budou mít stejnou pravděpodobnost výskytu v načtení modelu. Dále pokud bude náklad a užitek z Altruismu nulový a tím se vlastně egoistická výhoda vymaže. Stejně tak by měla být nulová hodnota parametru „disease“ a „harshness“ aby nebyl ovlivňován okolními vlivy. V takto nastaveném parametrovém prostoru by hypoteticky měla být fitness funkce co nejvyšší.

3.3. EXPERIMENTY

Všechny experimenty se budou provádět s ohledem na porovnání vyhledávacích algoritmů. Algoritmy budou mít ve všech experimentech podobné nastavení počátečních parametrů.

Na obrázku č. 3.3 je vidět vyhledávací algoritmus pro náhodné vyhledávání, který je bez parametrů. Postupně budou představeny všechny vyhledávací algoritmy s jejich parametry. Zároveň pro všechny vyhledávací algoritmy bude použito kódování „GrayBinaryChromosome“, tedy v binárních řetězcích podle Grayova kódu.

- **RandomSearch (náhodné vyhledávání) –** Naivní algoritmus, zkouší náhodně různé kombinace parametrů, a tedy nelze na tento algoritmus žádné další parametry aplikovat.
- **MutationHillClimber (horolezecký algoritmus) –** Algoritmus začíná v náhodné lokaci (nastavení parametrů) a poté hledá lepší nastavení parametrů podle mutace. Pravděpodobnost mutace byla zvolena 0,05. Restartování algoritmu do nové náhodné lokace je zaručeno po 200 pokusech o novou lepší lokaci, tento parametr je nastaven z důvodu, že by algoritmus mohl uvíznout v lokálním maximu.
- **SimulatedAnnealing (simulované žíhání)** – Posouvá se i do horších lokací podle „teploty“ systému. S určitým procentuálním podílem každým krokem teplota systému klesá. Tento podíl je zde nastaven na 0,99, což je druhý parametr algoritmu. To znamená, že každým krokem se teplota systému sníží o jedno procento. Prvním parametrem algoritmu je, stejně jako u horolezeckého algoritmu, pravděpodobnost mutace. Určuje, jak často se má mutace objevit při vybírání nové lokace. Zde je opět nastavena na pravděpodobnost 0,05. Třetím parametrem nastavení tohoto algoritmu je počáteční teplota systému. Tento parametr by měl být nastaven na průměrný očekávaný rozdíl fitness funkce mezi dvěma náhodnými lokacemi (nastavením parametrů). Pro získání této informace se využijí výsledky z náhodného algoritmu. Každé měření se odečte od toho

předchozího a potom se použije aritmetický průměr absolutních hodnot rozdílů fitness funkce. Toto číslo se tedy zvolí jako iniciální teplota. Posledním parametrem tohoto algoritmu je restartování teploty v případě, že se neposune do nové lepší lokace po *x* pokusech. Zvolí se restartování po 300 pokusech o novou lokaci.

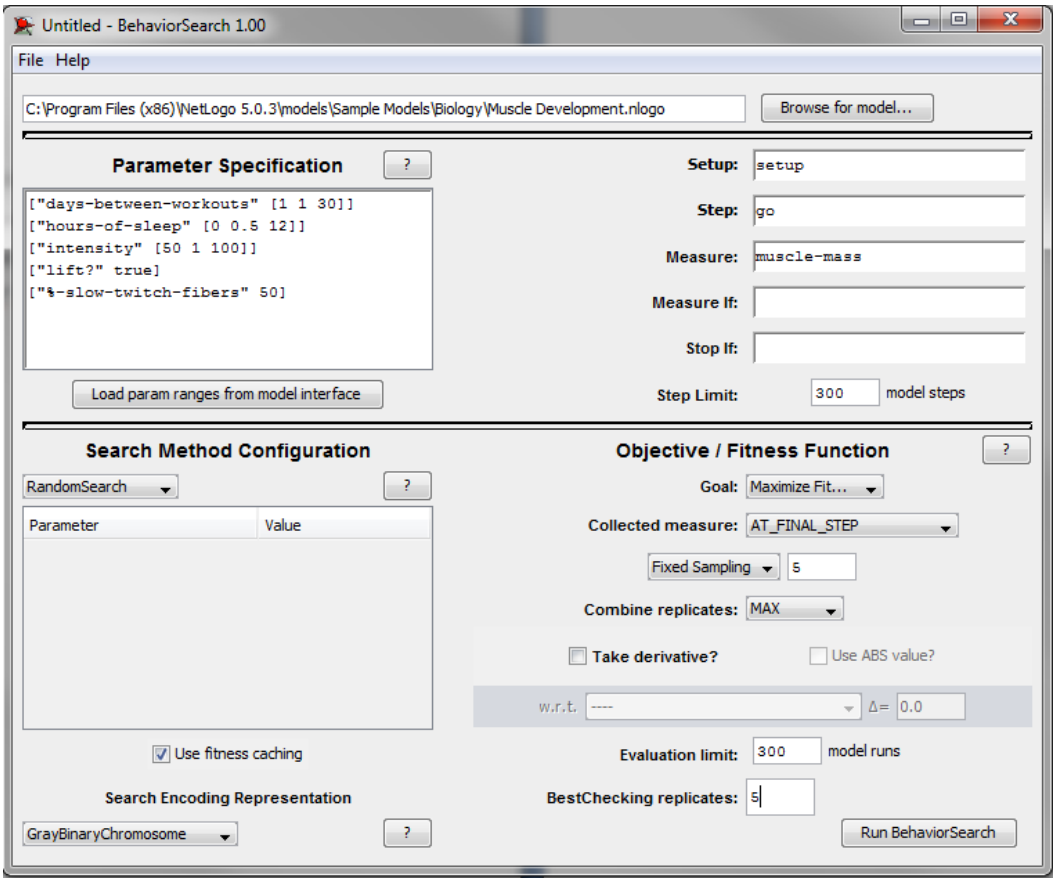
- **StandardGA (standardní genetický algoritmus)** – Algoritmus má pět nastavitelných parametrů. Prvním parametrem je možnost mutace v nové generaci, která bude nastavena, stejně jako u předchozích algoritmů, na 0,05. Velikost populace je nastavena na 30. Pravděpodobnost křížení mezi potomky bude nastavena na 0,7. Model populace je nastaven na „generational“, což znamená, že se vymění celá populace naráz. Velikost turnajové selekce je nastavena na 3. Toto nastavení doporučuje Holland [15] ve své knize.

3.3.1. EXPERIMENT MUSCLE DEVELOPMENT

V tomto experimentu se hledá fitness funkce, kterou představuje v modelu „Muscle Development“ proměnná muscle-mass. Muscle-mass ukládá součet velikostí svalových vláken a je tak hlavním měřítkem celkové svalové hmoty.

Na obrázku č. 3.3 je vidět nastavení nástroje BehaviorSearch pro experiment Muscle Development. Rozsah parametrů „days-between-workouts“, „hours-of-sleep“ a „intensity“ je ponechán z modelu v plném rozsahu a představuje tak 37 500 (30×25×50) různých možností, protože parametry „lift?“ a „%-slow-twitch-fibers“ představují jen jednu možnost, viz kapitola 3.2.

V návrhu měření se nastavuje model příkazem „setup“ a spouští se příkazem „go“. Hlavním zájmem měření je proměnná muscle-mass. V tomto případě bude velikost fitness funkce rovna proměnné muscle-mass. Podmínka pro měření a ani ukončovací podmínka se dávat nemusí, protože se měří do doby, dokud se model nespustí 300krát což představuje stav po 300 dnech cvičení.

