Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Genetické algoritmy v multiagentním systému

2024

Filip Rybníček

Vedoucí práce:
Mgr. Petr Osička, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Filip Rybníček
Název práce:	Genetické algoritmy v multiagentním systému
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	Mgr. Petr Osička, Ph.D.
Počet stran:	33
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Filip Rybníček
Title:	Genetic algorithms in multiagent systems
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	Mgr. Petr Osička, Ph.D.
Page count:	33
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Tato bakalářská práce se zabývá integrací genetických algoritmů do multiagentních systémů s cílem prozkoumat jejich potenciál v simulaci a optimalizaci složitých problémů. Práce kombinuje teoretický přístup s praktickou implementací, kde jsou genetické algoritmy aplikovány na modelování a evoluci autonomních agentů v dynamickém prostředí. Skrze analýzu a experimenty bylo prokázáno, že spojení těchto dvou technologií umožňuje řešení problémů, jako je optimalizace chování agentů a adaptace na proměnlivé podmínky prostředí. Práce tak přispívá k lepšímu porozumění dynamice genetických algoritmů a multiagentních systémů a jejich aplikaci na řešení reálných problémů.

Synopsis

This bachelor's thesis focuses on integrating genetic algorithms into multi-agent systems to explore their potential in simulating and optimizing complex problems. The work combines a theoretical approach with practical implementation, where genetic algorithms are applied to model and evolve autonomous agents in a dynamic environment. Through analysis and experiments, it was demonstrated that the integration of these two technologies enables the solving of issues such as optimizing agent behaviors and adapting to changing environmental conditions. Thus, the thesis contributes to a better understanding of the dynamics of genetic algorithms and multi-agent systems and their application to solving real-world problems.

Klíčová slova: Genetické algoritmy, multiagentní systémy, optimalizace, evoluce agentů

Keywords: Genetic algorithms, multi-agent systems, optimization, agent evolution

Chtěl bych poděkovat mému vedoucímu práce Mgr. Petru Osičkovi, Ph.D. za jeho nápady a pomoc na projektu.

Odevzdáním tohoto textu jeho autor místopřísežně prohlašuje, že celou práci včetně příloh vypracoval samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

Seznam obrázků

1 Úvod

V současné době, kdy se technologie dynamicky vyvíjí a požadavky na řešení složitých problémů neustále rostou, čelíme výzvám, kde tradiční algoritmické přístupy často selhávají v efektivitě nebo proveditelnosti. Optimalizační metody nabízejí nástroje pro hledání přijatelných, a často i překvapivě efektivních řešení i pro komplexní problémy.

Jednou z těchto metod jsou genetické algoritmy, které v sobě zahrnují evoluční principy přírody, jako jsou selekce, mutace a křížení, a mají schopnost efektivně prozkoumávat obrovské prostory možných řešení a reagovat na dynamicky se měnící podmínky.

Multiagentní systémy představují rámec pro modelování složitých interakcí mezi autonomními agenty, kteří soutěží nebo koordinují své akce za účelem dosažení společných nebo individuálních cílů. Tento přístup umožňuje simulovat různé sociální, ekonomické a technologické systémy.

Tato bakalářská práce se zaměřuje na spojení genetických algoritmů a multiagentních systémů. Hlavním cílem je prozkoumat, jak lze tyto dva přístupy spojit pro řešení složitých optimalizačních problémů, a především jak mohou genetické algoritmy zvýšit schopnosti a efektivitu agentů v multiagentním prostředí.

Práce je strukturována do několika částí. Po úvodu, následuje teoretická část, jež se podrobně věnuje principům, historii, vývoji obou technologií, a ukázkám projektů kde byly využity. Praktická část práce je zaměřena na design, implementaci a evaluaci specifického multiagentního systému, demonstrující praktické aplikace a možnosti genetických algoritmů v rámci tohoto systému. Závěrem jsou shrnuty hlavní výsledky a diskutována možná budoucí rozšíření této práce.

2 Genetické algoritmy

2.1 Historie a vývoj genetických algoritmů

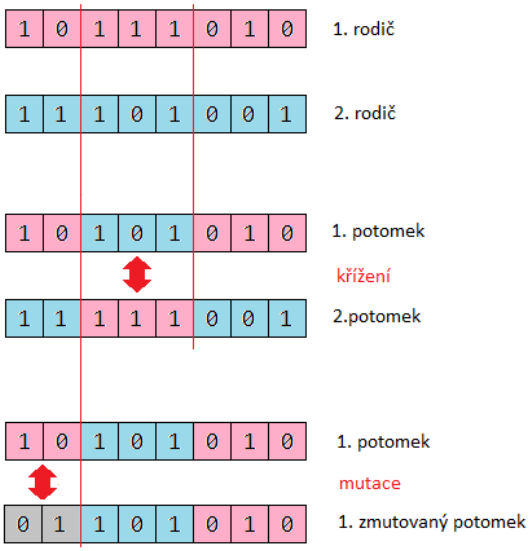
Genetické algoritmy (GA) byly poprvé formalizovány a rozvinuty v 60. letech 20. století Johnem Hollandem a jeho studenty na University of Michigan. Hollandova kniha *Adaptation in Natural and Artificial Systems* z roku 1975 položila teoretické základy pro GA a představila koncept optimalizačního algoritmu, který by využíval principy biologické evoluce.

Na přelomu 80. a 90. let 20. století byla metoda genetických algoritmů použita jako výchozí bod Johnem Kuzou pro vývoj genetického programování [1]. Tato metoda rozšiřuje možnosti GA tím, že místo evoluce objektů dochází k evoluci celého programu za účelem najít optimální řešení.

V posledních letech došlo k vývoji hybridních systémů, které kombinují GA s jinými optimalizačními metodami pro zvýšení výkonu a efektivity. GA se ukázaly být účinné v komplexních úlohách, jako je návrh distribuovaných systémů, optimalizace energetických sítí a vývoj pokročilých robotických systémů.

2.2 Základní principy a pojmy genetických algoritmů

Princip GA [2] je vytvářet na sebe navazující řešení, kde každá další generace je blíže k nejlepšímu řešení. Myšlenka GA je převzata z biologie, kde pomocí selekce jsou ti nejsilnější jedinci vybráni k reprodukci, aby se jejich geny přenášely do dalších generací. Mezi základní pojmy patří:



Obrázek 1: Křížení a mutace (Autor: Miloš Křivan)

Chromozom je základní prvek GA a jedná se o abstraktní reprezentaci možného řešení. Podobně jako v přírodní biologii, kde DNA v chromozomu popisuje všechny vlastnosti a funkce organismu, chromozomy v GA symbolizuje soubor informací, které definují, jak je řešení problému strukturováno.

Generace je soubor chromozomů, které jsou aktuálně zpracovávány. První generace je vygenerovaná náhodně, a pak každá následují se snaží přiblížit k nejlepšímu řešení.

Fitness funkce přiřazuje každému chromozomu ohodnocení, které reprezentuje jak dobré řešení daný chromozom reprezentuje. GA se tedy snaží maximalizovat (nebo minimalizovat) hodnotu fitness funkce, tak aby našel nejlepší řešení problému.

Selekce je proces, při kterém jsou z generace vybíráni nejlepší jedinci pro reprodukci. Existují různé metody selekce, jako je například ruletová selekce, turnajová selekce nebo selekce na základě hodnosti, které zajišťují, že jedinci s vyšší fitness mají větší šanci na reprodukci.

Křížení je proces, při kterém jsou kombinovány geny dvou rodičů za účelem vytvoření jednoho nebo více potomků. Tento proces umožňuje vznik nových jedinců, kteří dědí charakteristiky od obou rodičů, což může vést k nalezení lepších řešení problému.

Mutace je náhodná změna v genech jedince, která zajišťuje genetickou diverzitu v populaci. Mutace může představovat náhodnou změnu hodnoty genu, což pomáhá algoritmu prozkoumat nové oblasti prostoru řešení a zabránit předčasné konvergenci na suboptimální řešení.

