VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

CELULÁRNÍ AUTOMATY A PRAVIDLA DEFINOVANÁ GENETICKÝM ALGORITMEM

CELLULAR AUTOMATA AND RULES DEFINED USING GENETIC ALGORITHM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Darek Goliáš

AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. Radomil Matoušek, Ph.D.

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Darek Goliáš**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**prof. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2023/24

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Celulární automaty a pravidla definovaná genetickým algoritmem

Stručná charakteristika problematiky úkolu:

Buněčné automaty (CA, Cellular Automata) jsou systémy, které používají pravidla k popisu vývoje populace definované na diskrétní mřížce. Genetické algoritmy jsou postupy inspirované procesem přírodního výběru určené k optimalizaci. V této práci půjde o implementaci buněčného automatu, jehož pravidla nastavována s využitím genetického algoritmu (GA, Genetic Algorithms).

Cíle bakalářské práce:

1/ Nastudovat a stručně shrnout problematiku CA ve vztahu k realizaci práce.
2/ Nastudovat a stručně shrnout problematiku GA ve vztahu k realizaci práce.
3/ Realizovat vlastní implementaci 2D CA se systémem generování pravidel pomocí GA.
4/ Provést diskuzi k synergii CA a GA na vhodném příkladu.

Seznam doporučené literatury:

MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ (1993). Umělá inteligence. Praha: Academia. ISBN 80-200-0472-6.

[online Retrieved July 12, 2009]. Elementary Cellular Automaton. Wolfram Mathworld. http://mathworld.wolfram.com/ElementaryCellularAutomaton.html.

BäCK, Thomas, Ron BREUKELAAR (2005). Using Genetic Algorithms to Evolve Behavior in Cellular Automata. 1-10. 10.1007/11560319_1.

WOLFRAM, S., (1994). Cellular automata and complexity. Wokingham: Addison-Wesley. ISBN 0-201-62664-0.

UMBERTO Cerruti, Simone DUTTO, Nadir MURRU. (2020). A symbiosis between cellular automata and genetic algorithms. Chaos, Solitons & Fractals. Volume 134. ISSN 0960-0779.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2023/24

V Brně, dne

L. S.

Ing. Pavel Heriban, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Tato bakalářská práce se zabývá propojením celulárních automatů (CA) a genetických algoritmů (GA). Stručně přibližuje historii a principy CA a GA, včetně jejich klíčových komponent. Teoretická část se věnuje různým druhům CA, jejich klasifikaci, okrajovým podmínkám a příkladům využití. Dále jsou popsány genetické algoritmy, jejich reprezentace populace, selekce, křížení a mutace. Praktická část zahrnuje návrh a implementaci dvourozměrného CA se systémem generování pravidel pomocí GA.

ABSTRACT

This bachelor's thesis explores the integration of cellular automata (CA) and genetic algorithms (GA). It briefly outlines the history and principles of CA and GA, including their key components. The theoretical part covers various types of CA, their classification, boundary conditions, and examples of applications. Additionally, it describes genetic algorithms, their representation of population, selection, crossover, and mutation. The practical part involves the design and implementation of a two-dimensional CA with a rule generation system using GA.

KLÍČOVÁ SLOVA

Celulární automaty, genetické algoritmy, dynamické systémy, diskrétní modely

KEYWORDS

Cellular automata, genetic algorithms, dynamic systems, discrete models

ÚSTAV AUTOMATIZACE A INFORMATIKY

2024

BIBLIOGRAFICKÁ CITACE

GOLIÁŠ, Darek. *Celulární automaty a pravidla definovaná genetickým algoritmem.* Brno, 2024. Dostupné také z: `https://www.vut.cz/studenti/zav-prace/detail/157952`. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, Vedoucí práce: prof. Ing. Radomil Matoušek, Ph.D.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato bakalářská práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků.

V Brně dne 24. 5. 2024 .....

Darek Goliáš

PODĚKOVÁNÍ

Na tomto místě bych rád poděkoval vedoucímu své bakalářské práce, prof. Ing. Radomilu Matouškovi, Ph.D., za jeho odborné rady, připomínky a vedení. Dále bych chtěl poděkovat své rodině a přátelům za podporu nejen při studiu. Největší dík patří mé přítelkyni Odoně za její neutuchající podporu.

OBSAH

1 ÚVOD

Celulární automaty (CA) a genetické algoritmy (GA) jsou dvě významné oblasti výzkumu v oblasti diskrétních matematických modelů a evolučních algoritmů. Celulární automaty, poprvé představené Johnem von Neumannem a Stanisławem Ulamem v 40. letech 20. století, představují model výpočetního systému, který se skládá z diskrétních buněk s konečným množstvím stavů, které se vyvíjejí v diskrétním čase podle předem definovaných pravidel. Genetické algoritmy, inspirované principy přírodní evoluce, byly poprvé formálně představeny Johnem Hollandem v 60. letech 20. století a významně rozvinuty D. J. Goldbergem, který tyto algoritmy propagoval průlomovou knihou "Genetic Algorithms in Search, Optimization and Machine Learning"(1989). GA jsou využívány k hledání optimálních nebo přibližně optimálních řešení komplexních problémů prostřednictvím procesů selekce, křížení a mutace.

Cílem této bakalářské práce je propojit tyto dvě oblasti a prozkoumat synergii mezi celulárními automaty a genetickými algoritmy. Konkrétně se zaměřím na implementaci dvourozměrného celulárního automatu a vytvoření systému generování pravidel pomocí genetického algoritmu. Tento přístup umožňuje zkoumat, jak genetické algoritmy mohou přispět k nalezení pravidel, která vedou ke vzniku zajímavého, symetrického a komplexního chování celulárních automatů.

Teoretická část bude nejprve zahrnovat rešerši historického kontextu a základních principů celulárních automatů. Dále budou prozkoumány jednorozměrné (1D) a dvourozměrné (2D) celulární automaty, různé typy okrajových podmínek a klasifikaci celulárních automatů podle jejich dynamického chování. Následně bude věnována pozornost historii a základním principům genetických algoritmů, včetně jejich klíčových komponent, jako jsou selekce, fitness funkce, křížení a mutace.

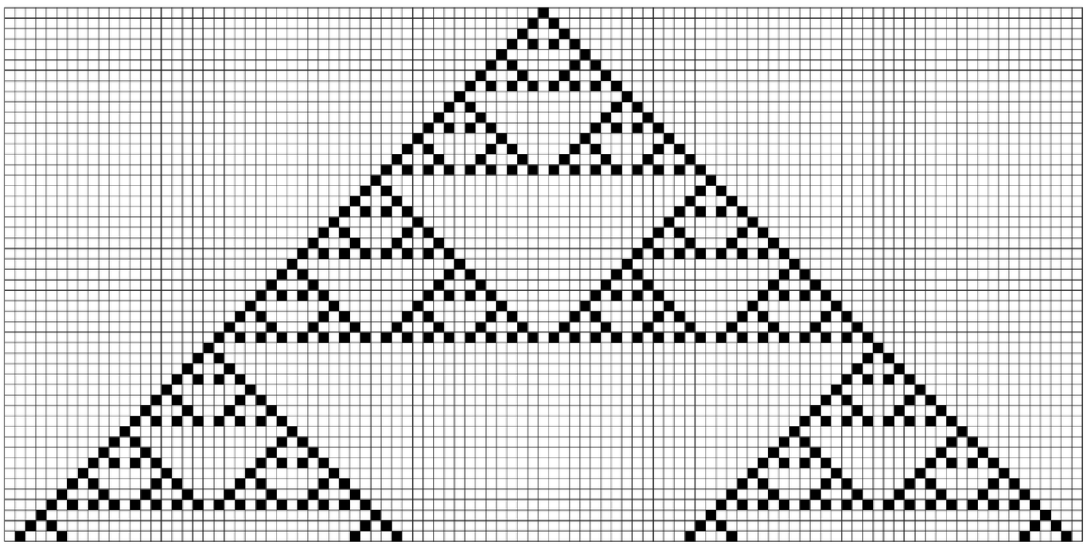
Praktická část této práce bude zahrnovat návrh a implementaci systému, ve kterém genetický algoritmus generuje pravidla pro dvourozměrný celulární automat. Tento systém bude testován na různých příkladech, aby byla demonstrována efektivita a potenciál generování pravidel. Diskuze se bude zabývat výhodami a nevýhodami propojení celulárních automatů a genetických algoritmů a limitami systémů.

Použité metody zahrnují programování celulárního automatu a genetického algoritmu, který má za úkol hledat pravidla vykazující zajímavé a komplexní chování. Pro implementaci bude jako programovací jazyk zvolen Python pro jeho relativní rozmanitost.

2 CELULÁRNÍ AUTOMATY

Celulární automat (CA, *buněčný automat*) je matematický model složený z diskrétních buněk, které mohou být v různých stavech a mění svůj stav podle pevně daných pravidel. Formální zápis pravidel celulárního automatu může být reprezentován jako funkce: $S_{i,j}(t+1) = f(S_{i,j}(t), S_{i-1,j}(t), S_{i+1,j}(t), S_{i,j-1}(t), \ldots)$, kde $S_{i,j}(t)$ je stav buňky na pozici (i, j) v čase t a f je aktualizační funkce, resp. soubor pravidel, která určují nový stav buňky na základě stavů jejich sousedů.

Buněčné automaty jsou tedy jednoduché diskrétní výpočetní modely, u kterých můžeme pozorovat zajímavé komplexní chování. Skládají se z pravidelné mřížky buněk, z nichž každá nabývá právě jednoho z určeného počtu stavů. Těch může být n, ale nejčastěji nabývají právě binárních hodnot a to 1, aneb stavu živého a 0, aneb stavu mrtvého. Stav CA bývá jednoznačně určen hodnotami v těchto buňkách. Mřížka bývá jedno, dvou, či vícedimenzionální a její tvar může být různý. Nejjednodušší takovou mřížkou je jednorozměrná čára. U dvoudimenzionální mřížky to potom může být tvar čtvercový, trojúhelníkový, či šestiúhelníkový. Počátečním stavem se rozumí přiřazení stavu každé buňce v mřížce. Ten může být nastaven náhodně, nebo může být navržen tak, aby se zkoumaly specifické vlastnosti systému. Další stavy buněk se odvíjí synchronně, krokově v čase podle stavů ostatních buněk v okolí a tvoří tak nelehce predikovatelné vzory. Okolím se rozumí buňka samotná a její bezprostřední okolí buněk. Pravidla definující změnu stavů jsou v průběhu evoluce konstantní [1].