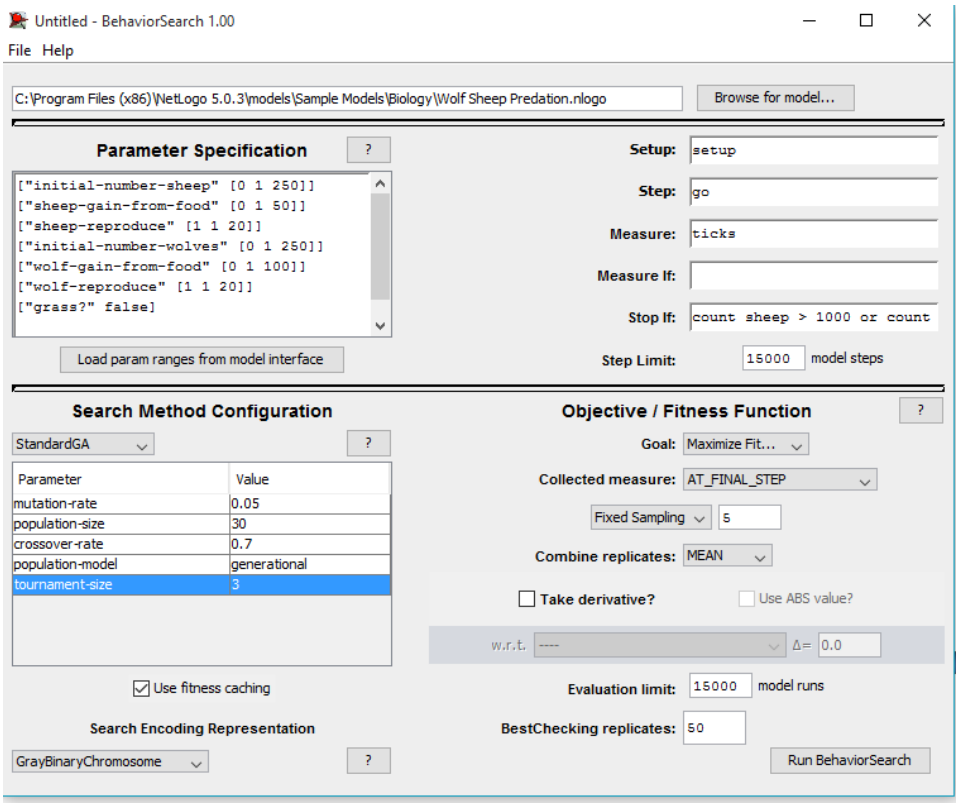


Obrázek č. 3.3: Nastavení pro experiment Muscle Development

Vyhledávacím cílem je funkci maximalizovat, hledá se co možná největší svalová hmota z modelu. Měření se bude sbírat až po uběhnutí 300 kroků modelu, tedy zvolí se funkce „AT_FINAL_STEP“. Jelikož je model založen částečně na náhodě a probíhá pokaždé lehce jinak, zvolí se spuštění modelu se stejnými parametry pětkrát a vezme se největší hodnota z těchto pěti spuštění. Pro kontrolu fitness funkce se model spustí dodatečně ještě pětkrát, když nalezne nové lepší nastavení parametrů.

3.3.2. EXPERIMENT WOLF SHEEP PREDATION

Hledaná fitness funkce je nejdelší úsek bez aplikování ukončovací podmínky. Tak se najde nejlepší kombinace parametrů pro udržení rovnováhy v ekosystému. Na obrázku č. 3.4 je vidět nastavení vyhledávacího procesu pro model „Wolf Sheep Predation“ v programu BehaviorSearch.



Obrázek č. 3.4: Nastavení pro experiment Wolf Sheep Predation

Rozsah pro šest vybraných parametrů pro vyhledávání byl nastaven na maximální hodnoty z modelu, s tím rozdílem, že u vlků není možná nulová reprodukce, kterou nabízí model. Je to z důvodu, aby byl ekosystém aktivní a neměl po celou dobu stejnou velikost populace vlků. Toto nastavení rozsahu parametrů představuje 129 807 260 400 (251×51×20x251x101x20x1) různých možností.

V návrhu měření, je klasické nastavení modelu přes příkaz „setup“ a spouštění přes „go“. Měřit se budou kroky v modelu. Podmínka pro měření není žádná. Ukončovací podmínka běhu modelu je důležitá pro princip měření. Podmínka je *„count sheep > 1000 or count wolves > 1000 or count sheep = 0 or count wolves = 0“*, tedy pokud počet ovcí nebo vlků je větší než 1000 (přemnožení

jednoho druhu) nebo pokud počet ovcí nebo vlků je roven nule (zahynutí jednoho druhu), tak se ukončí běh modelu a je započítáno aktuální množství kroků systémů „ticks“. Limit kroků modelu je nastaven na 15000 a nedá se předpokládat, že by se model dostat do tak pokročilé fáze, díky nastavení ukončovací podmínky.

Cílem vyhledávání je funkci maximalizovat. Hledá se co největší počet kroků, co lze modelem docílit do ukončovací podmínky. Sběr dat bude pochopitelně probíhat ve finálním kroku. Model bude spuštěn pokaždé pětkrát se stejnými parametry a tyto data z pěti běhů modelu budou pak zprůměrovány pro novou fitness funkci. Celkový počet běhů modelu je nastaven na 15000, kvůli velkému počtu možností a dostatečnému prostoru pro vyhledávací algoritmy. Algoritmy pak mají možnost vyzkoušet 3000 různých kombinací parametrů. Model disponuje vysokou stochasticitou a z toho důvodu je nastaven parametr „BestChecking replicates“ na 50. Zvýší se tím náročnost algoritmů, ale pro lepší výsledky je nutné každou novou fitness funkci otestovat mnoha běhy modelu, kvůli eliminaci stochasticity.

3.3.3. EXPERIMENT ALTRUISM

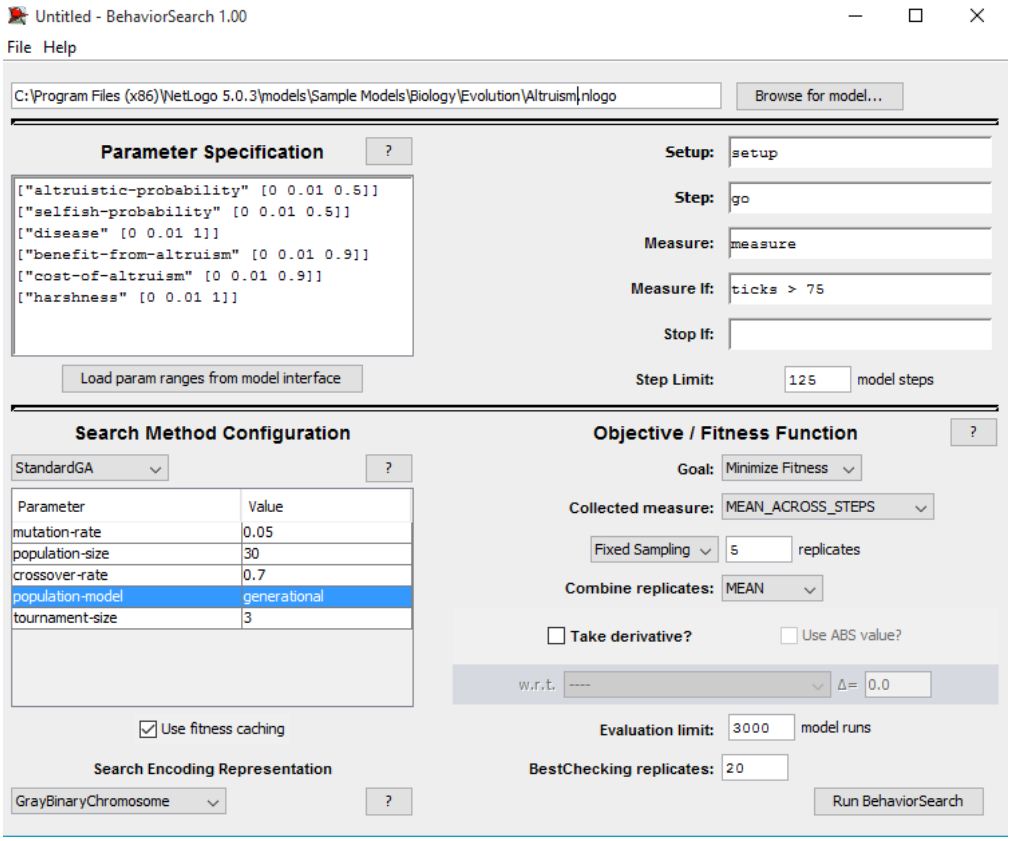
Hledaná fitness funkce je průměrná směrodatná odchylka mezi 75 až 125 kroky (ticky) modelu. Další koeficient pro měření je počet prázdných polí. Tento počet bude k průměrné funkci připočítáván. Z těchto všech požadavků se dá předpokládat fitness funkce, kde není mnoho prázdných polí a zároveň, kde je vyrovnaná populace egoistů a altruistů mezi 75 a 125 ticky.