2.3 Průběh genetického algoritmu

Klíčem k úspěchu GA je efektivní simulace evolučních principů, které umožňují algoritmu průběžně vylepšovat kandidátská řešení až do dosažení optimálního nebo dostatečně vhodného řešení daného problému. Tento postup se skládá z následujících kroků:

1. Inicializace generace. Prvním krokem je vytvoření počáteční generace potenciálních řešení. Prvotní generace by měla obsahovat různorodá řešení a pokrývat celou oblast ve které se snažíme najít nejlepší řešení. Pokud by oblast nebyla dostatečně široká, existuje vyšší šance, že GA uvízne v lokálním maximu (minimu) a poskytne nám pouze suboptimální řešení. Velikost generace je obvykle pevně stanovena a je důležitým parametrem, který ovlivňuje výkonnost a efektivnost algoritmu, větší generace však mohou také klást vysoké náklady na simulaci.

2. Hodnocení generace. Po vytvoření generace dojde k ohodnocení každého chromozomu pomocí fitness funkce, aby jsme zjistili které z nich jsou užitečné, a které neposkytují dobrá řešení. Tato fáze je velmi závislá na charakteru fitness funkce, ale většinou bývá výpočetně nejnáročnější.

3. Selekce a křížení. Pro tvorbu nové generace vybereme nejdříve pomocí selekce nejúspěšnější chromozomy k reprodukci. Mezi těmito chromozomy je pak provedeno křížení, pomocí kterého získáme chromozomy s potenciálně dobrými vlastnostmi.

4. Mutace. Před dokončením nové generace může s jistou malou pravděpodobností dojít k mutaci, která náhodně změní jeden nebo více genů v chromozomu. Tímto způsobem je dosaženo genetické diverzity generace a zabrání se předčasné konvergenci k suboptimálním řešením. Pravděpodobnost mutace je ale obvykle nastavena nízko, aby se neporušila již existující adaptace.

5. Nahrazení. Po vytvoření nové generace dochází k nahrazení části nebo celé stávající generace novými chromozomy. Strategie nahrazení se liší - některé algoritmy uchovávají nejlepší chromozomy přes generace (elitismus), zatímco jiné kompletně nahrazují starou populaci nově vytvořenými chromozomy.

6. Iterace. Proces se opakuje od 2. bodu hodnocení generace, dokud není splněno jedno z ukončovacích kritérií, jako je dosažení maximálního počtu generací, stagnace fitness hodnoty populace, nebo nalezení řešení, které splňuje předem stanovená kritéria kvality.

2.4 Využití genetických algoritmů

Optimalizace Genetické algoritmy jsou široce využívány pro řešení optimalizačních úloh v různých oblastech, včetně inženýrství, logistiky, finančnictví a dalších. Tyto algoritmy excelují v hledání nejlepších možných řešení v rámci daných omezení, zejména v situacích, kde je prostor řešení příliš velký nebo příliš složitý pro tradiční optimalizační metody.

Umělá inteligence a strojové učení V oblasti umělé inteligence (AI) a strojového učení mohou GA zlepšovat a optimalizovat parametry učících algoritmů, struktury a návrhy neuronových sítí, což vede k výkonnějším modelům. GA mohou pomoci identifikovat nejefektivnější konfigurace algoritmů a usnadňují automatizovaný návrh inovativních architektur neuronových sítí.

NEAT [3] je inovativní algoritmus pro evoluční vývoj neuronových sítí, který postupně zvyšuje složitost sítí během evoluce. Tento přístup se ukázal jako velmi účinný v několika úlohách strojového učení a AI, včetně videoher a řízení autonomních robotů.

Evoluční design V architektuře, průmyslovém a produktovém designu mohou genetické algoritmy identifikovat optimální návrhy, které splňují předem definovaná kritéria výkonu a estetiky, a zároveň respektují omezení materiálu a nákladů. GA poskytují nástroj pro experimentování s velkým množstvím variant návrhu a umožňují návrhářům objevovat inovativní řešení.

NASA [4] využila GA pro návrh antény pro mise Space Technology 5 (ST5). GA optimalizoval tvar a rozmístění antén na satelitech (obrázek 2), což vedlo k vytvoření vysoce efektivního a nekonvenčního designu. Tento případ dokazuje schopnost GA objevovat inovativní řešení, která jsou lidským návrhářům skryta.

Hudba a umění Genetické algoritmy nacházejí uplatnění i v oblasti hudby a umění, kde mohou pomáhat vytvářet nové hudební skladby, vizuální umění nebo dokonce inovativní designy prostřednictvím experimentování s různými kombinacemi zvuků, barev a tvarů. Umožňují umělcům a designérům objevovat nové estetické formy a experimentální koncepty

Projekt "DarwinTunes"[5] využíval GA k evoluci hudebních skladeb z jednoduchých zvukových bloků, kde uživatelé hlasováním o svých preferencích řídili evoluční proces, což vedlo k vytvoření nových a originálních skladeb.

Hry V herním průmyslu a simulacích genetické algoritmy umožňují vývoj složitých herních strategií a simulaci evolučních procesů v digitálních ekosystémech.

Obrázek 2: Design ST5 antény

GA mohou být použity k automatickému generování úrovní, k vytváření adaptabilních a inteligentních nepřátelských AI, nebo pro modelování evolučních dynamik v simulovaných prostředích, což vede k realističtějším a poutavějším herním zážitkům.

I když přímo nevyužívá genetické algoritmy v čisté formě, projekt AlphaGo od Google DeepMind [6] ukazuje, jak principy evolučních algoritmů a strojového učení mohou spolupracovat k dosažení významných průlomů, jako je porážka světových šampionů ve hře Go, což bylo považováno za jednu z největších výzev v AI.

3 Multiagentní systémy

3.1 Historie a vývoj multiagentních systémů

První nápady a teoretické koncepty multiagentních systémů (MAS) [7] vznikly v rámci širšího pole distribuované umělé inteligence (DAI). Počátkem 70. let se představili myšlenky, které měly velký vliv na formování základů MAS, včetně konceptu agentů jako autonomních počítačových programů schopných vzájemné komunikace a vzájemného působení.

V 80. letech došlo k dalšímu rozvoji a specifikaci konceptů MAS, zejména modelování sociálních interakcí a rozhodovacích procesů v systémech obsahujících mnoho agentů. Vědci jako Michael Wooldridge a Nicholas Jennings začali formulovat definice a vlastnosti, které by umožnily efektivní spolupráci a koordinaci mezi agenty, čímž položili základy pro dnešní komplexní multiagentní systémy.

Největší růst zaznamenali MAS v 90. letech, kdy došlo k uznání, že MAS mohou efektivně řešit složité a distribuované problémy, které jsou obtížně zpracovatelné pro monolitické systémy. Od té doby vědci a vývojáři začali aplikovat MAS v různých oblastech, od robotiky přes telekomunikace až po distribuované plánování a simulace.

3.2 Základní principy a pojmy multiagentních systémů

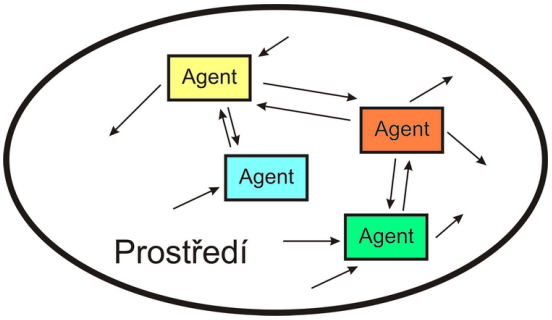
Princip MAS [8] spočívá v koordinaci autonomních agentů za účelem řešení úkolů, které jsou příliš komplexní nebo rozsáhlé pro jednotlivé agenty či tradiční výpočetní metody. Jednotliví agenti jsou postradatelní a nekonají žádnou velkou roli, ale skrze jejich interakce vznikají zajímavé emergentní vlastnosti celkového systému, pomocí kterých lze řešit mnohé jinak obtížné problémy. Základní vlastnosti MAS:

Autonomie: Každý agent je schopen nezávislého rozhodování na základě svého vnímání a vnitřního stavu.