Obr. 1: Příklad elementárního jednorozměrného CA, převzato z [2] (S. Wolfram).

2.1 Historie celulárních automatů

První návrh buněčných automatů vznikl u vědeckého pracovníka Jhona von Neumanna ve 40. letech dvacátého století. Jeho práce se zaobírala vývojem abstraktního sebereprodukujícího se automatu (stroje, schopného vlastní reprodukce) a ze začátku jeho myšlenky cílily na 3D stroje popsané parciálními diferenciálními rovnicemi. Na základě podnětů jeho kolegy Stanislawa Ulama se rozhodl model zjednodušit, a tak vznikl první 2D buněčný automat [2]. Sestrojený buněčný automat měl pro každou buňku 29 možných stavů a jeho pravidla byla komplikovaná. Komplexita pravidel byla nastavena tak, aby napodobovala činnost součástí elektronického počítače a jiných mechanických zařízení [3].

Během dalších let se buněčné automaty dostaly do širšího povědomí, a tak začaly vznikat návrhy postupně jednodušších CA schopných autoreprodukce a univerzálního výpočtu. Od počátku šedesátých let bylo zaznamenáno několik jednoduchých vlastností CA, u kterých se předpokládalo, že jsou důležité k samoreprodukci. Stanislaw Ulam okolo roku 1960 pomocí počítače na národní laboratoři v Los Alamos vytvořil příklady rekurzivně definovaných geometrických objektů - výsledků vývoje zobecněných 2D buněčných automatů. Ulam si později po získání větších snímků všiml, že v některých případech jednoduchá pravidla růstu ve výsledku generují komplexní a nepředvídatelné řešení [2].

Právě prací Stanislawa Ulama a jeho společníků na téma „simulation games“ se nechal inspirovat John Conway, který začal roku 1968 provádět experimenty s různými pravidly 2D buněčných automatů a později roku 1970 přišel se sadou pravidel, kterou nazval „Hra života“. Jednalo se o jednoduchý nekonečný dvoudimenzionální buněčný automat s ortagonální mřížkou obsahující dva stavy (buňka mohla být živá, či mrtvá). I přes jednoduchost ale hra vykazovala komplexní chování. Tento fenomén z velké části zpopularizoval Martin Gardner v článku časopisu Scientific American. Automat obsahující sadu pravidel, které popisují při jakých podmínkách buňka přežívá, umírá, či vzniká, se tak stal zábavou a prostorem pro mnohá zkoumání, a tak začaly vznikat různé zajímavé uskupení buněk a vzory, které se dělí podle jejich chování. I přes zvýšený zájem o buněčné automaty jako takové se jim nedostalo dostatečné analýzy a hlubšího zkoumání.

V 70. letech byl přiveden na svět první dvourozměrný deterministický automat, který byl vytvořen pro studium statistických vlastností plynů s názvem HPP(Hardy, Pomeau, de Pazzis) model. Tímto způsobem se otevřela cesta pro zkoumání tekutin, či zrnitých látek pomocí buněčných automatů.

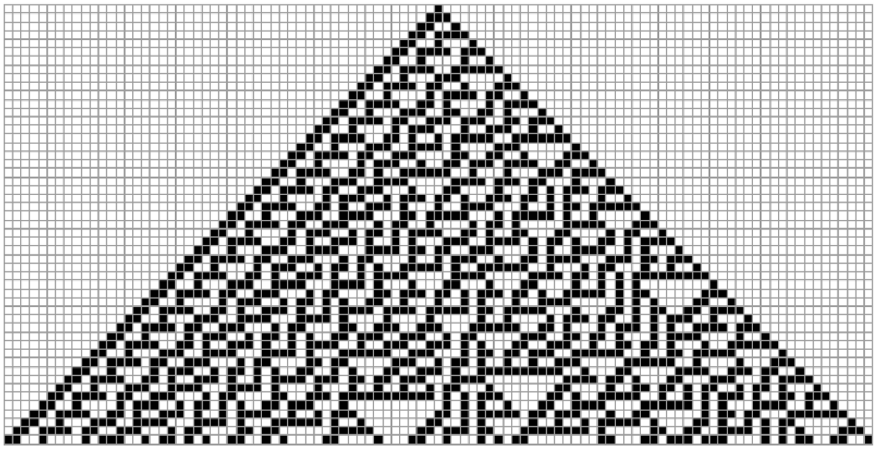
Velká změna nastala začátkem 80. let, kdy se fyzik Stephen Wolfram začal ve své práci s názvem „Statistical mechanics of cellular automata“ hlouběji zabývat buněčnými automaty. Tento zdánlivě jednoduchý, avšak komplexní svět ho natolik inspiroval, že v řadě následujících prací ještě více prohloubil své poznání a přitom světu představil snímky různých buněčných automatů tak, jak je ještě předtím neviděl [4]. Také jednorozměrné buněčné automaty podle jejich chování rozdělil do čtyř tříd. Svět celulárních automatů ho natolik pohltil, že se rozhodl tomuto fenoménu věnovat spoustu času, a tak v průběhu deseti let zkoumal různé druhy buněčných automatů a květnu roku 2002 publikoval svou knihu „A New Kind of Science“. Ta obsahuje 1200 stran a autor v ní přestavuje kompletní přehled buněčných automatů, jejich rozdělení a chování. Prokazuje také, že jdou využít k modelování komplexity reálného světa, jako třeba růst, evoluce, či chemické reakce. Wolfram je také tvůrcem programu Matematica, který je schopen tvoření simulačních modelů pomocí buněčných automatů [2].

2.2 Druhy celulárních automatů

Buněčné automaty mohou být konstruovány v různých dimenzích. To rozšiřuje jejich možné typy chování a také jejich možnosti aplikace. Můžeme je tak dělit např. podle dimenzionality na jednorozměrné, dvourozměrné a vícerozměrné celulární automaty.

2.2.1 Jednorozměrné celulární automaty

Jednorozměrné buněčné automaty (1D CA) představují základní modely, které poskytují užitečný pohled na dynamiku systémů s jednoduchými pravidly. Tento typ automatu se skládá z řady buněk, které jsou uspořádány do jednoho rozměru a rozprostírají se nekonečně v obou směrech. Každá buňka v tomto systému může nabývat jednoho z k možných stavů (například diskrétní hodnoty 0 a 1, nebo pro vizualizaci černá a bílá). V počátečním čase[1] t_0 je definován počáteční stav pro každou buňku v řadě. Během následujících generací se stavy jednotlivých buněk mění synchronně v souladu s pravidly, která závisí pouze na stavech sousedících buněk a stavu samotné buňky.

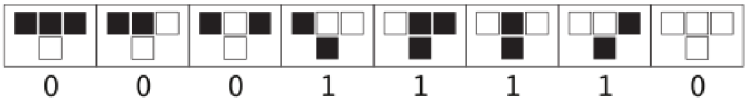


Obr. 2: Příklad elementárního jednorozměrného CA, převzato z [2] (S. Wolfram).

Mezi nejjednodušší varianty jednorozměrných buněčných automatů patří elementární CA. Tyto automaty se vyznačují tím, že každá buňka se při určování svého stavu řídí pouze předchozím stavem buněk v sousedství a předchozím stavem buňky samotné. Pro buňky v sousedství platí, že existuje právě $2 \times 2 \times 2 = 2^3 = 8$ binárních stavů, tím pádem existuje přesně $2^8 = 256$ možných elementárních buněčných automatů, z nichž lze každý určit 8bitovým číslem [5].

[1] V ohledu prezentovaných CA jde o diskrétní čas, tj. ekvivalentně bude dále hovořeno o iteračních krocích CA.

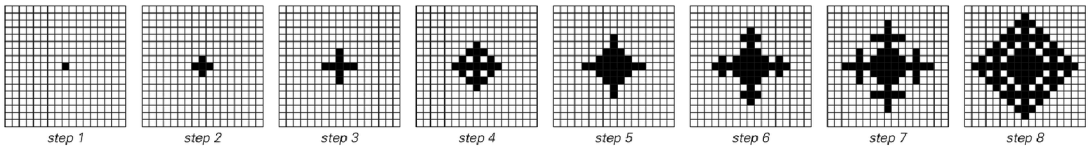
Příklad může být pravidlo 30 ($30 = 00011110_2$). V horní části diagramu se nachází tři možné stavy buněk ze sousedství v čase t_i a v dolní části výsledný stav řešené buňky v čase t_{i+1}, tzv. v další generaci.



Obr. 3: Určení stavů pro pravidlo 30, horní řádek odpovídá kroku t_i, dolní řádek je stav buňky v t_{i+1}v, převzato z [2] (S. Wolfram).

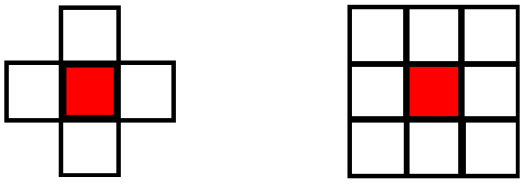
2.2.2 Dvourozměrné celulární automaty

Když dojde k rozšíření prostoru o jednu dimenzi, buněčné automaty mění své sousedství. Na rozdíl od jednorozměrných CA se nyní buňky nachází na pravidelné dvourozměrné mřížce. Každá buňka nabývá k možných stavů, nejčastěji $k = 2$ (živá, neživá), ty se pak aktualizují v čase synchronně podle pravidla, které závisí na stavu buněk v sousedství a stavu buňky samotné.



Obr. 4: Příklad 2D CA, pravidlo 942 podle S. Wolframa, převzato z [2].