Pro účel možnosti měření mezi 75 a 125 ticky bude lehce přepracován model Altruism. Protože tento model může skončit ještě před 75 ticky, kde zahynou obě populace agentů a vzniknou pouze prázdná pole. Při pouhém zrušení ukončovací podmínky by model měl 0 altruistů a 0 egoistů a směrodatná odchylka by byla 0 a tím by se toto nastavení stalo tou nejlepší fitness funkcí. Takže pokud jedna z populací klesla na 0, bude nastavena hodnota měření na 1000.

V modelu byla přidána globální proměnná „measure“, která bude poté měřena v experimentu. Proměnná measure je vypočítána pomocí směrodatné odchylky a přičtením pětiny prázdných polí, aby výsledná fitness funkce měla vysoký počet neprázdných agentů. Tento výpočet probíhá v proceduře „go“ a

vypadá takto*: „ set measure sqrt ((1 / 2) * (((count patches with [pcolor = green or pcolor = pink] / 2 - count patches with [pcolor = pink]) * (count patches with [pcolor = green or pcolor = pink] / 2 - count patches with [pcolor = pink])) + ((count patches with [pcolor = green or pcolor = pink] / 2 - count patches with [pcolor = green]) * (count patches with [pcolor = green or pcolor = pink] / 2 - count patches with [pcolor = green]))))) + (count patches with [pcolor = black] / 5)“.*

Jedná se o výpočet směrodatné odchylky mezi dvěma populacemi agentů a přičtení populace prázdných polí.



Obrázek č. 3.5: Nastavení pro experiment Altruism

Rozsah všech parametrů v modelu je od 0. V pravděpodobnosti altruistické a egoistické populaci je maximální hodnota 0,5. V užitku a nákladech Altruismu je maximální hodnota 0,9. A v „nemoci“ a „drsnosti“ modelu je maximální hodnota 1. Všechny parametry jsou krokovány po 0,01. Toto nastavení představuje 219 718 125 081 (51×51×101×101×91×91) různých kombinací parametrů. Tedy ještě téměř dvakrát více jako u „Wolf Sheep Predation“.

V návrhu měření (obrázek č. 3.5) se klasicky model nastavuje příkazem „setup“ a pouští příkazem „go“. Měření bude probíhat na nově vytvořené proměnné „measure“. A měřit se bude až po uplynutí 75 ticků modelů, tedy nastavení „ticks > 75“. Ukončovací podmínka zde není nutná. A měřit se bude až do 125 kroku modelu.

Vyhledávácím cílem je tentokrát funkci minimalizovat. Zájem je o nejnižší možnou směrodatnou odchylku mezi dvěma populacemi agentů, zvýšenou o pětinu prázdných polí. Při měření se bude používat pět replik modelů se stejným nastavením a z tohoto měření se vezme klasicky průměr. Spuštění modelů pro testování je nastaveno na 3000. Vzhledem k faktu, že model je opět velmi stochastický, použije se parametr „BestChecking replicates. A pokaždé pokud se najde lepší fitness funkce, tak se model dodatečně spustí 20krát pro zkontrolování nalezené fitness funkce.

3.4. VÝSLEDKY

Kapitola se zabývá prezentací všech učiněných experimentů v předchozích kapitolách.

3.4.1. VÝSLEDKY EXPERIMENTU MUSCLE DEVELOPMENT

V tabulce č. 3.1 je možné porovnat hodnoty měření z programu BehaviorSearch na jednotlivých algoritmech po 300 krocích. Testována byla časová náročnost, maximální fitness funkce a nastavení parametrů pro fitness funkci.

Vysvětlivky k tabulce č. 3.1:

- DBW – days between workouts (dny mezi cvičeními)
- HOF –hours of sleep (hodin spánku mezi dny)
- I – intensity (intenzita cvičení v procentech)

Algoritmus	Čas	Max fitness	DBW	HOF	I
RandomSearch	55:32	3269,26	9	5	88
SimulatedAnnealing	1:04:40	3319,58	15	4	95
MutationHillClimber	59:07	3286,19	13	4,5	97
StandardGA	1:06:29	3296,59	9	3,5	67

Tabulka č. 3.1: Porovnání algoritmů po 300 krocích

Výsledky z tabulky č. 3.1 potvrdily hypotézu ohledně časové náročnosti algoritmů. Jako nejrychlejší algoritmus se ukázalo náhodné vyhledávání, je následován, podle předpokladu, horolezeckým algoritmem, ale o nepatrný rozdíl. Dalším algoritmem v pořadí je simulované žíhání s 1 hodinou 4 minutami a 40 vteřinami. A jako nejpomalejší se jeví standardní genetický algoritmus.

Co se týče kvality výsledné fitness funkce, tedy takové nastavení parametrů, které po 300 dnech cvičení zaručuje největší nárůst svalové hmoty, není mezi algoritmy velký rozdíl, je to zapříčiněné krátkým procesem vyhledávání a nízkým počtem možností (37500).

U všech algoritmů má vliv na výsledek určitý prvek náhody, ať už je to prvek mutace nebo křížení. Každý genetický algoritmus se nějak vyvíjí a je možné, že i méně sofistikované algoritmy dosahují nepatrně lepšího výsledku, pokud se zrovna nevyvíjel optimálně a se správným prvkem náhody.

Hlavním cílem experimentu bylo optimalizovat parametry modelu „Muscle Development“ a zjistit, při jakém nastavení je docílen největší nárůst svalové hmoty. Překvapivým výsledkem je, že u všech algoritmů se jeví jako nejlepší doba spánku 3,5 – 5 hodin mezi jednotlivými dny. Podle několika studií např. [24], [25] je minimální doba spánku, potřebná pro zdraví rozvoj člověka, 7 – 8 hodin. Kde např. [24] doporučuje i větší dávky spánku, třeba až 10 hodin denně. Z tohoto lze usoudit, že model obsahuje nějaké chyby ve výpočtu potřeby spánku na posilování. I autor modelu poukazuje na to, že pokud člověk nemá dostatek spánku, je velmi obtížné získat svalovou hmotu. [25] Dalším zkoumaným parametrem jsou dny mezi cvičeními. Zde je velká rozmanitost výsledků od 5 do 29 dní. To stejné platí i pro intenzitu cvičení, která se pohybuje ve výsledcích od 52% do 97%. Výsledek dnů mezi cvičeními a intenzitou je logický provázaný. Když má jedinec velké přestávky, tak cvičí s vyšší intenzitou, kdežto pokud cvičí s menšími denními rozestupy, pak musí cvičit s nižší intenzitou.