Lokální vnímání: Agenti mají omezené informace o celém systému a musí jednat na základě lokálně dostupných dat.

Decentralizace: Neexistuje žádný centrální řídící mechanismus, rozhodování je distribuováno mezi agenty.

Interakce: Agenti komunikují a spolupracují s ostatními agenty prostřednictvím vzájemné výměny informací.



Obrázek 3: Princip multiagentních systémů (Autor: Martin Noska)

Základní pojmy:

Agent: Autonomní entita s vlastními cíli a schopnostmi. Agenti mohou být jednoduché programy s omezenou inteligencí nebo složité systémy s pokročilou schopností rozhodování a adaptace. Agenti v MAS mohou být homogenní (stejné typy s podobnými schopnostmi) nebo heterogenní (různé typy s různými schopnostmi).

Prostředí: Je to prostor, kde agenti interagují s ostatními agenty a se systémem. Může být fyzické nebo virtuální.

Interakce: Základním stavebním kamenem MAS. Agenti pomocí nich komunikují, vyjednávají, kooperují nebo soutěží s ostatními agenty k dosažení svých cílů.

Kooperace: Proces, při kterém dva nebo více agentů pracuje společně na dosažení společného cíle. Kooperativní chování je klíčové pro řešení složitých úkolů a dosahování synergických výsledků.

Kompetice: V některých systémech agenti soutěží o omezené zdroje nebo pro dosažení protichůdných cílů. Kompetice může vést k inovaci a optimalizaci strategií.

3.3 Průběh multiagentních systémů

MAS se zaměřuje na simulaci agentů a jejich reakce na dynamické prostředí. Tento postup se skládá z následujících kroků

1. Design a implementace agentů. Vytvoření agentů zahrnuje definování jejich cílů, schopností a pravidel pro interakci s prostředím a ostatními agenty. Agenti musí být schopni efektivně komunikovat a interagovat, což zahrnuje výměnu informací, vyjednávání a společné rozhodování. Systém musí zajistit mechanismy pro koordinaci akcí a kooperaci mezi agenty, aby bylo možné dosáhnout společných cílů a optimalizovat celkový výkon systému. Agenti v MAS by měli být schopni se adaptovat na změny v prostředí a učit se z interakcí, což zvyšuje jejich efektivitu a schopnost řešit nové problémy.

2. Percepce okolí. Agenti využívají senzory pro vnímání svého okolí, což jim umožňuje shromažďovat informace o aktuálním stavu systému a prostředí, ve kterém operují. Toto vnímání je základem pro veškeré následné rozhodování a interakce s ostatními agenty.

3. Rozhodování. Na základě získaných informací a vlastních cílů agenti samostatně rozhodují o svých dalších akcích. Tento proces zahrnuje vyhodnocování možných alternativ a výběr optimálního postupu k dosažení cílů s ohledem na současnou situaci.

4. Koordinace a spolupráce. Agenti komunikují a sdílejí informace s ostatními agenty, aby koordinovali své akce a spolupracovali na dosažení společných nebo komplementárních cílů. Efektivní koordinace a kooperace mezi agenty jsou zásadní pro řešení složitých úkolů, které přesahují možnosti jednotlivých agentů.

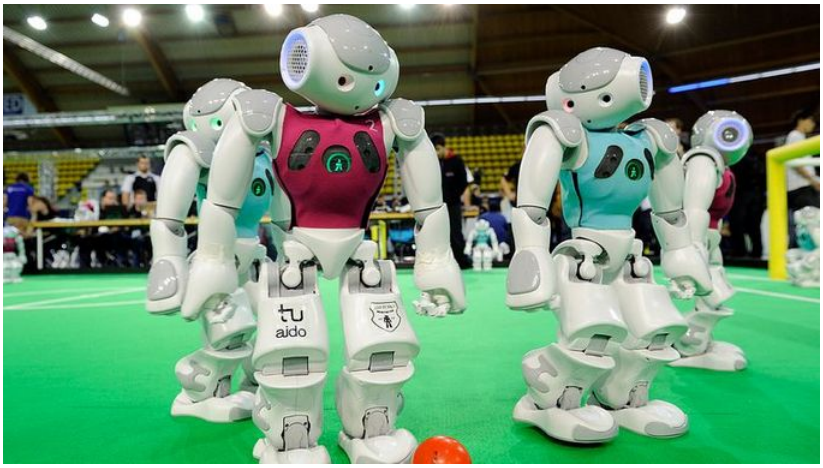
5. Realizace. Po rozhodnutí následuje realizace, během které agenti provádějí konkrétní akce v prostředí. Tyto akce mohou ovlivnit stav systému, prostředí nebo jiné agenty a jsou zásadní pro dosažení cílů agenta.

6. Iterace. Celý proces od percepce až po realizaci se iterativně opakuje, což umožňuje agentům dynamicky reagovat na změny v prostředí. Tato schopnost adaptace je klíčová pro udržení relevance a efektivity agentů v proměnlivých nebo nepředvídatelných situacích.

3.4 Využití multiagentních systémů

Robotika. V robotice se MAS využívají pro koordinaci skupin robotů při řešení úkolů, jako je průzkum terénu, záchranné mise, nebo automatizovaná výroba, kde každý robot funguje jako samostatný agent spolupracující na dosažení společného cíle.

RoboCup [9] je mezinárodní iniciativa, která poskytuje platformu pro výzkum a vývoj v oblasti robotiky a umělé inteligence, se zaměřením na fotbalové roboty. Tento projekt představuje významnou výzvu v oblasti kooperace a koordinace mezi robotickými agenty, kde týmy robotů musí demonstrovat strategické plánování, týmovou práci a adaptabilitu v dynamickém a nepředvídatelném prostředí.



Obrázek 4: Robocup 2014

Výpočetní věda. MAS hrají klíčovou roli ve výpočetní vědě, zejména při řešení složitých optimalizačních úloh, v simulacích a ve strojovém učení. Například v oblasti rozvrhování pomáhají efektivně alokovat zdroje, zatímco v zpracování obrazu umožňují paralelní zpracování informací pro rychlejší a přesnější analýzy.

OpenAI Five [10] je projekt společnosti OpenAI, který vyvinul tým pěti umělých inteligencí (AI agentů) schopných společně hrát komplexní strategickou hru Dota 2 proti lidským oponentům. Tento projekt demonstruje schopnosti MAS v oblasti strategického myšlení a týmové spolupráce.

Ekonomika a obchod. Multiagentní systémy umožňují simulaci ekonomických a obchodních modelů, kde agenti reprezentují jednotlivé účastníky trhu (například spotřebitele, firmy) a interagují na základě stanovených pravidel. Tyto

simulace pomáhají pochopit složité ekonomické jevy a testovat dopady různých regulací nebo tržních strategií.

Agent-based Computational Economics [11] je pole výzkumu, kde se MAS využívají k simulaci a analýze ekonomických a sociálních systémů, umožňující zkoumání tržních mechanismů, obchodních strategií a makroekonomických politik.

Řízení energetických sítí. V energetice umožňují MAS efektivní řízení distribuce a spotřeby energie. Agenti mohou reprezentovat výrobní zařízení, spotřebitele, nebo distribuční uzly a koordinovat své akce tak, aby se dosáhlo optimalizace celého systému z hlediska efektivity a spolehlivosti.

GridLAB-D [12] je platforma pro simulaci distribuce elektrické energie, která integruje modely spotřebitelů, distribučních sítí a výrobních zdrojů s využitím MAS pro optimalizaci a analýzu energetických systémů.