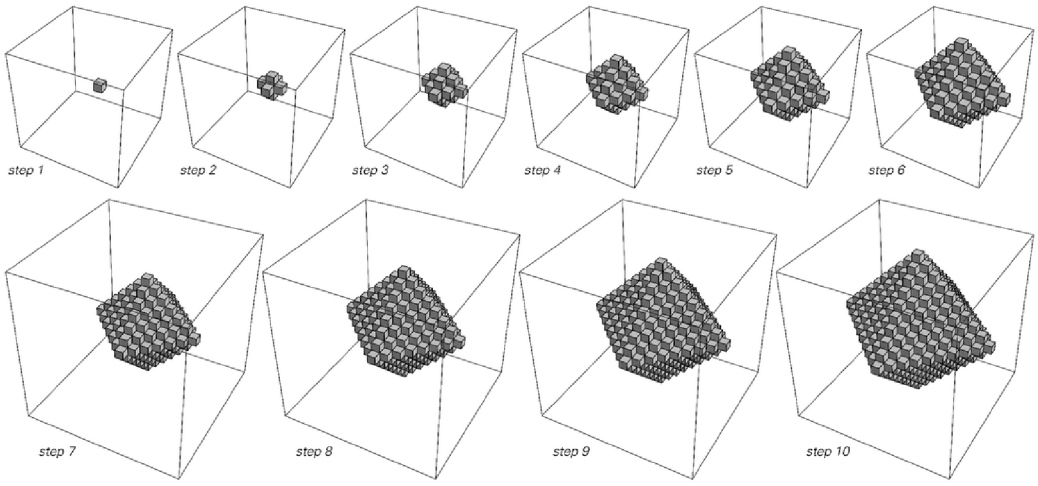
Sousedství se nejčastěji dělí na dva typy. Ty nám specifikují, jaké buňky do něj patři a započítávají se tak do funkce určující další stav buňky v čase [6]. První z nich představil Jhon von Neumann, nazývá se „Neumannovo okolí“ a skládá se z pravidelné ortagonální mřížky, čtyř přilehlých buněk a buňky samotné. Druhé nese název „Moorovo okolí“ po Edwardu F. Mooreovi a skládá se také z pravidelné ortagonální mřížky, osmi přilehlých buněk a buňky samotné. Lze uvažovat také trojúhelníkové a šestiúhelníkové mřížky.



Obr. 5: Ukázka von Neumannova (vlevo) a Moorova (vpravo) okolí.

2.2.3 Vícerozměrné celulární automaty

Vícerozměrné buněčné automaty představují pokročilý model v teorii výpočtů a komplexních systémů, který rozšiřuje koncepci klasického buněčného automatu do prostorů s více než dvěma dimenzemi. Tento rozšířený model není omezen na jedinou rovinu nebo lineární uspořádání buněk, ale umožňuje reprezentaci a analýzu jevů v trojrozměrném prostoru, čtyřrozměrném prostoru a dokonce i prostorových strukturách s ještě vyšším počtem dimenzí [7]. Tato rozšíření nabízí široké spektrum aplikací a výzkumných možností napříč různými disciplínami. Od biologie, kde mohou být použity k modelování interakcí mezi buňkami v tkáních a vývoji morfologie organismů, po fyziku, kde mohou sloužit k simulaci turbulentních proudění tekutin a modelování fyzikálních jevů ve třírozměrném prostoru [8].



Obr. 6: Příklad 3D CA, převzato z [2] (S. Wolfram).

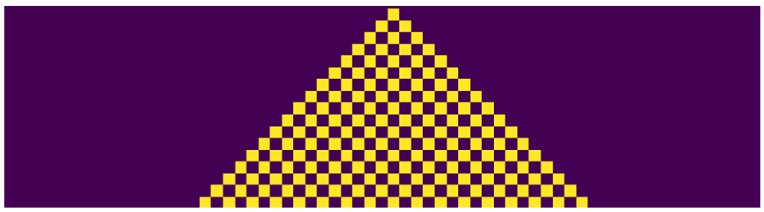
2.3 Dělení celulárních automatů podle chování

Stephen Wolfram se nechal elementárními buněčnými automaty inspirovat a tak začal experimentovat s různými pravidly, u kterých zaznamenával lišící se typy chování. I přes menší rozdíly mezi jednotlivými CA ale zjistil, že se dají kategorizovat dle typu chování do čtyř tříd [2]. Ty očísloval vzestupně podle složitosti chování:

- *Třída 1 – Homogenní chování* : CA vykazuje velmi jednoduché chování. Po několika krocích dospěje systém do ustáleného stavu a již se dále nemění.

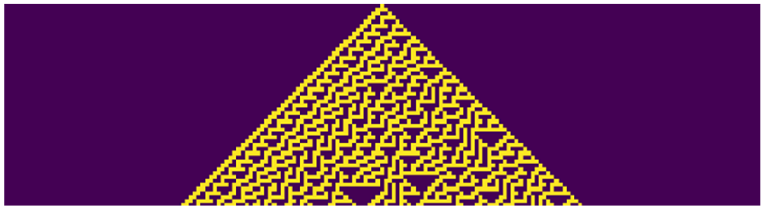
Obr. 7: Příklad třídy 1., pravidlo 0.

- *Třída 2 – Periodické chování* : Chování buněk se po výrazné počáteční aktivitě ustálí do cyklického nebo periodického vzoru, tj. po určitém, ale pevně daném počtu kroků se stav CA opakuje.



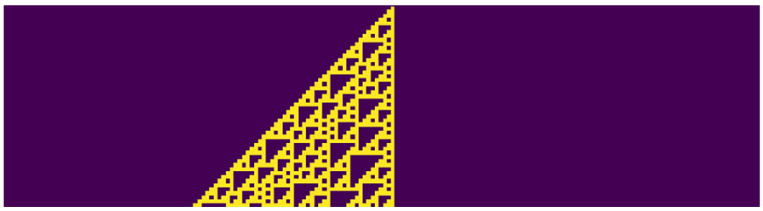
Obr. 8: Příklad třídy 2., pravidlo 250.

- *Třída 3 – Chaotické chování* : Chování je mnohem komplikovanější, působí chaoticky a nelze jednoduše najít pravidelnost. Lze díky nim generovat pseudonáhodná čísla.



Obr. 9: Příklad třídy 3., pravidlo 30.

- *Třída 4 – Komplexní vzory* : Vytváří symbiózu mezi pravidelností a náhodností. Vznikají zde lokální struktury, které se zdají být jednoduché, ale svým chováním a interakcí s ostatními vytváří komplexní podivuhodné chování.

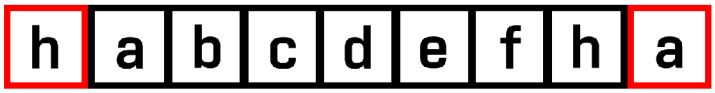


Obr. 10: Příklad třídy 4., pravidlo 110.

2.4 Okrajové podmínky celulárních automatů

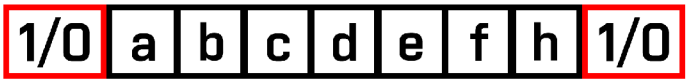
Ve formální definici CA je nutné uvažovat nekonečnou mřížku, ale při realizaci je z hlediska vypočítatelnosti a složitosti rozumné zvolit okrajové podmínky. Těmi se rozumí nastavení chování okrajů buněčného automatu. Nejčastěji se uvažují čtyři typy okrajových podmínek [9]. Ty jsou:

- *Periodická okrajová podmínka* - Dochází k slepení protějších stran, a tak pokud buňka doputuje do krajní pozice a v další generaci by měla postupovat dále, bude pokračovat na druhé straně mřížky.U jednorozměrných automatů uvažujeme pole jako smyčku. U dvourozměrných automatů se vytváří topologie torusu.



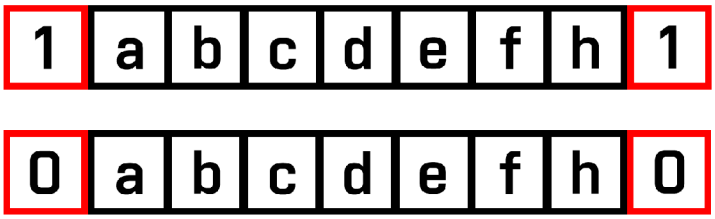
Obr. 11: Příklad periodické okrajové podmínky.

- *Náhodná okrajová podmínka* - Dochází k náhodnému nastavení konstantní hodnoty na okraji mřížky. Tento stav může být nastaven na začátku simulace a být neměnný, nebo se může měnit každou iteraci.



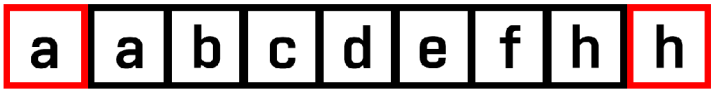
Obr. 12: Příklad náhodné okrajové podmínky.

- *Fixní (konstantní) okrajová podmínka* - Buňky na hranici mřížky mají fixně nastavenou hodnotu, která zůstává konstantní po celou dobu simulace (například 1, či 0).



Obr. 13: Příklad odrazové okrajové podmínky.

- *Reflektivní (zrcadlová) okrajová podmínka* - Stav buněk na okraji mřížky je určen ze stavů buněk ve vnitřní části prostoru. Používá se v simulacích, kde se chceme vyhnout "efektu stěny".



Obr. 14: Příklad reflektivní okrajové podmínky.

2.5 Využití celulárních automatů

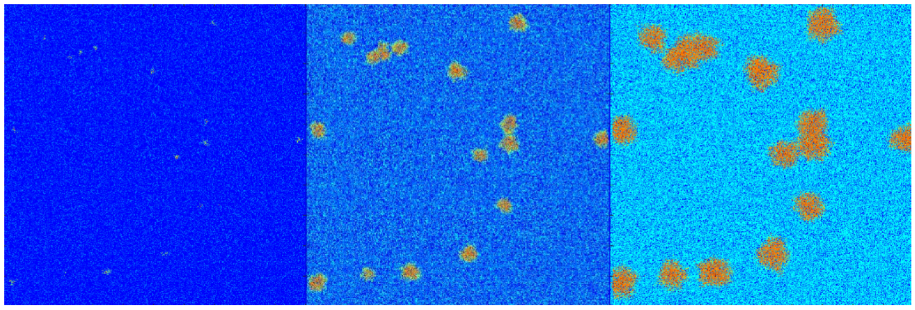
Celulární automaty představují mocný nástroj pro studium dynamických systémů a prostorově nehomogenních struktur. Jejich jednoduchá struktura a schopnost modelovat složité chování na základě lokálních pravidel vytváří ideální prostředí pro použití v různých oblastech vědy a techniky [10]. CA jsou v oblasti fyziky využívány k simulaci šíření fázových přechodů, difuze a šíření látek v prostoru s různými koncentracemi, nebo třeba dalších komplexních jevů. V biologii jsou používány k modelování buněčných interakcí, modelování evolučních procesů, nebo šíření infekčních nemocí a při plánování preventivních opatření. V sociálních vědách jsou využívány k simulaci chování jedinců ve společnosti, analýze ekonomických trhů, nebo ke studiu kolektivního chování.