Lidské tělo je natolik komplikovaný a individuální systém, že nelze dokonale namodelovat jeho chování při jakékoliv činnosti. Cílem těchto experimentů je nalézt takové nastavení parametrů, které se v určitých podmínkách chová nejlépe.

3.4.2. VÝSLEDKY EXPERIMENTU WOLF SHEEP PREDATION

Porovnání algoritmů je možné vidět v tabulce č. 3.2. Porovnává se podle času a maximální fitness funkce, která je použita z „BestChecking replicates“, tedy po 50 různých běhů modelu při jednom nastavení parametrů.

Vysvětlivky k tabulce č. 3.2 a obrázku č. 3.5:

Algoritmy:

- RS – random Search (algoritmus náhodného vyhledávání)
- SA – simulated Annealing (simulované žíhání)
- MHC – multiple hill climbing (horolezecký algoritmus)
- GA – genetic algorithm (standardní genetický algoritmus)

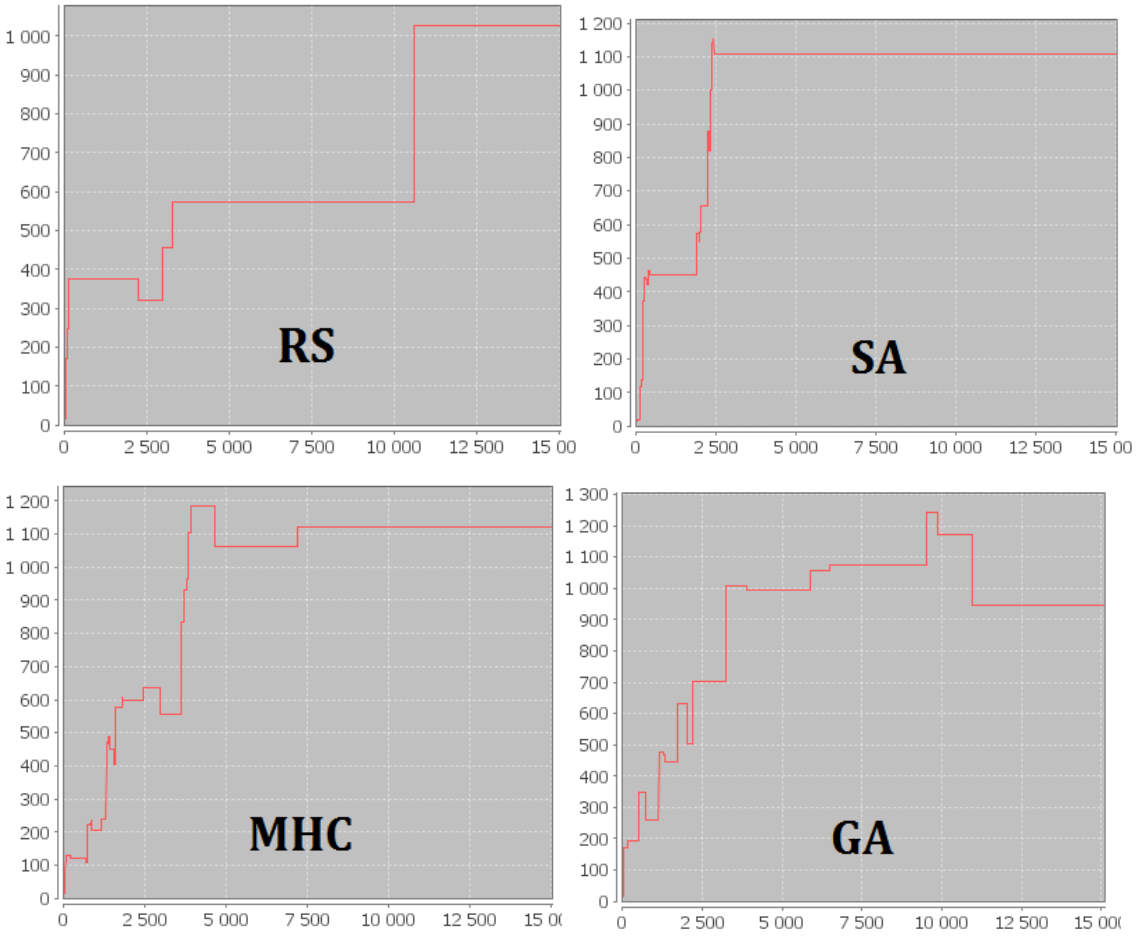
Hlavička tabulky:

- INS – initial number sheep (počáteční populace ovcí)
- SGFF – sheep gain from food (získaná energie ovcí z potravy)
- SR – sheep reproduce (reprodukce ovcí v procentech)
- INW – initial number wolves (počáteční populace vlků)
- WGFF – wolf gain from food (získaná energie vlků z potravy)
- WR – wolf reproduce (reprodukce vlků v procentech)

Algoritmus	Čas	Fitness	INS	SGFF	SR	INW	WGFF	WR
RS	7:15	1028,12	65	40	1	25	46	1
SA	26:02	1154,12	147	36	1	51	30	1
MHC	29:22	1186,16	139	24	1	52	28	1
GA	22:53	1244,08	152	24	1	45	27	1

Tabulka č. 3.2: Porovnání algoritmů po 15000 krocích

Výsledek tohoto experimentu přináší překvapivé výsledky, zejména ohledně časové náročnosti. Jako nejrychlejší je celkem očekávaně náhodný algoritmus, který zvládl proces vyhledávání za 7 minut a 15 vteřin. Což představuje o 15 minut rychlejší proces, nežli u druhého nejrychlejšího algoritmu. Zde nastává největší překvapení, a to, že druhým nejrychlejším je genetický algoritmus, poté ho následuje simulované žíhání a nejpomalejším je horolezecký algoritmus. Je to způsobeno především tím, že horolezecký algoritmus a simulované žíhání se více rozvíjí v průběhu vyhledávání, a tím se častěji spouští funkce „BestChecking replicates“, která zaručuje 50 dodatečných spuštění modelu. Toto chování algoritmů je možné sledovat i na obrázku č. 3.5, kde je zobrazen jejich vývoj v průběhu času. U simulovaného žíhání a horolezeckého algoritmu jsou častěji nalezené nové nejlepší fitness funkce a mají i podobný průběh, kde najdou nejlepší nastavení už po 2500 krocích. Obrázek č. 3.5 zobrazuje grafy, kde je zobrazena pouze hodnota „rechecked“ z BestChecking replicates a nikoliv fitness funkce, proto je možné v grafech vidět i pokles směrem dolů.