4 Integrace Genetických algoritmů do multiagentních systémů

Integrace GA do MAS představuje přístup k návrhu a optimalizaci složitých systémů. Tato kombinace technologií využívá silné stránky obou přístupů pro vytvoření účinějších systémů, které budou schopny se adaptovat na nečekané podmínky ve složitých prostředích.

4.1 Spojení Genetických algoritmů a multiagentních systémů

V první fázi integrace GA do MAS jsou genetické algoritmy používány pro evoluční vývoj strategií jednotlivých agentů. V tomto kontextu chromozomy v GA budou integrovány jako součást parametrů agentů, které určují chování agentů. Tyto mohou zahrnovat strategie pro interakci s prostředím, komunikační protokoly pro interakci s ostatními agenty, nebo rozhodovací logiku pro plnění úkolů.

GA umožňují agentům dynamicky se přizpůsobovat a učit z interakcí s prostředím a ostatními agenty. Průběhem selekce, křížení a mutace neustále vyvíjí nové generace agentů, které jsou lépe přizpůsobeny aktuálním podmínkám. Tato adaptace je zvláště užitečná v proměnlivých nebo nepředvídatelných prostředích, kde se mohou požadavky na chování agentů rychle měnit.

Integrací GA do MAS lze také optimalizovat strategie skupiny agentů tak, aby efektivně spolupracovali na dosažení společného cíle. GA mohou identifikovat nejúčinnější strategie pro alokaci zdrojů, koordinaci akcí mezi agenty, nebo pro navigaci v komplexním prostředí. To umožňuje MAS dosáhnout vyššího stupně efektivity a efektivnosti při řešení úkolů.

4.2 Výzvy integrace

Komplexita Designu. Integrace GA do MAS vyžaduje pečlivý návrh a ladění obou systémů, aby spolu efektivně fungovaly. Vytváření modelů, které správně reprezentují genetické operace v kontextu autonomních agentů, může být složité, zejména pokud má systém splňovat specifické požadavky nebo omezení.

Výpočetní Náročnost. Genetické algoritmy mohou být výpočetně náročné, zejména když pracují s velkými generacemi nebo v prostředích s vysokou úrovní detailů. Integrace GA do MAS zvyšuje tuto náročnost, protože vyžaduje neustálou aktualizaci a vyhodnocování strategií agentů. Toto může vést k potřebě výkonnějších hardwarových zdrojů nebo efektivnějších algoritmů.

Dynamika Prostředí. MAS obvykle operují v dynamických prostředích, kde se podmínky neustále mění. GA musí být schopné rychle reagovat na tyto změny a adaptovat strategie agentů v reálném čase. Tato rychlá adaptace může být výzvou, zvláště v prostředích, kde se mění klíčové parametry nebo cíle.

Koordinace a Kooperace. V multiagentních systémech je koordinace a kooperace mezi agenty zásadní. GA musí být schopné nejen optimalizovat individuální strategie agentů, ale také zajistit, že tyto strategie podporují koordinovanou a kooperativní práci. Vyvážení soutěživých a kooperativních aspektů v rámci MAS může být výzvou.

5 Srovnání s ostatními metodami

5.1 Alternativy ke genetickým algoritmům

Simulované žíhání (Simulated Annealing). Simulované žíhání [13] je optimalizační metoda, která se taktéž jako GA řadí mezi evoluční algoritmy. Tato metoda byla inspirovaná procesem žíhání v metalurgii, kde materiál je postupně ochlazován k dosažení nižšího energetického stavu a tím stabilnější struktury. Podobně jako GA se simulované žíhání snaží nalézt globální optimum v prostoru řešení. Simulované žíhání pracuje pouze s jedním řešením v daném čase, postupně snižuje "teplotu"systému a pravděpodobnost přijetí horších řešení. Tato metoda může být efektivnější v případech, kdy je vhodně nastaven zchladzovací plán, a umožňuje efektivní průzkum prostoru řešení s možností uniknout z lokálních optim.

Tabu vyhledávání (Tabu Search). Tabu vyhledávání [14] je metoda lokálního vyhledávání, která se zaměřuje na systematické prozkoumávání prostoru řešení tím, že si udržuje historii nedávných tahů (tabu list) a zabrání tak jejich opakování. Na rozdíl od GA, tabu vyhledávání se zaměřuje na zlepšování jednoho řešení v čase a používá mechanismy paměti k zamezení cyklení. Tato

metoda může být účinnější v některých optimalizačních úlohách, kde je důležité prozkoumat lokální okolí stávajících řešení a kde je vhodné strategicky se vyhnout opětovnému prozkoumávání nedávno navštívených řešení.

Optimalizace hejnem částic (Particle Swarm Optimization - PSO). Optimalizace hejnem částic [15] je optimalizační algoritmus inspirovaný sociálním chováním ptáků nebo ryb. Stejně jako GA se snaží nalézt optimální řešení v prostoru možností, ale místo použití evolučních operací se částice (řešení) v roji pohybují ve směru optimálních řešení na základě vlastních zkušeností a zkušeností svých sousedů. PSO je obzvláště účinný pro spojité optimalizační problémy a jeho hlavní výhodou je jednoduchost implementace a schopnost rychle konvergovat k optimálnímu řešení, i když může být náchylný k uvíznutí v lokálních optimech.

Optimalizace mravenčí kolonií (Ant Colony Optimization - ACO). Optimalizace mravenčí kolonií [16] je optimalizační technika inspirovaná chováním mravenců při hledání potravy a jejich schopností nalézt nejkratší cestu k zdroji potravy. ACO používá skupinu agentů (mravenců), kteří prozkoumávají prostor řešení a komunikují mezi sebou pomocí umělých feromonů. Na rozdíl od GA, kde populace prochází evolučními změnami, v ACO agenti iterativně zlepšují řešení na základě zpětné vazby od ostatních agentů. ACO je zvláště účinná pro problémy trasování a plánování, kde je možné efektivně využít její schopnost najít optimální cesty v grafech.

5.2 Alternativy k multiagentním systémům

Umělé neuronové sítě (Artificial Neural Networks - ANN). Umělé neuronové sítě [17] jsou výpočetní modely inspirované biologickými neuronovými sítěmi, které se používají pro učení, rozpoznávání vzorů a predikci. Stejně jako multiagentní systémy (MAS), ANN mohou zpracovávat komplexní informace a adaptovat se na základě zkušeností. Na rozdíl od MAS, které jsou založeny na interakci mezi autonomními agenty, ANN modelují problémy pomocí vrstvených sítí neuronů a učení se založeného na posílení spojení mezi těmito neurony. ANN jsou obzvláště silné v úlohách, kde je potřeba extrahovat vzory z velkých datových sad, zatímco MAS excelují v modelování komplexních systémů s interakcemi mezi agenty.

Distribuované výpočty (Distributed Computing). Distribuované výpočty [18] jsou metodou zpracování informací, kde úkoly jsou rozděleny mezi více výpočetních uzlů spojených v síti. Podobně jako MAS, distribuované výpočty umožňují řešení složitých problémů paralelizací a kooperací mezi uzly. Na rozdíl od MAS, které se zaměřují na autonomii a rozhodovací schopnosti jednotlivých agentů, distribuované výpočty se soustředí na efektivní rozdělení a zpracování

úloh bez nutnosti modelovat chování nebo interakce na úrovni agentů. Distribuované výpočty jsou ideální pro výpočetně náročné úlohy, zatímco MAS nabízejí lepší nástroje pro modelování a simulaci dynamických systémů s mnoha interagujícími komponentami.

Inteligence hejna (Swarm Intelligence). Inteligence hejna [19] je subdisciplína umělé inteligence, která se zabývá kolektivním chováním decentralizovaných, sebeorganizovaných systémů, přírodních nebo umělých. Tato metoda je podobná MAS v tom, že oba přístupy využívají interakce mezi jednotlivci (nebo agenty) k dosažení globálního chování nebo řešení problémů. Na rozdíl od MAS, které mohou zahrnovat široké spektrum interakcí a složitých rozhodovacích procesů mezi agenty, hromadné chování se typicky zaměřuje na jednodušší pravidla interakce vedoucí k emergentním chováním. Tento přístup je obzvláště účinný v optimalizačních úlohách a problémech, které vyžadují koordinované chování, jako je vyhledávání cest nebo distribuované řízení.