2.5.1 Model šíření epidemie

Za vznikem těchto typů modelů stojí historické důvody, které sahají až do roku 1927, kdy Kermack a McKendrick sestavili model náchylných–infikovaných–uzdravených (SIR) k studiu šíření černé smrti v Londýně. V jejich studii byla navržena teorie prahu k určení, zda je infekční nemoc epidemií nebo ne, což položilo základy pro studium dynamiky infekčních nemocí [11].

Rychlá globalizace, častá cestování a kontakty mezi lidmi napříč státy mohou způsobit, že infekční nemoci se šíří neuvěřitelnou rychlostí a mohou se zdát nepredikovatelné, ale při práci s viry se zanáší velká míra abstrakce, jelikož velká část virů má velice podobné chování a tak jako inicializační podmínky konkrétních případů mohou dostačovat pouze infekčnost, inkubační doba a úmrtnost.

Kvůli své poměrné jednoduchosti, efektivitě, možnosti nastavení interakcí a lokálnímu přístupu se pro modelování jevily buněčné automaty jako správná volba pro modelování tak dynamického procesu, jakým je epidemie.



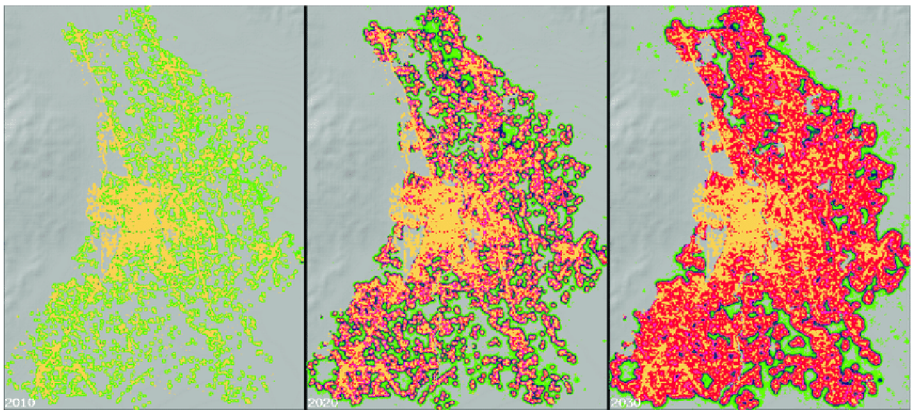
Obr. 15: Příklad prostorového vývoje epidemie, převzato z [11].

V průběhu let se matematické modely staly velmi sofistikovanými, a tak při jejich vhodném nastavení může poskytnout rozhodovacím orgánům a klinickým profesionálům užitečné informace z teoretických simulací, které mohou napomoct k zamezení šíření nemoci. Mohou se předem simulovat a zkoumat různé strategie itervencí pro kontrolu epidemie, jako třeba zamezení pohybu populace, nebo třeba vakcinace.

2.5.2 Model růstu měst

Historie modelování růstu měst sahá až do 20. století, kdy byly vyvinuty první urbanistické modely pro analýzu a predikci urbanistických jevů. Tradiční modely, jako například lineární ekonometrické modely nebo Lowryho Pittsburghský model, se zaměřovaly především na statické simulace a modelování prostorové interakce mezi různými typy pozemků v městských oblastech.

V průběhu 80. let 20. století vznikly modely využívající CA jako alternativa k tradičním urbanistickým modelům. CA modely byly považovány za vhodné pro modelování růstu měst díky své schopnosti provádět dynamické prostorové simulace s vysokým rozlišením. Díky své přirozené afinitě k Geografickým Informačním Systémům (GIS) mohou CA modely efektivně využívat prostorové informace a data získaná z dálkového snímání, což zvyšuje jejich využitelnost a aplikovatelnost v urbanistickém plánování [12].



Obr. 16: Model růstu SLEUTH: Urbanizace 2010, 2020 a 2030 , převzato z [13].

Výzkumy ukazují, že CA modely mají potenciál nejen simulovat růst měst, ale také sloužit jako nástroj pro tvorbu projekčních scénářů a zodpovídání otázek "co kdyby". V kontextu rychle se rozvíjejících zemí, jako je Indie, kde je naléhavá potřeba udržitelného urbanistického rozvoje, mohou CA modely poskytnout důležitý rámec pro plánování a rozhodování v urbanismu.

3 GENETICKÉ ALGORITMY

Ve světě lze najít mnoho reálných aplikací optimalizace cílů, jakými může být například minimalizace nákladů, spotřeby energie, či maximalizace účinnosti, nebo udržitelnosti. Ve spoustě případů jsou optimalizační problémy při formulování složité, vysoce nelineární a řešení se stává vysoce náročné. Navzdory rostoucí výpočetní síle dnešních počítačů je použití hrubých silových metod často neefektivní a nežádoucí [14] . Většina klasických optimalizačních metod, jako jsou gradientové algoritmy nebo metody založené na lokálním hledání, mohou narazit na problémy s konvergencí k lokálním extrémům a jsou závislé na počátečních podmínkách.

Trendem dnešní doby je využívání metaheuristických algoritmů v optimalizaci. V posledních letech se takových algoritmů objevilo široké spektrum, například *Optimalizace pomocí mravenčí kolonie* (ACO)[15], *Optimalizace pomocí netopýrů* (BA)[16], či *Genetický algoritmus* (GA)[17]. Tyto algoritmy bývají často jednoduché, flexibilní a přitom překvapivě účinné při globální optimalizaci.

3.1 Historie genetických algoritmů

Historie genetických algoritmů sahá až do poloviny 20. století. První průkopníci, jako John Holland, se objevili v 50. a 60. letech a položili základy pro vývoj těchto algoritmů vycházejících z Darwinovy teorie o vývoji druhů. Právě název „Genetický algoritmus“, dnes již zaužíváno jako GA, představil John Holland ve své knize „Adaptation in Natural and Artificial Systems“ [17] roku 1975 a tím položil základy všem dalším bádáním, která od té doby vznikla. Student J. Hollanda, David E. Goldberg, roku 1989 publikoval vlivnou knihu s názvem „Genetic Algorithms in Search, Optimization, and Machine Learning“ [18]. V 90. letech se GA stávaly stále populárnějšími a začaly být využívány v široké škále aplikací. Díky tomu se i původní koncept rozštěpil, a tak dnes můžeme vidět širokou škálu genetických algoritmů a jejich aplikací.

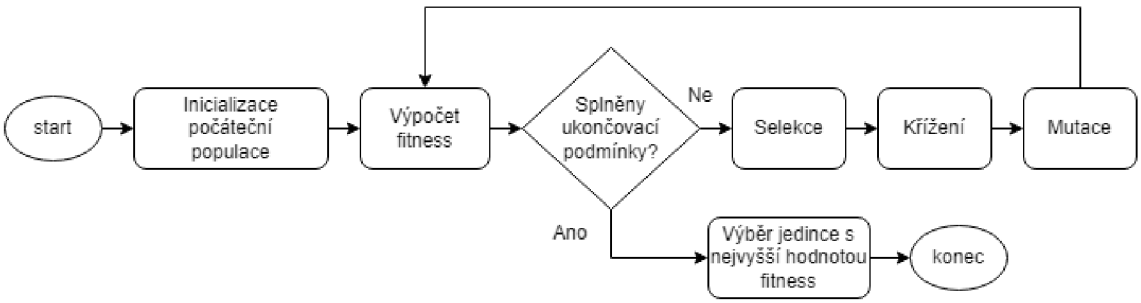
3.2 Princip genetických algoritmů

Genetický algoritmus je jedna z optimalizačních metod inspirující se principy přírodního výběru a genetiky. Jakožto evoluční algoritmus, simuluje procesy přirozeného výběru a genetického dědictví k nalezení nejlepšího řešení problému. GA pracuje s prvotní populací potenciálních řešení, která je postupně zlepšována pomocí genetických operátorů jako je selekce, křížení a mutace. Velkou výhodou GA je schopnost řešit jak spojité, tak diskrétní funkce a mnohoúčelové problémy [19]. Díky svému evolučnímu přístupu a schopnosti postupného zlepšování jsou schopny prozkoumat široký prostor hledání a nalézt optimální globální řešení. Tato schopnost je důležitá zejména při řešení složitých problémů s mnoha lokálními extrémy.

Při vytváření GA je třeba primárně definovat:

- Reprezentaci problému (genotyp) a jeho kódování problému (fenotyp).
- Fitness funkci pro ohodnocení jednotlivých řešení (jedinců populace).

Při inicializaci algoritmu dochází k počátečnímu nastavení populace, tj. všech jedinců jako potenciálních řešení problému. Každý člen populace je ohodnocen fitness funkci podle toho, jak vyhovuje pro dané řešení. Následně dochází pomocí operátoru selekce k výběru jedinců, kteří se budou podílet na nové generaci. Ti jsou vybírání s určitou pravděpodobností, která vychází z fitness každého jedince. Zde může také uplatněn koncept *elitismu*, kdy část nejlépe ohodnocených jedinců může postupovat do další generace přímo a nemusí podstupovat další operace (*selekce, křížení a mutace*).



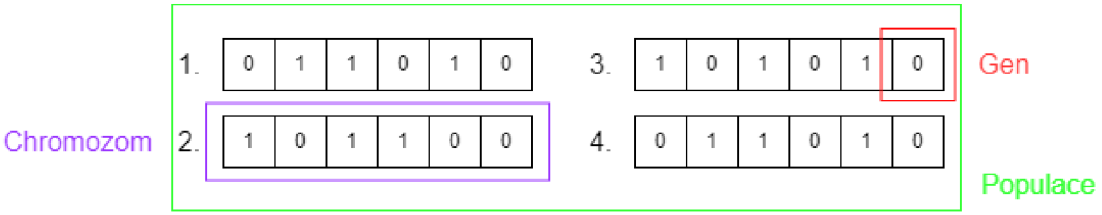
Obr. 17: Vývojový diagram genetického algoritmu (bez uplatnění elitismu).

Dále dochází k tvorbě nadcházející generace jedinců. Ta bere jedince co prošli přes selekci a vytváří potomky za pomocí operátoru křížení a mutace. Pro křížení je nutno jedince spárovat, na každý pár je potom aplikováno křížení, obvykle s velmi vysokou pravděpodobností. Při křížení dojde k výměně genů mezi jedinci. To může znamenat například rozdělení jedinců na dvě části a prohození konců. Musí se dbát na to, aby po křížení vznikl jedinec, reprezentující přípustné řešení.