Obrázek č. 3.6: Vývoj algoritmů Wolf Sheep Predation z programu BehaviorSearch

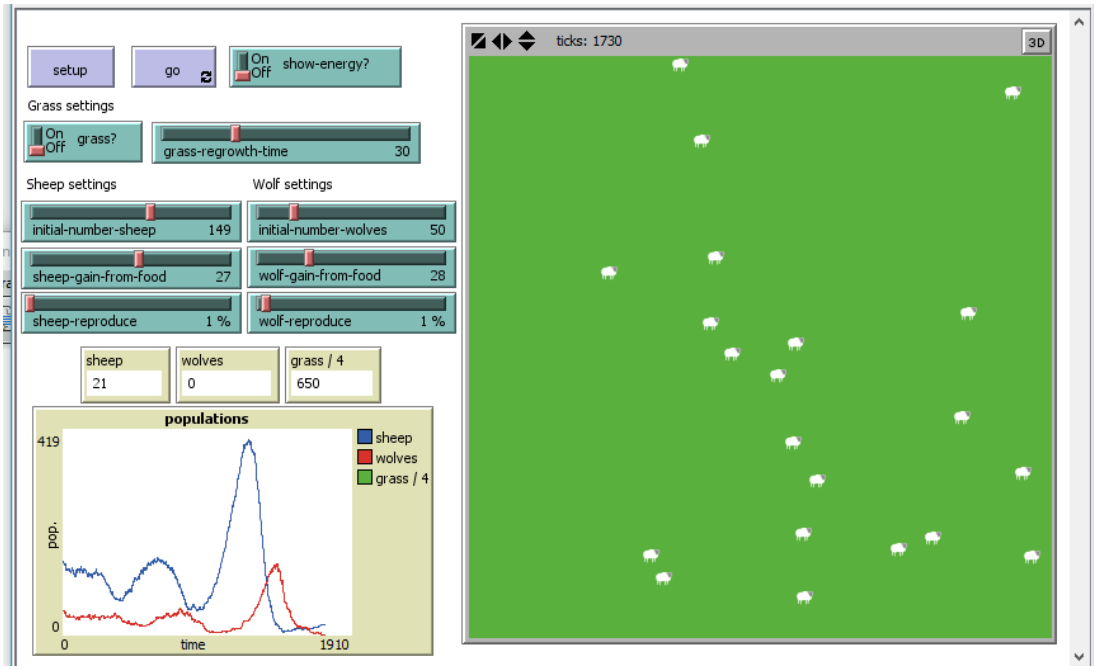
Výsledek fitness funkce, tedy té, která je překontrolovaná funkcí BestChecking replicates, vyhrává genetický algoritmus s 1244,08 průměrných ticků z 50 spuštění modelů. Následuje horolezecký algoritmus s 60 tickovým rozdílem a dále s nepatrným rozdílem simulované žíhání. Jako nejslabší algoritmus znovu vychází náhodné vyhledávání.

Nejlepší algoritmy mají výsledné nastavení parametrů podobné. Se 139 – 152 počátečním stavem ovcí, 24-36 energií získanou z potravy a procentem reprodukce 1. Co se týče populace vlků tak počáteční stav je mezi 45 až 52, kde získaná energie z potravy je mezi 27 až 30 a opět pouze 1% reprodukcí vlků.

Reprodukce obou druhů je vybrána algoritmy na co nejnižší úrovni. Je to logický fakt, vzhledem k tomu, že je potřeba vytvořit co nejvyrovnanější ekosystém v nestabilním prostředí, tak se populace nesmí příliš rychle měnit. Vysoký nebo nízký počet počáteční populace ovcí má podobný efekt na nerovnováhu

ekosystému. Pokud je počet ovcí vysoký, vlci se rychle namnoží, je jich více, než ovcí a hůře shánějí potravu, poté vymírají a rychle narůstá populace ovcí, což vede k přemnožení ovcí. Pokud je nízký počet ovcí tak se tento proces urychlí. Celá tato rovnice platí i pro počáteční stav vlků, když je jich více jak ovcí, těžce shánějí potravu a znamená to pro ně rychlé vymírání a obrovský nárůst populace ovcí.

Na obrázku č. 3.6 je možné vidět výsledek optimalizování modelu, podle výsledné funkce z heuristických algoritmů. V tomto případě se model dokonce dostal na 1730 ticků.



Obrázek č. 3.7: Nastavení modelu s výslednými parametry

3.4.3. VÝSLEDKY EXPERIMENTU ALTRUISM

Výsledky z experimentu Altruism je možné vidět v tabulce č. 3.3. V tabulce je vidět časová náročnost a kvalita fitness funkce jednotlivých algoritmů. Pro fitness funkci bude opět využito funkce „BestChecked replicates“.

Vysvětlivky k tabulce č. 3.3 a obrázku č. 3.8.

Algoritmy:

- RS – random Search (algoritmus náhodného vyhledávání)
- SA – simulated Annealing (simulované žíhání)
- MHC – multiple hill climbing (horolezecký algoritmus)

- GA – genetic algorithm (standardní genetický algoritmus)

Hlavička tabulky:

- AP – altruistic probability (počáteční pravděpodobnost altruistů)
- SP – selfish probability (počáteční pravděpodobnost egoistů)
- COA – cost of Altruism (náklady na Altruismus)
- BFA – benefit from Altruism (užitek z Altruismu)
- D – disease (nemoc modelu, pravděpodobnost prázdného pole)
- H – harshness (ostrost modelu, fitness pro prázdná pole)

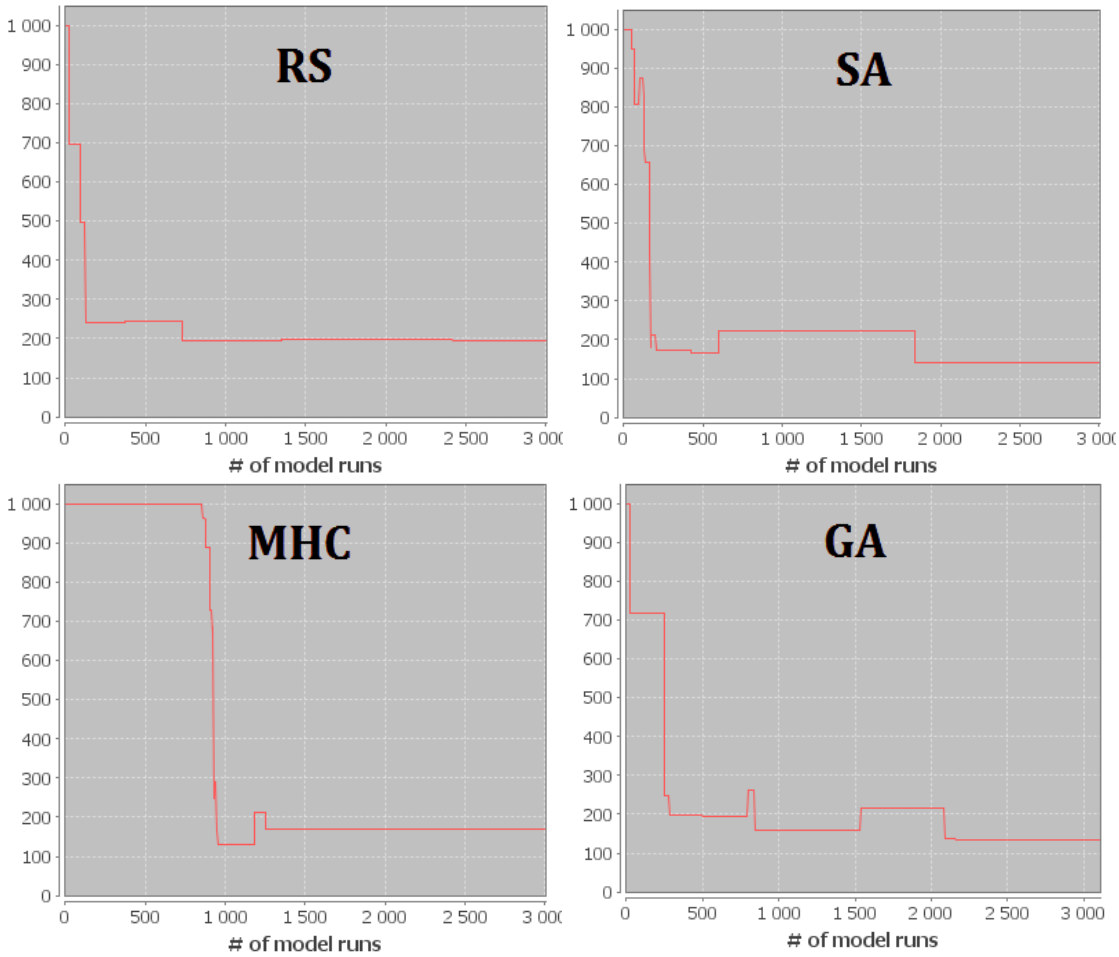
Algoritmus	Čas	Fitness	AP	SP	COA	BFA	D	H
RS	21:05	195,41	0,45	0,40	0,17	0,87	0,15	0,44
SA	23:15	140,20	0,25	0,31	0,07	0,39	0,04	0,22
MHC	23:12	170,06	0,15	0,08	0,17	0,71	0,41	0,08
GA	20:59	133,06	0,18	0,44	0,15	0,85	0,19	0,36