6 Definice projektu

Genetické algoritmy (GA) jsou známé svou schopností dosahovat efektivních výsledků, které často překvapí svou originalitou. V evolučním kontextu jsou zase mikroskopické organismy, zejména bakterie, působivé díky své rychlé adaptaci na změny v prostředí. Tuto schopnost lze připsat jejich rychlé reprodukci, která umožňuje bakteriím podstupovat evoluční změny během dnů, nebo často dokonce hodin.

V rámci tohoto projektu proto využíváme bakterie jako modelové agenty v simulaci, která pomocí multiagentních systémů (MAS) umožní zkoumat nejen procesy adaptace a evoluce, ale i dynamiku kooperace a konkurence v neustále se měnícím prostředí.

Hlavním cílem projektu je vytvořit simulaci, v níž budou agenti, reprezentovaní bakteriemi, vzájemně interagovat v dynamickém prostředí. Díky využití genetických algoritmů budou tito agenti schopni upravovat své genetické vlastnosti tak, aby se efektivně přizpůsobili novým podmínkám prostředí. Vzájemné interakce a adaptace agentů poskytnou vhled do dynamiky chování a evolučních strategií, což přispěje k pochopení principů evoluce a adaptability v umělých systémech.

7 Návrh a implementace projektu

7.1 Výběr programovacího prostředí

Pro realizaci projektu bylo zvoleno prostředí GameMaker Studio 2, což je vývojové prostředí od společnosti Yoyo games určené především pro tvorbu videoher. Volba tohoto prostředí byla motivována předchozími zkušenostmi s ním a jeho

vhodností pro simulaci GA a MAS. GameMaker Studio 2 poskytuje intuitivní a flexibilní nástroje pro rychlou a efektivní vývoj simulací, což je klíčové pro experimenty s genetickými algoritmy a chováním multiagentních systémů.

7.2 Architektura Systému

Návrh systému. Vytvořený systém je založen na simulaci prostředí bez překážek, ve kterém interagují agenti reprezentovaní bakteriemi. Tyto bakterie jsou rozlišeny čtyřmi barvami, přičemž každá barva symbolizuje odlišnou skupinu s různými strategiemi pro soutěžení o zdroje. Jednoduchost prostředí umožňuje soustředit se na interakce mezi agenty a dynamiku populace.

Návrh Agentů. Agenti tvoří jádro multiagentního systému, v tomto projektu jsou všichni navrženi jako homogenní a reprezentováni formou bakterií. Tento design implikuje, že jejich schopnost vnímat okolní prostředí je omezena pouze na fyzický kontakt. Každý agent je také vybaven genetickým kódem (chromozomem), který definuje jeho chování v simulovaném prostředí, strategii pro vyhledávání zdrojů, soupeření s agenty jiných barev a reprodukci.

Návrh fitness funkce. Při vývoji systému využívajícího GA pro evoluci a adaptaci agentů, klíčovým prvkem je vhodně definovaná fitness funkce. Fitness funkce pro hodnocení agentů v tomto projektu je definována na základě schopnosti agentů získávat energii, která bude dostupná za nalezení potravy nebo eliminace konkurentů. Dalším parametrem fitness funkce je pak čas, který agent zůstane naživu a počet potomků které vytvoří. Vysoce hodnocení agenti jsou pak selektováni pro další generaci, což zajišťuje udržení a rozvoj adaptivních vlastností populace.

Selekce a Reprodukce. Pro generování nových generací agentů je aplikován selekční proces, ve kterém jsou zachováni nejlepší jedinci, aby se zabránilo ztrátě již ověřených genetických vlastností. Dále je aplikován proces křížení mezi vybranými agenty, kde dojde k vytvoření nových agentů z kombinace jejich genů, tato strategie podporuje genetickou diverzitu a umožňuje vznik nových adaptivních strategií.Poslední část nové generace je tvořena agenty s náhodnými geny pro průzkum alternativních řešení.

7.3 Implementace prostředí

Design prostředí. Pro účely simulace byla vytvořena virtuální místnost o rozměrech 1000x1000 pixelů. Tento prostor je bez překážek, poskytující agentům a dalším objektům možnost volného pohybu a vzájemné interakce.

Rozhraní objektové logiky. Za účelem zajistění konzistentní funkčnosti a efektivity simulace byla implementována třída *objectLogic*, která slouží jako rozhraní pro všechny objekty v místnosti simulace. Toto rozhraní umožňuje centralizované řízení klíčových aspektů, jako je deaktivace objektů během pauzy simulace, nebo udržení agentů a jiných objektů v rámci definovaných hranic místnosti. Této funkčnosti je docíleného pomocí *smyčky místnosti*, která transportuje objekty opouštějící jeden okraj místnosti na opačný okraj, což simuluje nekonečný prostor a umožňuje nepřerušovanou aktivitu agentů.

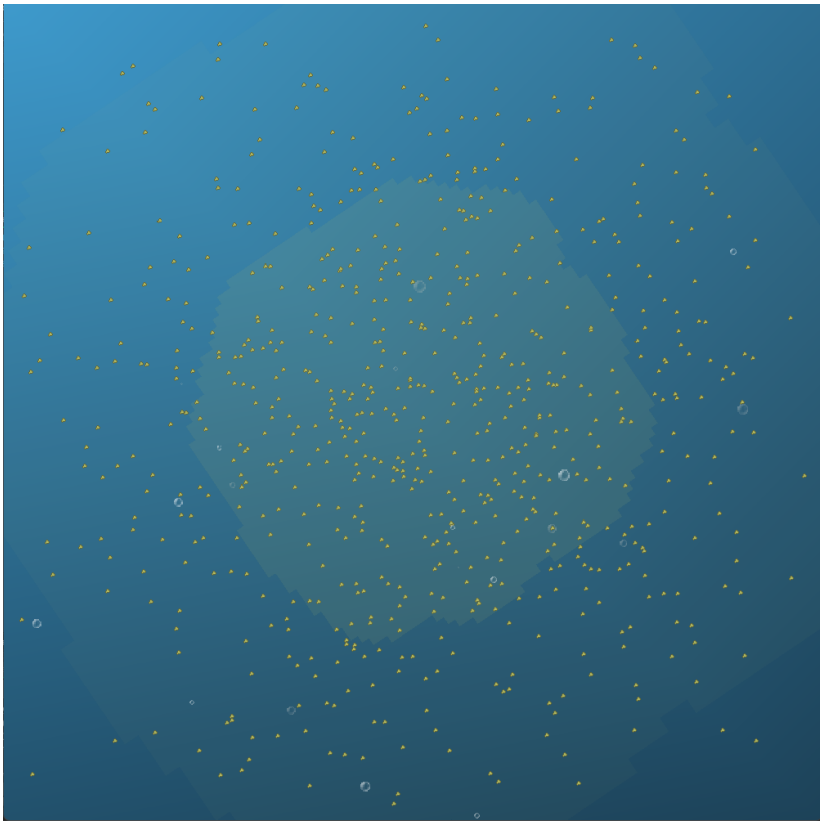
Generování Jídla. Jako klíčový zdroj energie pro agenty, byl zaveden systém distribuce jídla, který generuje jídlo od středu místnosti s klesající hustotou. Tento mechanismus podporuje konkurenci mezi agenty a poskytuje strategickou lokaci v rámci místnosti. Regenerace jídla je závislá na množství chybějícího jídla, což zajišťuje dynamickou rovnováhu mezi dostupností zdrojů a spotřebou agentů.