Posledním krokem je mutace. Ta má velmi nízkou pravděpodobnost výskytu a může se objevit u každého jedince. Mutace napomáhá k udržení genetické varibility a tak zabraňuje tomu, že se optimalizace ubere jen k lokálním extrémům. Stejně jako tomu bylo u křížení tak i zde u mutace je nutno dbát na to, aby jedinec, který projde procesem mutace reprezentoval přípustné řešení. Tímto způsobem je získána nová generace, se kterou se proces opakuje tak dlouho, dokud není splněna ukončovací podmínka. Ukončovací podmínkou, či kombinací může být například dosažení předem daného počtu generací, překročení časového limitu, či hodnota změny fitness v definovaném generačním intervalu [20].

3.3 Genetická reprezentace problému

Typicky bývá problém reprezentován binárním řetězcem (pole bitů), což v ohledu biologie představuje genotyp. Kódování binárními řetězci má analogii v genetice, kde řetězce odpovídají chromozomům, pozice v řetězci odpovídají genům a konkrétní hodnoty odpovídají alelám [20]. Při inicializaci počáteční generace se náhodně generuje prvotní populace, která by měla rovnoměrně pokrýt všechna možná řešení. Lze uvažovat také variantu se zvýšenou koncentrací ve specifikované oblasti. Binární řetězec je pro aplikaci fitness funkce dekódován na konkrétní hodnoty (biologickým protějškem je fenotyp) dle problémové reprezentace úlohy, tj. jakým způsobem je problém parametrizován (např. binárně, celočíselně, reálnými čísly)

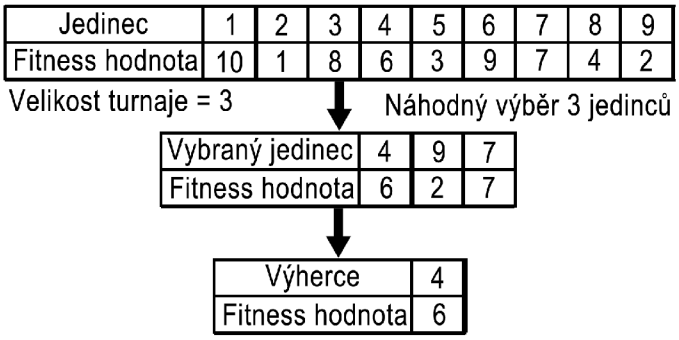


Obr. 18: Reprezentace problému, resp. kódování jedince v GA.

3.4 Selekce

Při procesu selekce dochází k výběru nejlepších jedinců pro tvorbu budoucí generace.

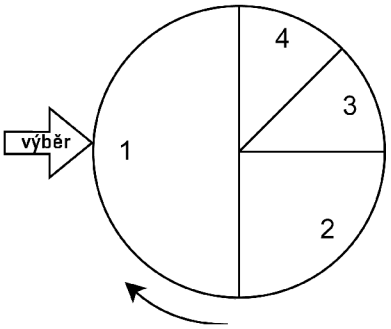
Turnajová selekce
Jednou z významných typů selekcí je tzv. Turnajový výběr. Z populace N jedinců se náhodně vybere n jedinců (typicky dva až pět). Z těchto n jedinců je dle hodnoty fitness vybrán vítěz, tj. jedinec do další generace. Těchto výběrů, resp. turnajů je logicky realizováno N (V GA se standardně počet jedinců v populaci během generací nemění.). Touto metodou se zajistí, že se nejsilnější jedinci s vysokou pravděpodobností dostanou k reprodukci, viz [21]. Pokud bychom chtěli nejlepší jedince produkovat pro následné operace s jistotou, existuje modifikace turnajového výběru ozn. jako Elitní turnajová selekce [22]. Turnajový výběr nepotřebuje seřazení populace, přičemž počtem jedinců n v turnaji může velmi dobře nastavovat sílu selekčního mechanismu, což lze považovat za zásadní výhodu této metody selekce.



Obr. 19: Příklad jednoho kroku turnajové selekce.

Ruletové kolo
Dalším typem selekce je tzv. Ruletové kolo. U té jsou rodiče vybrání podle jejich hodnoty fitness. Čím lepší tato hodnota je, tím větší mají pravděpodobnost výběru do další generace. Lze si tuto metodu představit jako klasickou ruletu, kde každý jedinec má vymezený úsek ve velikosti jeho fitness. Jedinec s větší hodnotou fitness má větší šanci být vybrán vícekrát, viz [21].

Poziční selekce
Pokud se v populaci nachází jedinec, který má v poměru ostatních mnohem větší hodnotu fitness, tak by mohlo při selekci metodou rulety dojít k nežádané chybě. Docházelo by k opakovanému výběru nejzdatnějšího jedince a výsledky by nebyly optimální. Tomuto jevu se dá předcházet pomocí metody poziční selekce. Ta bere daný problém v úvahu a představuje systém řazení jedinců, dle jejich fitness. Nejzdatnější jedinec s nejvyšší hodnotou fitness se řadí na první místo. Nejhorší jedinec



Chromozom	Fitness
1	10
2	5
3	2.5
4	2.5

Obr. 20: Schématické znázornění výběru ruletovým kolem.

zase na místo poslední. Následně dochází k přiřazení relativní fitness hodnoty, která bude pro nejhoršího jedince 1, druhého nejhoršího jedince 2, a to pokračuje až k nejlepšímu jedinci, který bude mít hodnotu relativní fitness X. Nevýhodou této metody je zpomalení procesu, jelikož se zde nereflekují reálné poměry fitness mezi jedinci, viz [21].

Elitismus

Velkou nevýhodou procesu křížení a mutace je hrozba ztráty nejlepšího jedince z populace. Tento problém řeší metoda elitismu. Elitismus spočívá v tom, že před zahájením křížení a mutace se vybere jedinec, či soubor jedinců, s nejlepší hodnotou fitness z populace a ti dále pokračují do nové generace a neúčastní se operace křížení a mutace. Tímto způsobem je zachován z jedné generace do druhé nejsilnější jedinec a tím urychlíme průběh GA. Po skončení selekce tito jedinci nahradí nejhorší jedince co prošli přes selekci,viz [21].

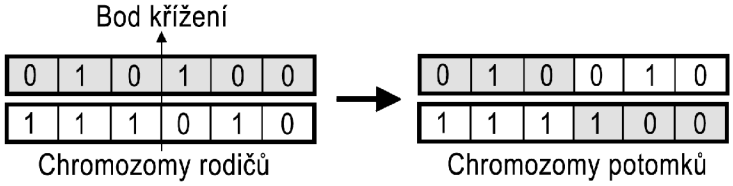
3.5 Křížení

Genetický operátor křížení je nezbytnou součástí genetického algoritmu. Do tohoto procesu vstupují vybraní jedinci z populace po selekci. Ti jsou podrobeni výměně genetické informace za cílem vytvoření silnějších potomků, kteří kombinujou vlastnosti rodičů a mohou vykazovat lepší vlastnosti [17].

Samotné křížení se může lišit svým chováním. Jak proces křížení probíhá zásadně ovlivňuje chování celého GA, proto se u jednotlivých aplikací může měnit podle toho jaký typ kódování je použito, či jaký problém je řešen tak, aby vyhovoval danému problému.

Jednobodové křížení

Nejzákladnějším typem křížení je jednobodové křížení. Po výběru rodičů se náhodně vybere jedna pozice řetězce jako bod křížení. Ta slouží jako indikátor místa, kde dojde k záměně. První potomek si přenáší první část genů do bodu křížení od prvního rodiče a druhou část od bodu křížení od druhého rodiče. Pro potomka druhého je tomu právě naopak. Tento přístup je jednoduchý a ve většině případů účinný.



Obr. 21: Schematické znázornění jednobodového křížení.

Dvoubodové křížení

Další metodou je dvoubodové křížení. Princip je podobný jako u jednobodového křížení, ale namísto jednoho bodu křížení se objevují dva. Po výběru rodičů jsou náhodně vybrány dva body křížení v řetězci. Pro prvního potomka jsou geny mezi prvním a druhým bodem křížení ponechány z prvního rodiče a zbytek genů je přebrán z rodiče druhého. Pro potomka druhého je tomu naopak.

Uniformní křížení

Dalším typem rekombinace genetické informace z rodičů je uniformní křížení. To funguje tak, že geny z obou rodičů jsou kombinovány do potomků s určitou pravděpodobností, která je stejná pro každý gen v řetězci. Pravděpodobnost výběru genu z jednoho z rodičů je obvykle určena pomocí Bernoulliho distribuce s parametrem p. Pokud je tedy $p = 0.5$ tak každý gen má 50 % šanci být vybrán z prvního rodiče a 50 % šanci být vybrán z druhého rodiče.

3.6 Mutace

Dalším genetickým operátorem u GA je mutace. Ta má důležitou pozici pro správné fungování celku GA. Zajišťuje, že nedojde ke konvergenci k lokálním extrémům tím, že s nízkou pravděpodobností náhodně změní genetické informace v jedincích populace. V rámci mutace může dojít ke změně jednoho, či více genů v řetězci jedince.

4 POZNÁMKY K REALIZACI PRÁCE

Při sestavování vlastní implementace celulárního automatu se systémem generování pravidel pomocí genetického algoritmu je důležité promyslet postup sestavení programu. Cílem je najít zajímavé vzory, které např. vykazují známky symetrie a/nebo a komplexnosti. Jako programovací jazyk je volen Python, kvůli jeho relativní jednoduchosti a flexibilitě při práci s datovými strukurami.