Tabulka č. 3.3: Porovnání algoritmů po 3000 krocích

Z tabulky č. 3.3 je možné vypozorovat velmi překvapivou nejnižší časovou náročnost u genetického algoritmu, který se vykonal o 6 vteřin rychleji nežli náhodný algoritmus. Možným důvodem tohoto chování může být obrovský prohledávací prostor, který zatěžuje náhodný algoritmus. Stejně jako u předchozího experimentu simulované žíhání a horolezecký algoritmus mají největší časovou náročnost, v tomto případě jsou na tom téměř stejně. Je to opět zapříčiněno častějším nalezením nové nejlepší funkce (parametr „BestChecking replicates“) a musí se tak spustit vícekrát, nežli ostatní dva algoritmy.

Na obrázku č. 3.8 je možné vidět vývoj jednotlivých algoritmů. Opět je použita funkce „BestChecking replicates“ a tedy grafy vývoje jednotlivých algoritmů můžou i klesat. Všechny algoritmy začínají svůj vývoj na fitness funkci 1000. Je to hodnota, která je fixní pro mnoho nastavení, protože to znamená, že jedna z populací je na nule. Tento fakt velmi zabrzdil horolezecký algoritmus. Kdyby nebyl nastaven parametr „restart-after-stall-count“ na 200, tak by se algoritmus ani nerozeběhl a jeho fitness funkce by byla 1000. Na tomto příkladu je vidět, kdy horolezecký algoritmus ohrožuje výslednou fitness funkci. Algoritmus se

rozběhl až přibližně na 800 kroku a pak následoval rychlý nález fitness funkce stejně jako u simulovaného žíhání. Genetický algoritmus má klasický rozvoj pro fitness funkci.

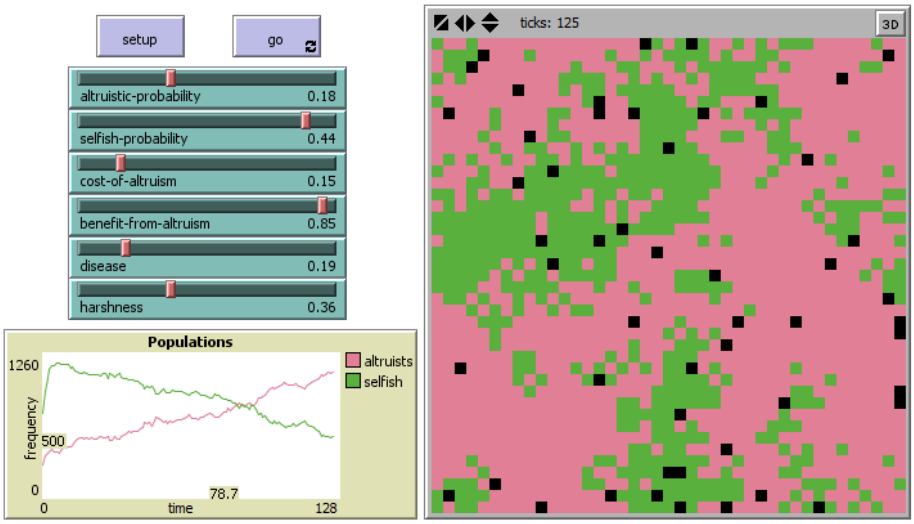


Obrázek č. 3.8: Vývoj algoritmů Altruism z programu BehaviorSearch

V tomto experimentu se hledala minimální fitness funkce. Tu nejlépe opět nalezl genetický algoritmus, kde 20 spuštění modelu ukázalo průměrnou hodnotu měření 133,06. Následované simulovaným žíháním se 140,20. Poznamenaný horolezecký algoritmus nalezl jen hodnotu 170,06 a klasicky nejhůře dopadl náhodný algoritmus, který vyhledal parametry, co stačili na fitness funkci 195,41. Vyhledané parametry se u všech algoritmů velmi měnili. Jediné podobné parametry jsou logicky u nákladů na Altruismus. Které by měli být nízké, aby nebyl velký rozdíl oproti egoistům. Nicméně je překvapivé, že tento parametr, žádný algoritmus nevynuloval, ale hodnoty se pohybují od pravděpodobnosti 0,07 do 0,17. Celkově byl předpoklad, že počáteční rozložení populace bude stejné jak u

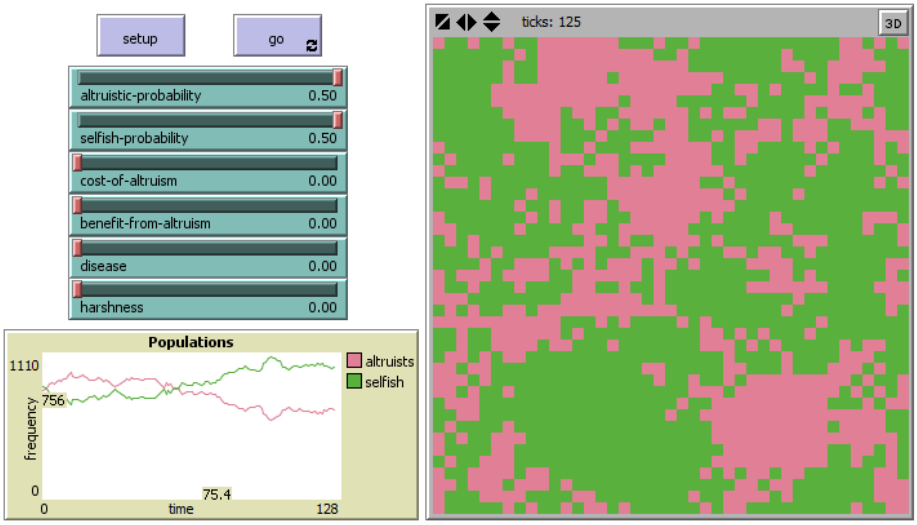
altruistů, tak u egoistů. Oproti tomu genetický algoritmus dokonce doporučuje počáteční nastavení pravděpodobnosti altruistů na 0,18 a pravděpodobnost egoistů na 0,44. Ostatní algoritmy mají počáteční pravděpodobnost agentů podobnou. Užitek z Altruismu se pohybuje mezi 0,39 – 0,87. Tento parametr logicky vyvažuje drsnost a nemocnost systému. Pokud je užitek z Altruismu nízký, musí být i drsnost a nemocnost nízká. Kdežto je-li užitek vyšší, ovlivňuje to i vyšší nastavení drsnosti a nemocnosti systému.

Oproti hypotéze, že bude počáteční populace vyrovnaná a ostatní parametry budou nulové, tak algoritmy našli velmi širokou škálu nastavení. Toto nastavení je pro vyrovnaný vývoj mezi 75 a 125 krokem modelu mnohem vhodnější. Na obrázku č. 3.9 je použito nastavení z genetického algoritmu. Tento běh modelu měl průměrnou směrodatnou odchylku zvýšenou o pětinu prázdných polí 136.



Obrázek č. 3.9: Nastavení parametrů podle genetického algoritmu

Na obrázku č. 3.10 je vidět nastavení parametrů podle hypotézy. Běh modelu podle hypotézy, měl průměrnou směrodatnou odchylku zvýšenou o pětinu prázdných polí 160. Pokud se nastaví vše na stejné hodnoty, model má pak někdy tendenci jednu populaci po 75 krocích velmi zvýhodňovat. I přes fakt, že při tomto nastavení není žádná nemoc ani drsnost, tak nastavení genetického algoritmu je v průměru lepší nežli původní hypotézy. Na tomto příkladu je vidět síla heuristických algoritmů a zvláště pak genetického algoritmu.