Vizuální Prvky. Pro prezentaci projektu došlo také k přidání několika vizuálních prvků do místnosti jako jsou stíny, animace, nebo stoupající bubliny. Tyto prvky přinášejí do simulace více realismu a zajímavosti, ale pro zefektivnění simulace a soustředění se na analýzu lze tyto grafické prvky podle potřeby deaktivovat.

7.4 Základní struktura agenta

Ve vývoji tohoto multiagentního systému jsou všichni agenti navrženi jako homogenní entity. To znamená, že každý z nich sdílí stejnou základní architekturu a soubor schopností. Proto jsem se rozhodl implementovat univerzální rozhraní *agentDefault*, které definuje základní atributy a funkce nutné pro všechny agenty. Mezi hlavní atributy zahrnuté v rozhraní agentDefault patří:

- *Energie*: Základní zdroj agenta, který umožňuje pohyb.
- *Barva*: Identifikace týmu agenta
- *Životy*: Vitální stav agenta.
- *Skóre*: Výsledek fitness funkce.
- *Stav chování*: Aktivita kterou agent dělá.
- *Směr pohybu*: Směr pohybu v místnosti.
- *Krokové počítadlo waitFor*: Počítadlo pro přepínání stavů agenta.



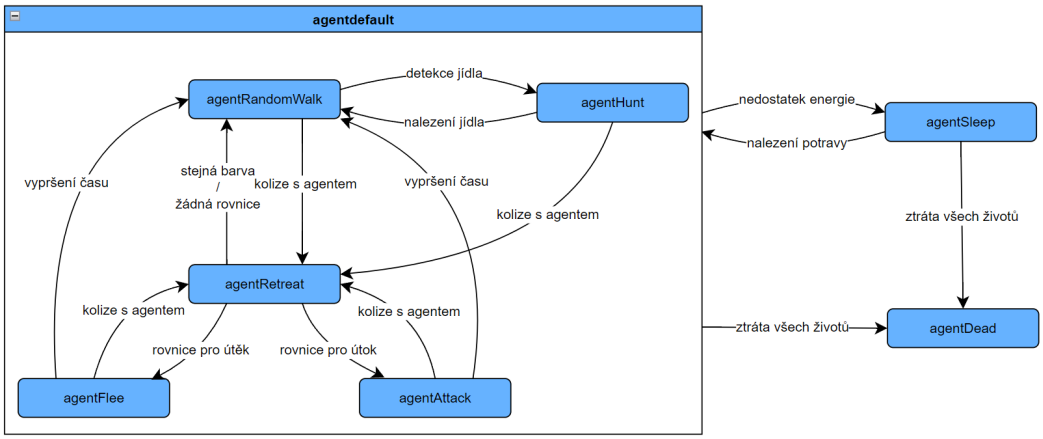
Obrázek 5: Místnost bez agentů s jídlem a vizuálními prvky

Kromě těchto atributů rozhraní agentDefault obsahuje definici genetického kódu agenta, který je tvořen geny rychlosti, agresivity, obratnosti, strachu a citlivosti. Tyto genetické geny jsou pro každého agenta unikátní a jsou určeny v momentě jeho inicializace. Tyto geny mají zásadní vliv na jeho chování v simulovaném prostředí. Dále rozhraní agentDefault poskytuje několik funkcionálních aspektů:

- *Cyklická ztráta energie*: Simuluje metabolismus agenta, vedoucí k postupnému snižování jeho energie v závislosti na jeho rychlosti a agresivitě.
- *Reprodukce*: Umožňuje agentovi vytvořit potomka po dosažení určité energetické úrovně.
- *Spánek*: Stav, do kterého agent upadne při nedostatku energie.
- *Smrt*: Stav, který nastane pokud agent vyčerpá všechny své životy.
- *Vykreslování*: Vizualizace agenta.
- *Interakce*: Reakce na setkání s ostatními agenty a přechod do náležitého stavu.

7.5 Stavy agenta

V rámci tohoto projektu jsou chování a akce agentů definovány prostřednictvím různých stavů. Každý stav je reprezentován objektem, který implementuje univerzální rozhraní agentDefault a rozšiřuje jeho základní funkce specifickým chováním pro daný stav.



Obrázek 6: Diagram přechodů mezi stavy agentů

agentRandomWalk. Základním stavem každého agenta je náhodný pohyb v místnosti. Dynamika tohoto pohybu je určena geny rychlosti a obratnosti, který ovlivňuje frekvenci změny směru pohybu agenta (viz kód 1).

```
1 direction += (random(7) - 3) * (agility * 3);
```

Zdrojový kód 1: Změna pohybu

agentHunt. Tento stav se aktivuje, když agent ve stavu náhodného pohybu detekuje potravu, což je řízené na základě genu citlivosti (viz kód 2). Tento stav řídí pohyb agenta směrem do středu místnosti, kde je vyšší koncentrace potravy. Stav končí buď vypršením času určeného genem citlivosti, nalezením potravy, nebo kolizí s jiným agentem.

agentRetreat. Stav ústupu se aktivuje po kolizi s jakýmkoli agentem a symbolizuje okamžik paniky a úniku. Agent zde hodnotí informace, které získal od druhého agenta a rozhoduje o dalším postupu. Doba trvání tohoto stavu je nepřímo úměrná genu rychlosti agenta.

```
1 if(geneticSensitivity > (irandom(100)+1)) {
2   waitFor = geneticSensitivity * irandom(30);
3   direction = point_direction(x, y, room_width/2, room_height/2);
4   instance_change(agentHunt, false)
5 }
```

Zdrojový kód 2: Kód pro přechod do stavu agentHunt

agentAttack. Pokud dojde ke kolizi s agentem jiné barvy a je splněna rovnice založená na agresivitě (viz kód 3), agent přechází do útočného stavu, kde dochází ke zvýšení jeho rychlosti a spotřeby energie v závislosti na agresivitě.

```
1 if(((15 * geneticAggression) * (energy / maxEnergy)) > irandom(50)){
2    speed = (geneticSpeed / 10) * (1 + geneticAggression / 5);
3    energyConsumption *= (1 + (geneticAggression / 5));
4    direction = point_direction(x, y, collidedObject.x,
        collidedObject.y);
5    instance_change(agentAttack, false);
6 }
```

Zdrojový kód 3: Kód pro přechod do stavu agentAttack

agentFlee. V případě kolize s nepřátelským agentem a splnění podmínek založených na strachu (viz kód 4) agent aktivuje stav útěku, kde se pokouší o prudkou změnu směru za účelem úniku z nebezpečné situace. Stav pokračuje až do vypršení doby určené genem strachu nebo do okamžiku další kolize.

```
1 if (((15 * geneticFear) / ((energy / maxEnergy) * entityHealth)) >
    irandom(50)){
2    waitFor = 10 * geneticFear;
3    agility = geneticAgility * geneticFear / 5;
4    direction += (agility * 12) * randomSign();
5    instance_change(agentFlee, false);
6 }
```

Zdrojový kód 4: Kód pro přechod do stavu agentFlee

agentSleep. Když agentovi dojde energie, přechází do stavu spánku. V tomto stavu zůstává nehybný a nereaguje na okolní podněty, přesto stále může absorbovat potravu. Postupně ztrácí životy (viz kód 5), což může vést k jeho konečné smrti, pokud nenarazí včas na jídlo.

```
1 if(waitFor < 1){
2    entityHealth--;
3    waitFor = 90;
4 }
```

Zdrojový kód 5: Cyklycká ztráta čivotů ve stavu spánku

agentDead. Finální stav, ve kterém je agent vyřazen ze simulace. V tomto stavu je agent neaktivní a z hlediska simulace neviditelný, ale jeho data zůstávají až do konce cyklu, aby mohli být vyhodnoceny.

7.6 Implementace genetiky

V simulovaném prostředí projektu dochází k reprodukci agentů dvěma způsoby: spontánně během simulace, když agenti akumulují dostatek energie, nebo systematicky na začátku každého cyklu, při vytváření nové generace. Tyto procesy zajišťují neustálou evoluci a adaptaci agentů na podmínky prostředí.