4.1 Návrh celulárního automatu

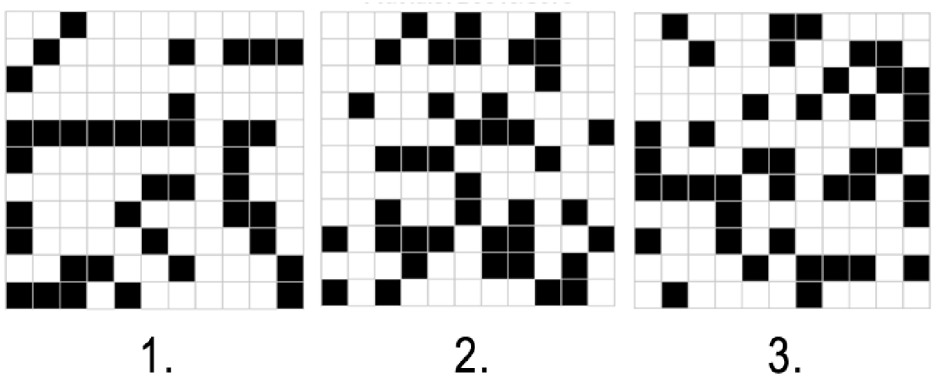
Nastavení, resp. koncept celulárního automatu pro tuto práci byl zvolen jako dvourozměrný s topologií anuloiudu, pracující s ortogonální čtvercovou mřížkou o rozměrech $N \times N$. Okrajové podmínky jsou nastaveny jako periodické. Když živá buňka doputuje na samý okraj mřížky a měla by pokračovat dál, v další iteraci bude na druhé straně. Pro výpočet přechodové funkce se využívá Moorovo okolí buněk. Celkový počet stavů buněk jsou dva, a to $1 = živá$ a $0 = mrtvá$. Tělo celulárního automatu je uloženo v souboru *cellularAutomaton.py* ve třídě *CellularAutomaton* a využívá knihoven *numpy* a *re*. Vstupní parametry této třídy jsou:

- Velikost řádků a sloupců, ty se rovnají pokud je uvažována čtvercová mřížka.
- Počáteční nastavení mřížky. Tato možnost se uvažuje pouze, pokud je ambice začínat s předem definovanou mřížkou. Jinak lze ignorovat a vygenerovat náhodnou počáteční mřížku. Počáteční mřížka by měla být ve formátu dvourozměrného pole o velikosti shodné s velikostí řádků a sloupců CA a měla by obsahovat pouze hodnoty 1, či 0.

4.1.1 Nastavení počátečního stavu mřížky

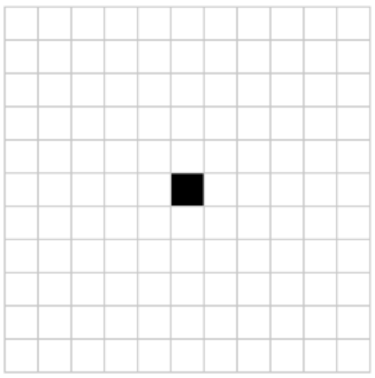
Inicializace počátečního stavu buněk na mřížce celulárního automatu významně ovlivňuje vývoj a chování celého systému. Existují dva základní přístupy k nastavení těchto počátečních stavů.

První možností je náhodné rozmístění buněk na mřížce, kde každá buňka má určitou pravděpodobnost, že bude na začátku "živá". Tuto možnost obsahuje funkce *randomize_grid*(). Ta má vstupní parametr pouze pravděpodobnost, která určuje jak často se na mřížce vyskytne živá buňka. Čím je tato hodnota větší, tím je větší i četnost živých buněk na mřížce. Pomocí této funkce je zajištěno, že při opakovaném spouštění celulárního automatu dojde vždy k náhodnému nastavení počáteční mřížky se stejnou pravděpodobností výskytu živých buněk.



Obr. 22: Tři příklady náhodného počátečního nastavení (pravděpodobnost výskytu živé buňky rovna 30 %).

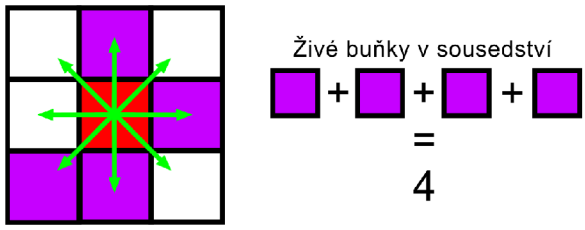
Druhou možností je přesné nastavení mřížky, kdy jsou počáteční stavy buněk pečlivě zvoleny. V kontextu práce je zvolen přístup, kdy je uprostřed mřížky umístěna jedna jediná živá buňka, zatímco zbytek mřížky zůstává neaktivní. Tuto možnost obsahuje funkce *initialize_center_cell*() Tento přístup se hodí pro zkoumání specifických vzorů a struktur, které se mohou vyvíjet z velmi jednoduchého počátečního stavu.



Obr. 23: Příklad počátečního nastavení s jednou živou buňkou.

4.1.2 Výpočet živých buněk v sousedství

Další funkcí třídy celulárního automatu je funkce *count_neighbors*(), která má jako vstupní parametry pozice x, y, na který se nachází buňka určená pro výpočet sousedství. Postupně dojde k průchodu okolím buňky a výstupem je zde číslo rovnající se počtu živých sousedů.

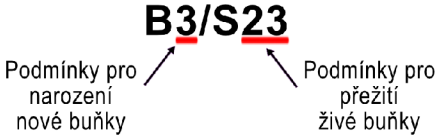


Obr. 24: Příklad výpočtu živých buněk v sousedství (Mooreovo okolí).

4.1.3 Aplikace pravidel a vytvoření další generace

Důležitou funkcí je funkce *apply_rules*(). Ta jako vstupní parametr bere set pravidel, který musí být striktně ve formátu „*Bxx/Sxx*“, kde čísla nacházející se za písmenem „B“ a před lomící čárou popisují, za jakého stavu dojde k zrození nového jedince z mrtvé buňky a čísla za písmenem „S“, popisují za jakých podmínek živý jedinec přežívá do další generace. Pravidlo projde kontrolou správnosti formátování a je rozděleno na dvě části, zbaveno písmen a rozděleno na jednotlivá čísla. Jedna část obsahuje čísla popisující vznik nové buňky a druhá obsahuje čísla určující podmínky pro přežití buňky.

Následně je iterováno přes všechny jedince v generaci a pomocí již dříve zmiňované funkce *count_neighbors*() pro každou buňku na mřížce vypočítán počet živých buněk v jejím sousedství. Následně jsou použita pravidla pro určení, zdali jednotlivé buňky přežijí do další generace, či z mrtvých buněk vzniknou živé a tyto hodnoty uloží do proměnné. Potom tuto jednotlivé hodnoty tohoto pole vloží do proměnné *next_grid*.



Obr. 25: Příklad kódování pravidel.

4.1.4 Simulace celulárního automatu

Hlavními jádry celé třídy celulárního automatu jsou funkce, pomocí kterých je CA simulován a pomocí kterých je sledováno určité chování. Právě tyto funkce budou využity dále v GA v těle fitness funkce. Pokud tomu není jinak, tak jako vstupní parametry do těchto funkcí vstupují pravidla pro CA (*rule_string*) a počet generací (*gens*), které mají být odsimulovány.

Nejmenší četnost živých buněk a zároveň alespoň jedna buňka živá
Funkce *min_count_alter*() je navržena tak, aby poskytovala podrobný pohled na schopnost pravidel celulárního automatu udržet minimální, ale stále nenulový počet živých buněk v průběhu simulace definovaného počtu generací pro náhodně nastavené počáteční rozložení mřížky. Funkce začíná náhodnou inicializací mřížky. Simulace pak probíhá přes definovaný počet generací (*gens*). V každé generaci aplikuje funkce na mřížku pravidla celulárního automatu, specifikovaná vstupním řetězcem *rule_string*.

Během simulace funkce neustále monitoruje počet živých buněk v mřížce. Počet živých buněk v každé generaci je přičítán k celkovému součtu *cell_count*. Zároveň se zaznamenává nejnižší počet živých buněk, které se vyskytly během jednotlivých generací, do proměnné *min_living_cells*.

Na konci simulace funkce vyhodnotí, zda byla mřížka schopna udržet alespoň nějaké živé buňky ve všech generacích. Pokud v některé z generací dojde k úplnému vymření buněk, vrátí funkce nulu, což signalizuje selhání pravidel udržet mřížku aktivní. Pokud mřížka přežila všechny generace s alespoň jednou živou buňkou, vrací funkce vypočtenou fitness hodnotu, která je navržena tak, aby vyšší skóre bylo přiděleno pravidlům, která efektivně minimalizují počet živých buněk, ale udržují alespoň jednu živou.

Co největší rozdílnost hodnot v dvou po sobě jdoucích mřížkách
Funkce *max_div*() se zaměřuje na sledování variability mezi generacemi v celulárním automatu. Tato funkce vypočítá celkový počet změn mezi po sobě jdoucími generacemi po dobu definovaného počtu generací. Po náhodné inicializaci mřížky a aplikaci první sady pravidel se ukládá stav mřížky. V každé následující generaci se pak porovnává nový stav mřížky s předchozím stavem a počítá se počet buněk, které se změnily (buď z živé na mrtvou nebo naopak). Tato suma změn je pak akumulována pro celou simulaci. Výsledná hodnota reprezentuje míru diverzity nebo změny v mřížce celulárního automatu.

Symetrie a zajímavé vzory

Funkce *symetry_fitness*() má za úkol hledat významné vzory a symetrie v celulárním automatu. Funguje tak, že sleduje, jak se mění vzor buněk během několika generací. Počáteční nastavení mřížky je uvažováno tak, že buňka v jejím středu je nastavena jako živá, zatím co zbytek buněk jako mrtvé. Toho je docíleno pomocí funkce *initialize_center_cell*().

Během výpočtu tato funkce klade důraz na několik klíčových vlastností vzoru. Jedna z nich je symetrie, tedy to, jak vzor vypadá ve srovnání se svým zrcadlovým obrazem. Dále se zajímá o to, jak se vzor v průběhu mění, jestli se objevují nové vzory, nebo zda se opakují. Zároveň sleduje, jak jsou buňky rozmístěny v mřížce, tedy jak moc je mřížka zaplněná. Funkce potom na základě těchto informací vyhodnocuje a vrací fitness hodnotu, která popisuje jak zajímavě se pravidlo v průběhu iterací chovalo.

Šachovnicový vzor

Funkce *alternating_pattern*() nabízí pokročilý přístup k analýze a hodnocení komplexních šachovnicových struktur v rámci buněčného automatu. Tato funkce je navržena tak, aby identifikovala výskyt šachovnicového vzoru v posledních 20 generacích simulace. Tento přístup je zvolen pro zrychlení celkového času stráveného hledáním. Na začátku je pomocí funkce *initialize_center_cell*() inicializována středová buňka na hodnotu 1 a zbytek mřížky je nastaven na hodnotu 0. Po této inicializaci následuje iterace přes definovaný počet generací (*generations*), během kterých se na mřížku aplikují pravidla pomocí metody *apply_rules*(). Pravidla určují, jak se buňky změní na základě stavu jejich sousedů. Po aplikaci pravidel je stávající mřížka (*grid*) aktualizována na nově vypočítanou mřížku (*next_grid*).