Obrázek č. 3.10: Nastavení parametrů podle hypotézy

4. ZÁVĚR

Kromě výsledků chování tří rozdílných modelů (Muscle Development, Wolf Sheep Predation a Altruism) lze formulovat některé obecnější závěry. Zaprvé, evoluční algoritmy, jako například genetický algoritmus, simulované žíhání a horolezecký algoritmus, jsou opravdu efektivním nástrojem pro zkoumání parametrového prostoru agentových modelů. Nalezení fitness funkce téměř vždy převyšovalo náhodný algoritmus. Zadruhé, díky opakovanému spuštění stochastického modelu a shromáždění dat parametrového nastavení těchto běhů můžeme odhalit předem nečekané chování modelu.

Na třech rozdílných zkoumaných modelech bylo demonstrováno, jaký typ algoritmu se hodí na daný problém. Použití náhodného algoritmu je vhodné jen u velmi malého parametrového prostoru, z důvodu nízké časové náročnosti. Horolezecký algoritmus a simulované žíhání jsou algoritmy rychle se rozvíjející, když je model náročný na čas a není možné kvůli časové náročnosti spustit mnohokrát model, pak přicházejí na řadu tyto evoluční algoritmy. Problémem horolezeckého algoritmu je možnost uvíznutí v lokálním maximu nebo minimu. Pokud model ukazuje stejný výsledek fitness funkce pro mnohé nastavení parametrů, je to velké ohrožení pro horolezecký algoritmus, kterému se pak musí nastavit nové počáteční parametry, aby mohl pokračovat v běhu. Tento problém se vyskytl u experimentu v modelu Altruism. Pro tyto účely by mělo přijít na řadu simulované žíhání, které si s tímto problémem dokáže poradit. Ovšem nastavení algoritmu simulovaného žíhání je složitější proces. A je na uživateli, který z těchto algoritmů podle svého uvážení zvolí.

Poslední možností programu BehaviorSearch je genetický algoritmus. Tento algoritmus dosahuje nejlepších výsledků v prostředí, kde je obrovský rozsah možných parametrů a opakovaného spouštění modelu. U experimentu s modelem Altruism dokonce předčil i v časové náročnosti náhodný algoritmus s mnohem lepší fitness funkcí. U tohoto algoritmu je největší komplikací volba počátečního nastavení. V práci bylo použito nastavení genetického algoritmu podle Hollandovy knihy. Zajímavé by bylo sledovat s jakým nastavením křížením (crossover-rate), mutací (mutation-rate), velikostí populace (population-size), nastavením výměny populace (population-model) a velikostí turnajové selekce (tournament-size), se

změní běh algoritmu. Tento výzkum se pokouší o rozkrytí dané problematiky z poměrně opomíjeného úhlu pohledu a nabízí tak možnost dalšího zkoumání této látky.

BehaviorSearch lze považovat za velmi silný nástroj pro účely zkoumání parametrového prostoru stochastických modelů za použití evolučních algoritmů. Je ovšem omezen pouze na agentové modely z NetLoga. Dalším problémem BehaviorSearch je volba funkce „BesChecking replicates“ pro překontrolování fitness funkce. Graf pak zobrazuje pouze „rechecked“ fitness funkci. Ovšem algoritmus musí najít lepší nastavení parametrů původní fitness funkce a nikoliv té „rechecked“. Bylo by vhodnější, pokud by při nalezení nové fitness funkce, která je lepší než „rechecked“, znova překontroloval fitness funkci. Oproti tomu, tato skutečnost, by mohla algoritmus výrazně zpomalit.

Obecně evoluční algoritmy mají pro experimentální účely skvělou budoucnost. A měly by být využívány pro vědecké účely čím dál tím více. Lze tak objevit zajímavé chování stochastických modelů.

5. LITERÁRNÍ A INTERNETOVÉ ZDROJE

[1] BRYSON, J. J., Y. ANDO a H. LEHMANN. Agent-based modelling as scientific method: a case study analysing primate social behaviour. Philosophical Transactions of the Royal Society B: Biological Sciences. 2007, 362(1485): 1685-1699. DOI: 10.1098/rstb.2007.2061. ISSN 0962-8436. Dostupné také z: http://rstb.royalsocietypublishing.org/cgi/doi/10.1098/rstb.2007.2061

[2] MACAL, CH. M. – NORTH, M. J. *Tutorial on agent-based modelling and simulation.* [online]. Center for Complex Adaptive Agent Systems Simulation. Argonne National Laboratory. USA., 2005 [cit. 2015-05-19]. Dostupné z: http://www.informs-sim.org/wsc05papers/002.pdf

[3] ROCHA, Luis M. Complex Systems Modeling: Using Metaphors From Nature in Simulation and Scientific Models. *Complex Systems Modeling* [online]. 2003 [cit. 2015-06-01]. Dostupné z: http://www.informatics.indiana.edu/rocha/complex/csm.html

[4] EPSTEIN, Joshua M a Robert AXTELL. *Growing artificial societies: social science from the bottom up*[online]. Washington, D.C.: Brookings Institution Press, c1996, xv, 208 p. [cit. 2015-06-01]. ISBN 02-625-5026-1.

[5] JENNINGS, Nicholas R. On agent-based software engineering. *Artificial Intelligence* [online]. 2000, **117**(2): 277-296 [cit. 2015-06-01]. DOI: 10.1016/S0004-3702(99)00107-1. ISSN 00043702. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0004370299001071

[6] BONABEAU, E. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences* [online]. 2002, **99**(Supplement 3): 7280-7287 [cit. 2015-06-3]. DOI: 10.1073/pnas.082080899. ISSN 0027-8424. Dostupné z: http://www.pnas.org/cgi/doi/10.1073/pnas.082080899

[7] GARDNER, Martin. Mathematical Games. *Scientific American* [online]. 1970, **223**(4): 120-123 [cit. 2015-06-03]. DOI: 10.1038/scientificamerican1070-120. ISSN 0036-8733. Dostupné z: http://www.nature.com/doifinder/10.1038/scientificamerican1070-120

[8] MACAL, C M a M J NORTH. Tutorial on agent-based modelling and simulation. *Journal of Simulation* [online]. 2010, **4**(3): 151-162 [cit. 2015-06-03]. DOI: 10.1057/jos.2010.3. ISSN 1747-7778. Dostupné z: http://www.palgrave-journals.com/doifinder/10.1057/jos.2010.3

[9] DELOACH, Scott a Jorge VALENZUELA. *An Agent-Environment Interaction Model* [online]. 2007, (4405) [cit. 2015-06-07]. Dostupné z: http://people.cis.ksu.edu/~jvalenzu/publications/AOSE2006.pdf

[10] NIAZI, Muaz a Amir HUSSAIN. Agent-based computing from multi-agent systems to agent-based models: a visual survey. *Scientometrics* [online]. 2011, **89**(2): 479-499 [cit. 2015-06-07]. DOI: 10.1007/s11192-011-0468-9. ISSN 0138-9130. Dostupné z: http://link.springer.com/10.1007/s11192-011-0468-9

[11] NetLogo Wave Machine model. WILENSKY, Uri. *NetLogo Models Library* [online]. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997 [cit. 2015-06-07]. Dostupné z: http://ccl.northwestern.edu/netlogo/models/WaveMachine

[12] NetLogo AIDS model. WILENSKY, Uri. *NetLogo Models Library* [online]. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997 [cit. 2015-06-22]. Dostupné z: http://ccl.northwestern.edu/netlogo/models/AIDS

[13] NetLogo Simple Birth Rates model. WILENSKY, Uri. *NetLogo Models Library* [online]. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997 [cit. 2015-06-22]. Dostupné z: http://ccl.northwestern.edu/netlogo/models/SimpleBirthRates

[14] RUSSELL, Stuart J a Peter NORVIG. Artificial intelligence: a modern approach. 3rd ed. Harlow: Pearson Education, c2014, ii, 1091 s. ISBN 978-1-29202-420-2. HOLLAND, John H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. 1st MIT Press ed. Cambridge, Mass.: MIT Press, 1992, xiv, 211 p. ISBN 02-625-8111-6.