Mechanismus reprodukce:

- K reprodukci dojde když agent akumuluje dostatek energie, nebo po střetu dvou agentů stejné barvy, pokud dohromady disponují 1,5 násobek potřebné energie pro reprodukci.
- Potomci dědí genetický materiál od svého rodiče, přičemž u dvojic rodičů se bere průměr jejich genů. Následuje mírná modifikace genů o číslo do malé konstanty, což napomáhá udržení genetické diverzity a podporuje vznik nových adaptivních charakteristik.

Struktura nové generace

- *Selekce nejúspěšnějších jedinců*: 1/5 nové generace je přímo odvozeno od nejúspěšnějších agentů dané barvy z předchozího cyklu, čímž jsou uchovány výhodné genetické vlastnosti.
- *Mírná mutace*: Další 1/5 agentů je generováno z mírně mutovaných genů, které reprezentují variace na průměrné geny od úspěšných agentů, což podporuje konvergenci k optimu.
- *Rekombinace genů*: 2/5 generace vzniká rekombinací, tedy náhodným výběrem genů od nejúspěšnějších agentů. Tato metoda také zvyšuje genetickou diverzitu a potenciál pro adaptaci.
- *Náhodná genetika*: Zbylá 1/5 agentů představuje jedince s kompletně náhodně generovanými geny, což zabraňuje genetické homogenizaci a podporuje objevování zcela nových vlastností.

7.7 Implementace uživatelského rozhraní

Projekt poskytuje dvě základní varianty simulace, každá s odlišným zaměřením a možnostmi:

Režim Simulace (Simulation Mode). Nabízí klasickou strukturu genetických algoritmů s definovanými cykly, v rámci kterých dochází k tvorbě nových generací na základě selekce a křížení. Tento režim je ideální pro pozorování evoluce a adaptace bakterií napříč cykly.

Sandbox Režim (Sandbox Mode). Představuje neomezený experimentální prostor, kde evoluce volně probíhá bez procesu selekce, křížení a cyklů. Uživatelé se mohou zaměřit na sledování přirozeného vývoje a vzájemných interakcí mezi bakteriemi.

Nastavení Simulace. Před zahájením simulace umožňuje uživatelské rozhraní přizpůsobit klíčové aspekty simulovaného prostředí a počátečních podmínek:

- *Genetické nastavení*: Umožňuje specifikaci genetické konfigurace pro jednotlivé barvy, včetně nastavení rozsahu genů.
- *Výběr barev*: Uživatelé si mohou vybrat, které barvy se zúčastní simulace.
- *Podmínky prostředí*: Nastavení jako je teplota a hustota potravy v prostředí.
- *Počáteční generace*: Definuje, kolik bakterií každé barvy bude přítomno na začátku simulace.

Pro Režim Simulace je navíc možné:

- *Nastavení délku cyklu*: Určuje, jak dlouho bude trvat každý cyklus.
- *Přeskočení cyklů*: Umožňuje rychlou simulaci cyklů, což urychluje evoluční proces.
- *Uzamčení barvy*: Funkce pro zastavení vývoje u specifikované barvy, usnadňuje provádění srovnávacích studií.

Vizualizace a Analýza. Během simulace jsou uživatelům k dispozici nástroje pro vizualizaci a analyzování evolučních trendů:

- *Grafy v Reálném Čase*: Ukazují průměrné hodnoty genů a počet jedinců pro každou barvu.
- *Historický Přehled*: Poskytuje grafický přehled počtu jedinců pro jednotlivé barvy v napříč cykly, což umožňuje hlubší pochopení dynamiky generací.

Obrázek 7: Nastavení simulace

8 Experimenty a pozorování

U všech experimentů se očekává oscilace okolo optimálních hodnot kvůli náhodnému generování jídla, mutací a úvodnímu umístění agentů.

Experiment 1: Evoluce bez konkurence

Cílem tohoto experimentu je analyzovat jak se bude jedna barva bakterií vyvíjet v prostředí, pokud nebude mít konkurenci.

- **Počáteční podmínky**: Jedna barva bakterií, počáteční hodnota genů 5, rozpětí genů ±5. Teplota -0°C, hustota jídla 100%, velikost generace 15, délka cyklu 60 sekund.
- **Hypotéza**: Očekává se, že absence konkurence a krátká délka cyklu upřednostní rychlejší bakterie a agresivita se stane nepříznivým genem z důvodu zvyšování spotřeby energie.

Výsledky. Po 100 cyklech rychlost bakterií kolísala okolo hodnoty 7, zatímco agresivita klesla pod 1, což odpovídá očekáváním. Nízká teplota prostředí negativně ovlivnila bakterie s vyšší rychlostí, které nepřežily celý cyklus. Počet bakterií se udržoval mezi 120 a 180.

Obrázek 8: Graf s počtem agentů každé barvy

V módu Sandbox někdy bakterie vyhynuly, v jiných se počet ustálil na 20 až 30 jedincích, a v některých případech počet rychle rostl až k 5000+. Klíčem k úspěchu byla skupina pomalejších bakterií sbírající jídlo ve středu místnosti a několik rychlejších jedinců získávajících jídlo z periferie. Bez konkurence se průměrná rychlost snížila pod 1 a agresivita dosáhla hodnoty kolem 0.1.

Experiment 2: Adaptace na extrémní prostředí

Cílem tohoto experimentu je pozorovat, jak se bakterie adaptují na extrémní teplotní podmínky.

- **Počáteční podmínky**: Jedna barva bakterií, počáteční hodnota genů 5, rozpětí genů ±5, teplota -30°C, hustota jídla 100%, velikost generace 15, délka cyklu 60 sekund.
- **Hypotéza**: Bakterie by měly nejprve vykazovat problémy, avšak časem se adaptovat na nízkou teplotu.

Výsledky. Po 100 cyklech rychlost bakterií kolísala okolo 4, agresivita klesla pod 1. Počet bakterií se v prvních 30 cyklech pohyboval od 0 do 60, přičemž

několikrát došlo k vyhynutí celé populace. V následujících 70 cyklech nedošlo k žádnému vyhynutí, s počtem bakterií oscilujícím mezi 10 a 70.

V módu Sandbox jsem postupně snižoval teplotu a čekal než se populace ustálí. Podařilo se mi udržet populaci okolo 100 jedinců při teplotě -20°C, ale populace bakterií začala pomalu vymírat při snížení teploty na -30°C. Tato teplota je očividně příliš nízká pro dlouhodobou populaci.

Experiment 3: Účinek uzamknutí barvy na evoluci

Cílem tohoto experimentu je prozkoumat, jak uzamčení evoluce pro určitou barvu ovlivňuje celkovou dynamiku a diverzitu v ekosystému.

- **Počáteční podmínky**: Dvě barvy bakterií s počátečními geny 5 a jedna barva uzamčena. Teplota 0°C, hustota jídla 100%, velikost generace 15, délka cyklu 60 sekund.
- **Hypotéza**: Z počátku by měli být oba druhy vyrovnané, ale postupně by měla barva s volnou evolucí získat výhodu.

Výsledky. Po 100 cyklech barva s uzamknutou evolucí převažovala v 8 cyklech, zatímco ve zbývajících 92 cyklech dominovala barva s volnou evolucí.

Experiment 4: Studie vzniku agresivního chování

Cílem tohoto experimentu je analyzovat podmínky vedoucí ke vzniku agresivního chování mezi bakteriemi.

- **Počáteční podmínky**: Vybrány dvě barvy bakterií s počátečními geny nastavenými na 1 a 5 a variabilitou genů ± 5, přičemž barva s nižšími geny je geneticky uzamčena. Teplota simulace 100°C, hustota jídla 20%, velikost generace 100, délka cyklu 60 sekund.
- **Hypotéza**: Nedostatek jídla by měl vést k vypěstování agresivního chování u bakterií s vyššími statistikami, které se budou živit bakteriemi druhé barvy.