V posledních 20 iteracích simulace CA se provádí kontrola celé mřížky proti dvěma modelovým šachovnicovým vzorům, kde jeden začíná s živou buňkou a druhý s mrtvou. Shoda modelového vzoru s vzorem na mřížce je použita pro výpočet *pattern_score*. Vyšší hodnoty tohoto skóre naznačují, že dané pravidlo bylo účinné v generování a udržování komplexních a esteticky zajímavých šachovnicových vzorů.

4.2 Návrh genetického algoritmu

Genetický algoritmus byl implementován s cílem nalézt nejoptimálnější pravidla pro dané chování celulárního automatu. Kód byl strukturován do tříd a metod, které umožňují jednoduchou a efektivní manipulaci s pravidly a populací pravidel. Kód se nachází v souboru *geneticAlgorithm.py* a využívá knihoven *random*, *ThreadPoolExecutor* a *operator*. Vložena je zde také třída *CellularAutomaton*.

4.2.1 Vytvoření počáteční populace

K inicializaci počáteční populace dochází ve funkci *generate_rule_string*(). Pro každé nově generované pravidlo jsou náhodně vybírány podmínky pro aktivaci buňky v okolí. Pravidlo je vytvořeno ve formátu „Bxx/Sxx“, kde B značí podmínky pro narození nové buňky a S značí podmínky pro přežití již existující buňky. Konkrétně jsou podmínky pro narození a přežití buňky vybírány náhodně z množiny čísel od 0 do 8. Tyto čísla reprezentují počet sousedních živých buněk, který je nutný pro aktivaci dané buňky. Po vygenerování podmínek jsou tyto podmínky setříděny a spojeny do řetězce, viz obr. 25.

Celý proces generování počáteční populace pravidel je opakován tolikrát, kolik je požadovaný počet jedinců v populaci. Každý nový jedinec je tak tvořen náhodně vybranými podmínkami, čímž se zajistí diverzita v počáteční populaci a možnost prohledávání různých částí prostoru možných pravidel.

4.2.2 Vyhodnocení fitness

Vyhodnocení fitness napomáhá k určení úspěšnosti jednotlivých jedinců v populaci a následnému rozhodnutí, kteří z nich budou vybráni pro reprodukci a budou postupovat do další generace.

Princip vyhodnocení fitness je založen na simulaci chování jedinců v prostředí nebo na základě určitých kritérií a metrik, které mohou být významné pro daný problém. V kontextu práce, fitness funkce hodnotí, jak dobře se dané pravidlo chová v rámci simulace automatu. To může zahrnovat faktory jako je počet živých buněk, stabilita stavu, vývoj struktury, nebo další charakteristiky, které byly popsány v rámci rozboru třídy celulárního automatu, viz podkapitola 4.1.4.

Proces vyhodnocení fitness probíhá následovně: Nejprve je každé pravidlo v populaci aplikováno na počáteční stav mřížky buněk a následně je simulováno chování CA po určitý počet generací nebo do dosažení určeného kritéria ukončení. Během této simulace je sledován vývoj stavu mřížky buněk v závislosti na aplikovaných pravidlech. Provádíme deset paralelních simulací pro každé pravidlo v populaci. Tento přístup umožňuje zejména u náhodného nastavení počátečního stavu vykazovat stabilní výsledky pro pravidla CA a zároveň zaručuje poměrnou rychlost oproti sériovému přístupu.

Po dokončení simulace jsou výsledky vyhodnoceny a na základě nich je každému jedinci v populaci přiřazena fitness hodnota. Tato hodnota vyjadřuje úspěšnost daného pravidla v rámci simulace automatu. Čím vyšší je hodnota fitness, tím lepší je dané pravidlo a má větší šanci být vybráno pro reprodukci v další generaci.

4.2.3 Proces selekce

Selekce je realizována ve funkci *select_parents*(). V Genetickém algoritmu slouží k výběru rodičů z populace, kteří budou použiti pro reprodukci (křížení). Cílem selekce je preferovat jedince s vyšší fitness, aby se zachovaly jejich dobré vlastnosti a aby bylo zajištěno pokračování evolučního procesu směrem k lepším řešením.

V této implementaci GA jsou použity dva přístupy k selekci. Prvním je selekce rodičů pomocí ruletového kola. Princip ruletového kola spočívá v tom, že každý jedinec má pravděpodobnost být vybrán pro rodičovství úměrnou jeho fitness hodnotě. Čím vyšší je fitness jedince, tím vyšší je pravděpodobnost, že bude vybrán jako rodič, viz obr. 20.

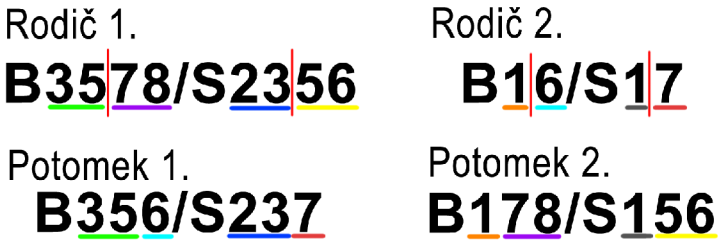
Druhým přístupem je turnajová selekce. Ta spočívá v tom, že z populace N pravidel je náhodně vybráno právě n jedinců, kteří jsou vybrání do duelu. Nejčastěji je uvažováno $n = 2$ až $n = 5$. Z turnaje odchází jedinec s nejvyšší hodnotou fitness jako výherce, viz obr. 19. Těchto výběrů, resp. turnajů je logicky realizováno N (V GA se standardně počet jedinců v populaci během generací nemění.).

Mezi těmito přístupy je možno přecházet. Z důvodů možnosti nastavení selekčního tlaku bylo ale u práce výhradně pracováno se selekcí turnajovou.

4.2.4 Proces křížení

Po výběru rodičů z populace pravidel je postupováno k operátoru křížení. Ten je důležitou součástí GA z důvodu udržení diverzity pravidel a jejich následné konvergenci k výsledkům. Při křížení dochází k výměně genetického materiálu mezi rodiči s cílem získání ještě lepších vlastností od potomků.

Křížení je uloženo ve funkci *crossover*(). Vstoupí do něj dva vybrané sety pravidel. Každé je rozděleno na dvě části. První obsahující podmínky pro narození buňky. Druhé obsahující podmínky pro přežití buňky. Obě dvě části jsou následně rozděleny na poloviny. První potomek tak dědí první polovinu podmínek pro narození buňky od prvního rodiče a druhou zase od druhého. Nápodobně tomu je tak i pro podmínky pro přežití. Druhý potomek potom dědí první polovinu od rodiče druhého a druhou polovinu od rodiče prvního. V posledním kroku jsou nově vytvořené podmínky pro narození a podmínky pro přežití uloženy do formátu „Bxx/Sxx“ a následně je provedena kontrola správnosti formátu. Výsledkem jsou dva potomci obsahující rekombinaci genů z rodičovských pravidel.



Obr. 26: Příklad procesu křížení jedinců.

4.2.5 Proces mutace

Po procesu křížení jsou všichni noví jedinci vystaveni možnosti mutace. Ta s pravděpodobností nejčasteji 1 % bere jedince a upravuje náhodně jeden jeho gen tak, aby v populaci byla zajištěna diverzita a nedocházelo k rychlé konvergenci k lokálním extrémům.

Operátor mutace je v práci reprezentován funkcí *mutate*(). Ta pomocí funkce *random.random*() vybere náhodné číslo mezi 0 až 1. To je srovnáno s hodnotou $self.mutation\_rate$. Pokud je vybrané náhodné větší, jedinec nepodstupuje mutaci a pokračuje beze změny. Pokud však je vybrané číslo menší, jedinec je vystaven mutaci.

Při mutaci je stejně jako u křížení pravidlo rozděleno na dvě části, poté je rozhodnuto, která část podstoupí mutaci. Z vybrané části je náhodně selektován jeden gen, u kterého se hodnota změní na novou, která se v dané části ještě nenachází. Poté je vybraná část spojena zpět se zbytkem do formátu „Bxx/Sxx“ a celé pravidlo je podrobeno kontrole formátování a duplikátů.

4.2.6 Proces evoluce

Během procesu evoluce, který je prováděn opakovaně v požadovaném počtu generací, je každé pravidlo v populaci určeno na základě jeho schopnosti plnit dané chování. Hodnocení fitness je prováděno simulací CA, kde jsou zkoumány různé charakteristiky vyvíjejících se mřížkových struktur, jako je diverzita vzorů, symetrické vzory a schopnost generovat stabilní či zajímavé struktury.

Na základě výsledků se vybírají pravidla pro reprodukci. Ta probíhá výhradně turnajovou selekcí. Po selekci jsou vybraná pravidla kombinována křížením, kde se genetický materiál rodičovských pravidel rekombinuje předá potomkům, kteří následně podstupují mutaci. Mutace je prováděna s určitou pravděpodobností a zahrnuje změny v řetězcích pravidel, což může vést k novým variacím chování.

Celý proces je zakončen elitismem, kde nejlepší jedinec z generace je přímo přenesen do nové generace bez úprav, což zajišťuje, že nejlepší nalezené řešení není ztraceno. Ten v nové generaci nahradí jedince s nejnižší hodnotou fitness.

Takto se proces opakuje po požadovaný počet generací, na jehož konci je vybráno nejlepší pravidlo, které je považováno za optimální řešení dané úlohy simulace celulárního automatu.

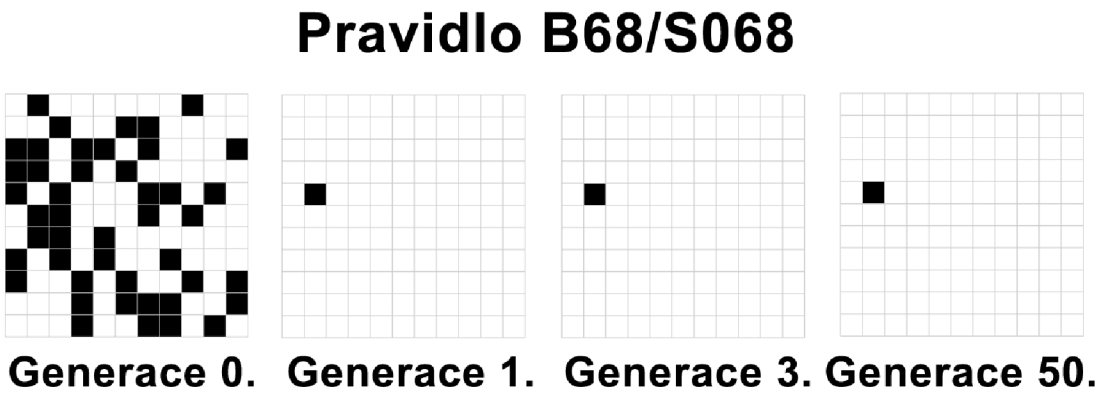
5 EXPERIMENTY S CA A NÁVRHEM PRAVIDEL S GA

V této kapitole jsou zhodnoceny a diskutovány výsledky ve formě pravidel, získaných pomocí genetického algoritmu. Cílem bylo identifikovat nové možnosti využití celulárních automatů v simulacích komplexních systémů a zjistit, do jaké míry mohou být pravidla efektivně generována pomocí genetického algoritmu. Získané výsledky jsou analyzovány a popsány, také jsou diskutovány limity simulace pravidel.