[15] HYNEK, Josef. *Genetické algoritmy a genetické programování*. 1. vyd. Praha: Grada, 2008, 182 s. ISBN 978-80-247-2695-3.

[16] MITCHELL, Melanie. *An introduction to genetic algorithms*. Cambridge, Mass: MIT Press, 1996. ISBN 978-058-5030-944.

[17] SRINIVAS, M., L.M. PATNAIK a Carolina SALTO. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics* [online]. XVIII Congreso Argentino de Ciencias de la Computación., 2012, **24**(4): 656-667 [cit. 2015-07-19]. DOI: 10.1109/21.286385. ISSN 00189472. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=286385

[18] STARK, Natalia, Gabriela F. MINETTI a Carolina SALTO. *A new strategy for adapting the mutation probability in genetic algorithms.* [online]. XVIII Congreso Argentino de Ciencias de la Computación., 2012 [cit. 2015-08-01]. Dostupné z: http://sedici.unlp.edu.ar/handle/10915/23593

[19] WILENSKY, Uri. NetLogo User Manual. *NetLogo* [online]. 2015 [cit. 2015-08-01]. Dostupné z: http://ccl.northwestern.edu/netlogo/docs/

[20] NetLogo Wolf Sheep Predation model. WILENSKY, Uri. *NetLogo Models Library* [online]. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1997 [cit. 2015-08-111]. Dostupné z: http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation

[21] STONEDHAL, Forrest a Uri WILENSKI. *BehaviorSearch.org* [online]. 2013 [cit. 2015-08-11]. Dostupné z: http://www.behaviorsearch.org/

[22] NetLogo Muscle Development model. WILENSKY, Uri. *NetLogo Models Library* [online]. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2002 [cit. 2015-10-27]. Dostupné z: http://ccl.northwestern.edu/netlogo/models/MuscleDevelopment

[23] ARCHIBALD, Dresdin. The Importance of Sleep for Weightlifters and Other Athletes. Breaking muscle [online]. 2015 [cit. 2015-10-30]. Dostupné z: http://breakingmuscle.com/olympic-weightlifting/the-importance-of-sleep-for-weightlifters-and-other-athletes

[24] MAH, Cheri D., Kenneth E. MAH, Eric J. KEZIRIAN a William C. DEMENT. The Effects of Sleep Extension on the Athletic Performance of Collegiate Basketball Players. SLEEP [online]. 2011 [cit. 2015-10-30]. DOI: 10.5665/sleep.1132. ISSN 0161-8105. Dostupné z: http://www.journalsleep.org/ViewAbstract.aspx?pid=28194

[25] NetLogo Altruism model. WILENSKY, Uri. *NetLogo Models Library* [online]. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1998 [cit. 2015-11-04]. Dostupné z: http://ccl.northwestern.edu/netlogo/models/Altruism

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Údaje o DIPLOMOVÉ PRÁCI studenta

Os. číslo:	I1201761	**Datum zadání:**	15.10.2013
Příjmení a jméno:	Zajíček Roman	**Plánované datum odevzdání:**	15.10.2014
Obor/komb.:	Aplikovaná informatika (ai2-p)	**Datum odevzdání:**	dosud neodevzdáno
Zadané téma:	Optimalizace agentových modelů pomocí evolučních algoritmů		
Stav práce:	Rozpracovaná práce		

Údaje o kvalifikační práci

1. Hlavní téma:
Optimalizace agentových modelů pomocí evolučních algoritmů

2. Hlavní téma v angličtině:
Optimization of agent-based models using evolutionary algorithms

3. Název dle studenta:
Optimalizace agentových modelů pomocí evolučních algoritmů

4. Název dle studenta v angličtině:
Optimization of agent-based models using evolutionary algorithms

5. Souběžný název:

6. Podnázev:

7. Anotace (krátký popis práce):
Cílem práce je popsat uplatnitelnost genetických algoritmů při kalibraci a optimalizaci agentových modelů.

8. Klíčová slova (oddělujte čárkou):
ISIT - Šk.rok 2012/3, číslo 71

9. Anotace v angličtině (krátký popis práce):

10. Anglická klíčová slova (oddělujte čárkou):

11. Přílohy volně vložené:

12. Přílohy vázané v práci:

13. Rozsah práce:

14. Jazyk práce:

15. Záznam průběhu obhajoby:

16. Zásady pro vypracování:
Anotace:
Diplomová práce se zabývá základním teoretickým popisem agentových modelů a genetických algoritmů. V teoretické části je dále nastíněn chod primitivního genetického algoritmu s ruletovou selekcí. Následuje základní popis NetLoga a navazujícího nástroje BehaviorSearch.
V praktické části nejprve probíhá výběr agentových modelů pro optimalizaci. Vybranými modely jsou: Muscle Development, Wolf Sheep Predation a Altruism. Ke každému vybranému modelu je navržena hypotéza a jsou provedeny experimenty v nástroji BehaviorSearch. Závěrem se hodnotí výsledky, z pohledu porovnání všech algoritmů nástroje BehaviorSearch.

Osnova:
1. Úvod
2. Teoretická část
2.1. Agentové modely
2.2. Genetické algoritmy
2.3. NetLogo a BehaviorSearch
3. Praktická část
3.1. Výběr modelů pro optimalizaci
3.2. Hypotézy

Údaje o DIPLOMOVÉ PRÁCI studenta

Os. číslo:	I1201761	**Datum zadání:**	15.10.2013
Příjmení a jméno:	Zajíček Roman	**Plánované datum odevzdání:**	15.10.2014
Obor/komb.:	Aplikovaná informatika (ai2-p)	**Datum odevzdání:**	dosud neodevzdáno
Zadané téma:	Optimalizace agentových modelů pomocí evolučních algoritmů		
Stav práce:	Rozpracovaná práce		

3.3. Experimenty
3.4. Výsledky
4. Závěr
5. Literární a internetové zdroje

17. Seznam doporučené literatury:

1. BRYSON, J. J., Y. ANDO a H. LEHMANN. Agent-based modelling as scientific method: a case study analysing primate social behaviour. Philosophical Transactions of the Royal Society.
2. MACAL, CH. M. NORTH, M. J. Tutorial on agent-based modelling and simulation. Center for Complex Adaptive Agent Systems Simulation. Argonne National Laboratory.
3. ROCHA, Luis M. Complex Systems Modeling: Using Metaphors From Nature in Simulation and Scientific Models. Complex Systems Modeling.
4. MACAL, C M a M J NORTH. Tutorial on agent-based modelling and simulation. Journal of Simulation.
5. HYNEK, Josef. Genetické algoritmy a genetické programování. 1. vyd.

18. Osoby VŠKP:

Vedoucí: Štekerová Kamila, doc. RNDr. Ph.D.

Oponent: Tesařová Barbora, Ing. Ph.D.

Elektronická forma kvalifikační práce

Zatím není přiložen žádný soubor s elektronickou formou práce...

Posudky kvalifikační práce

Posudek(y) oponenta	Není k dispozici...
Hodnocení vedoucího	Není k dispozici...

Soubor s průběhem obhajoby	Žádný není vložen