Výsledky. Po 100 cyklech barva s možností evoluce si vyvinula agresivní chování vůči druhé barvě. Gen agresivity dosahoval hodnoty kolem 5. Každý cyklus se vyznačoval rychlým nárůstem agresivních bakterií, které se začaly živit bakteriemi druhé barvy. Avšak kvůli nedostatku jídla a omezenému počtu bakterií, populace agresivních bakterií začala po 15 sekundách hladovět a do konce cyklu vymřela.

Závěr

V této bakalářské práci jsem se zabýval integrací genetických algoritmů do multiagentních systémů s cílem prozkoumat jejich potenciál v simulaci a optimalizaci složitých problémů. Práce kombinovala teoretický přístup s praktickou implementací, kde byly genetické algoritmy aplikovány na modelování a evoluci autonomních agentů v dynamickém prostředí. Analýza a experimenty prokázaly, že spojení těchto dvou technologií umožňuje efektivnější řešení problémů, jako je optimalizace chování agentů a adaptace na proměnlivé podmínky prostředí.

Na základě teoretického průzkumu těchto dvou technologií a praktického experimentování s vytvořeným simulovaným prostředím bylo dosaženo několika klíčových poznatků, které pomáhají k hlubšímu porozumění dynamiky evolučních procesů a interakcí mezi autonomními agenty.

Hlavním přínosem této práce je demonstrace, jak může spojení genetických algoritmů s multiagentními systémy posílit adaptabilitu agentů a zároveň podpořit jejich vzájemnou spolupráci, což vede k inovativním a efektivním řešením. Práce také přispěla k lepšímu porozumění dynamice genetických algoritmů a multiagentních systémů a jejich aplikaci na řešení reálných problémů.

Conclusions

In this bachelor thesis, I focused on the integration of genetic algorithms into multi-agent systems to explore their potential in the simulation and optimization of complex problems. The work combined a theoretical approach with practical implementation, where genetic algorithms were applied to the modeling and evolution of autonomous agents in a dynamic environment. Analysis and experiments demonstrated that the integration of these two technologies enables more efficient problem-solving, such as optimizing agent behavior and adapting to changing environmental conditions.

Based on the theoretical exploration of these two technologies and practical experimentation with the created simulated environment, several key insights were achieved that contribute to a deeper understanding of the dynamics of evolutionary processes and interactions between autonomous agents.

The main contribution of this work is to demonstrate how the integration of genetic algorithms with multi-agent systems can enhance the adaptability of agents while also fostering their mutual cooperation, leading to innovative and efficient solutions. The work also contributed to a better understanding of the dynamics of genetic algorithms and multi-agent systems and their application in solving real-world problems.

A Obsah elektronických dat

src/

Adresář se zdrojovým kódem (s příponou např. `.yyp`) pro simulaci Bacteria's Genome, určený k použití v prostředí GameMaker Studio 2. Kód obsahuje všechny potřebné skripty, objekty a nastavení pro sestavení a spuštění simulace. Soubor je nezbytný pro další vývoj a úpravy simulace.

text/

Adresář s textem práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech (textových) příloh, a všechny soubory potřebné pro bezproblémové vytvoření PDF dokumentu textu, tj. zdrojový text textu a příloh, vložené obrázky, apod.

README.txt

Textový soubor (s příponou např. `.txt`) s informacemi o provozu aplikace a funkčním postupu k zprovoznění software. Obsahuje kroky pro instalaci, konfiguraci a spuštění aplikace, jakož i popis jednotlivých funkcí a modulů softwaru.

Bacteria's Genome/

Adresář se spustitelným souborem (s příponou např. `.exe`) pro simulaci Bacteria's Genome. Tento soubor umožňuje uživatelům snadno spustit simulaci na platformě Windows bez nutnosti další instalace nebo konfigurace. Obsahuje všechny kompilované zdroje a závislosti potřebné pro běh simulace.

Literatura

[1] *Evoluční algoritmy*. [online]. 2024 [cit. 2024-4-14]. Dostupný z: ⟨https://www.kiv.zcu.cz/studies/predmety/uir/gen_alg2/E_alg.htm⟩.

[2] *Genetický algoritmus*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Genetick%C3%BD_algoritmus⟩.

[3] *Evolving Neural Networks through Augmenting Topologies*. [online]. 2002 [cit. 2024-4-14]. Dostupný z: ⟨https://nn.cs.utexas.edu/downloads/papers/stanley.cec02.pdf⟩.

[4] *Antenna Technology*. [online]. 2024 [cit. 2024-4-14]. Dostupný z: ⟨https://www.jpl.nasa.gov/nmp/st5/TECHNOLOGY/antenna.html⟩.

[5] *DarwinTunes*. [online]. 2024 [cit. 2024-4-14]. Dostupný z: ⟨http://darwintunes.org/⟩.

[6] *AlphaGo*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://en.wikipedia.org/wiki/AlphaGo⟩.

[7] *Multiagentní systém*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.wikipedia.org/wiki/Multiagentn%C3%AD_syst%C3%A9m⟩.

[8] *Multi-agent systems*. [online]. 2024 [cit. 2024-4-14]. Dostupný z: ⟨https://www.simulace.info/index.php/Multi-agent_systems/cs⟩.

[9] *RoboCup 2014: robot, technologie, Brazílie*. [online]. 2014 [cit. 2024-4-14]. Dostupný z: ⟨https://magazin.aktualne.cz/kuriozity/robocup-2014-robot-technologie-brazilie/r~8bf7795a124c11e49d450025900fea04/⟩.

[10] *OpenAI Five*. [online]. 2024 [cit. 2024-4-14]. Dostupný z: ⟨https://openai.com/research/openai-five⟩.

[11] *Agent-based computational economics*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://en.wikipedia.org/wiki/Agent-based_computational_economics⟩.

[12] *Guide to Programming GridLAB-D*. [online]. 2024 [cit. 2024-4-14]. Dostupný z: ⟨https://gridlab-d.shoutwiki.com/wiki/Guide_to_Programming_GridLAB-D⟩.

[13] *Simulované žíhání*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.m.wikipedia.org/wiki/Simulovan%C3%A9_%C5%BE%C3%ADh%C3%A1n%C3%AD⟩.

[14] *Tabu search*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://en.m.wikipedia.org/wiki/Tabu_search⟩.

[15] *Optimalizace hejnem částic*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.m.wikipedia.org/wiki/Optimalizace_hejnem_%C4%8D%C3%A1stic⟩.

[16] *Optimalizace mravenčí kolonií*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.m.wikipedia.org/wiki/Optimalizace_mraven%C4%8D%C3%AD_koloni%C3%AD⟩.

[17] *Umělá neuronová síť*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.m.wikipedia.org/wiki/Um%C4%9Bl%C3%A1_neuronov%C3%A1_s%C3%AD%C5%A5⟩.

[18] *Distribuovaný výpočet*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.m.wikipedia.org/wiki/Distribuovan%C3%BD_v%C3%BDpo%C4%8Det⟩.

[19] *Inteligence hejna*. [online]. 2024 [cit. 2024-4-14]. Wikipedia: The Free Encyclopedia. Dostupný z: ⟨https://cs.m.wikipedia.org/wiki/Inteligence_hejna⟩.

[20] *Programování grafických aplikací v C++*. [online]. 2024 [cit. 2024-4-14]. Matematicko-fyzikální fakulta, Univerzita Karlova. Dostupný z: ⟨https://cgg.mff.cuni.cz/~pepca/prg022/luner.html⟩.

[21] *Genetické algoritmy*. [online]. 2024 [cit. 2024-4-14]. Fakulta matematiky, fyziky a informatiky, Univerzita Komenského. Dostupný z: ⟨https://flurry.dg.fmph.uniba.sk/webog/SuboryOG/bohdal/9-GenetickeAlgoritmy.pdf⟩.

[22] Shoham, Yoav. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press, 2009. ISBN 978-0-521-89943-7.