Ve všech případech je uvažována a zobrazována mřížka 11×11, pro další velikosti jsou doplněny pravidla v dodatkových tabulkách, či v textu. V genetickém algoritmu bylo vždy využito turnajové selekce. Velikost populace GA je rovna 150, pravděpodobnost mutace je 1 %. Algoritmus běží po celkovou dobu 30 generací a poté se vyhodnotí jedinec s nejvyšší hodnotou fitness. Délka simulování jednotlivých pravidel CA pro vyhodnocení fitness je 100 generací.

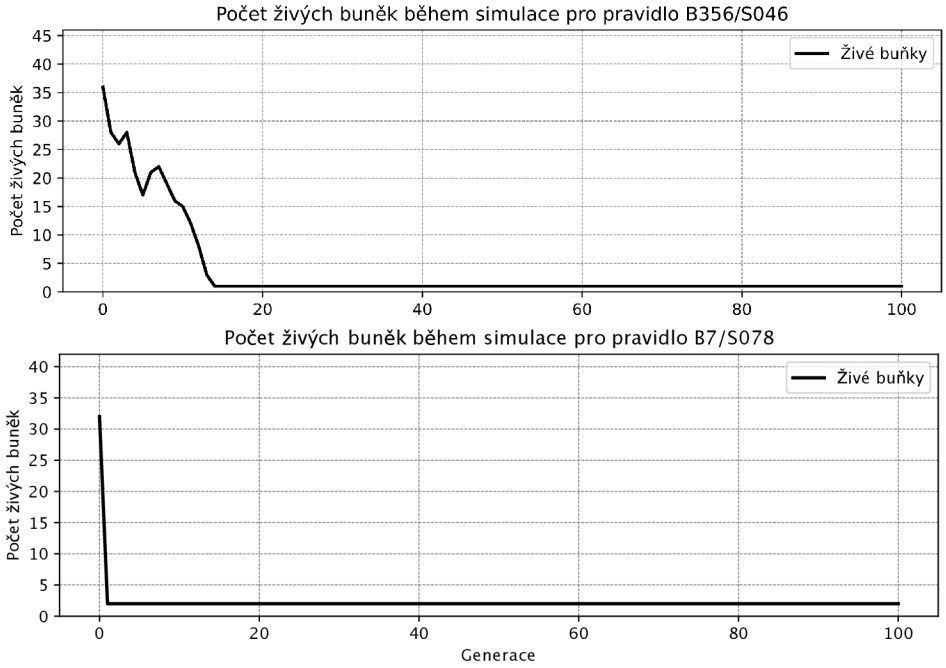
5.1 Minimální buněčná aktivita

Prvním sledovaným chováním je co nejmenší četnost živých buněk v průběhu simulace a zároveň alespoň jedna živá buňka na mřížce. Jako fitness funkce zde figuruje funkce *min_count_alter*(). Počáteční nastavení buněk na mřížce je náhodné s pravděpodobností 30 % výskytu živé buňky. Výsledkem genetického algoritmu bylo pravidlo pro CA, které reprezentuje jedince v množině pravidel, které jsou schopné z náhodného rozložení vždy udržet minimální a stabilní život na mřížce v celém průběhu simulace.



Obr. 27: Průběh simulace pravidla pro minimální buněčnou aktivitu.

Na obr. 27 můžeme vidět, že se z počáteční generace, kde jsou živé buňky náhodně rozložené po mřížce se redukují jedinci na co nejmenší počet, ale stále zůstává alespoň jeden živý. Tento stav je neměnný až do konce simulace.



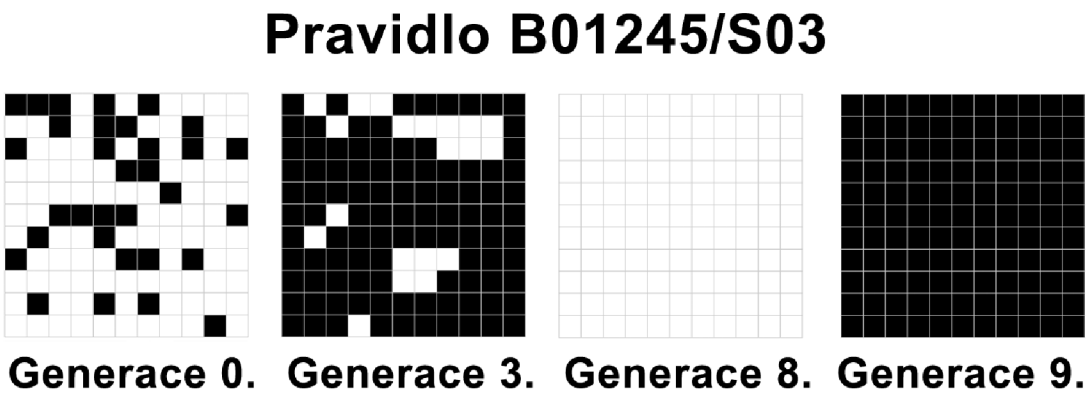
Obr. 28: Vývoj počtu živých jedinců pro pravidla s minimální buněčnou aktivitou.

Lze vidět, že se pravidlo pro různá počáteční nastavení dokáže adaptovat a dokáže udržet život na mřížce na minimální hodnotě do konce simulace. Tato vlastnost pravidla se přenáší i na jinak rozměrné mřížky, nutné je ale uvažovat velikost mřížky větší než 6×6, kvůli periodickým okrajovým podmínkám.

Po sérii testování a optimalizace byla nalezena tyto pravidla: „B47/S08, B57/S058, B8/S078, B37/S038, B7/S067, B356/S046, B8/S067, B68/S058, B7/S056“

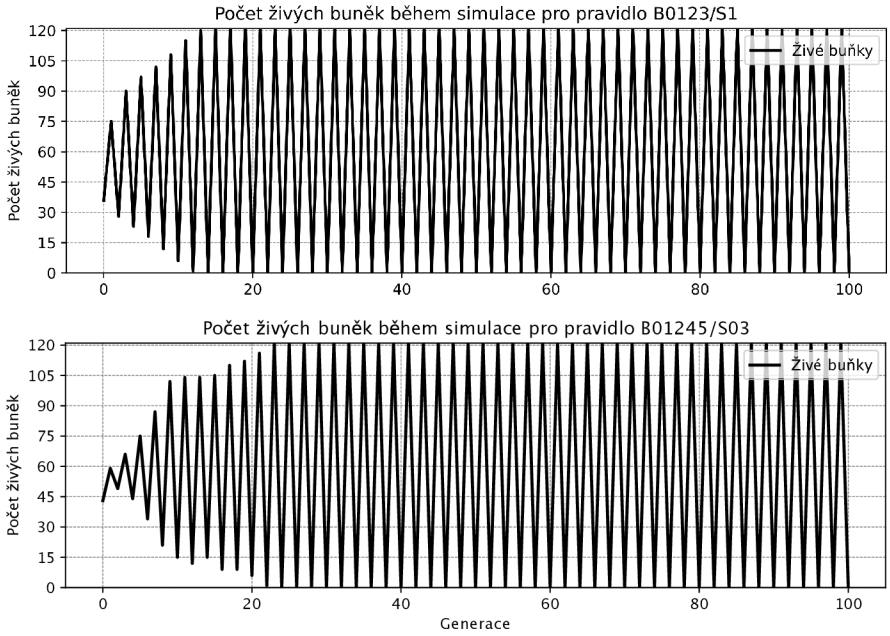
5.2 Maximální rozdílnost mřížek

Druhým sledovaným chováním je co největší rozdílnost dvou po sobě jdoucích mřížek. Jako fitness funkce zde figuruje $max\_div()$. Ta má za cíl zajistit co největší diverzitu a variabilitu, při počátečním náhodném nastavení mřížky. Po několika krocích od náhodného nastavení by měl automat dospět do stavu, ve kterém bude střídavě dosahovat maximálního a minimálního počtu živých buněk na mřížce.



Obr. 29: Průběh simulace pravidla pro maximální rozdílnost.

Hledané pravidlo má za úkol na počáteční mřížce náhodně zadané vždy konvergovat do stabilního periodického stavu. Pro tento případ jsou pravidla použitelná pro všechny velikosti mřížek, ale stejně jako u minulého použití tak i zde je nutno uvažovat velikosti vyšší jak 6×6.

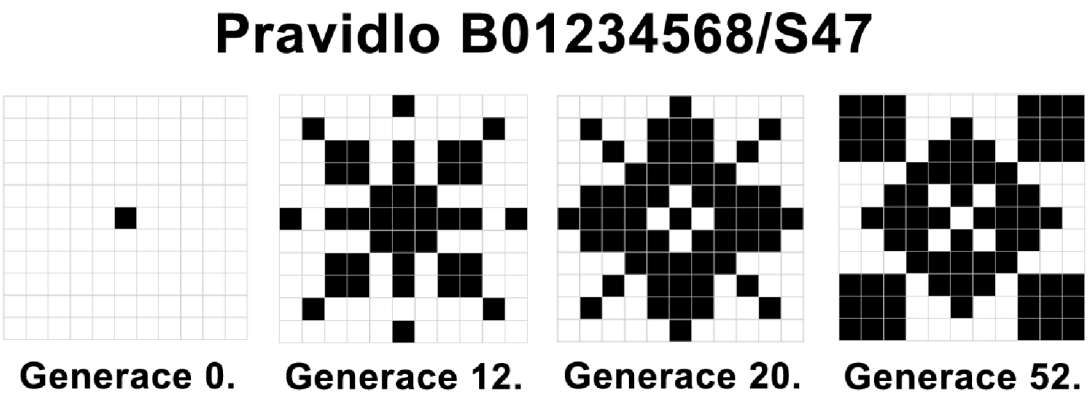


Obr. 30: Vývoj počtu živých jedinců pro pravidla maximální diverzity.

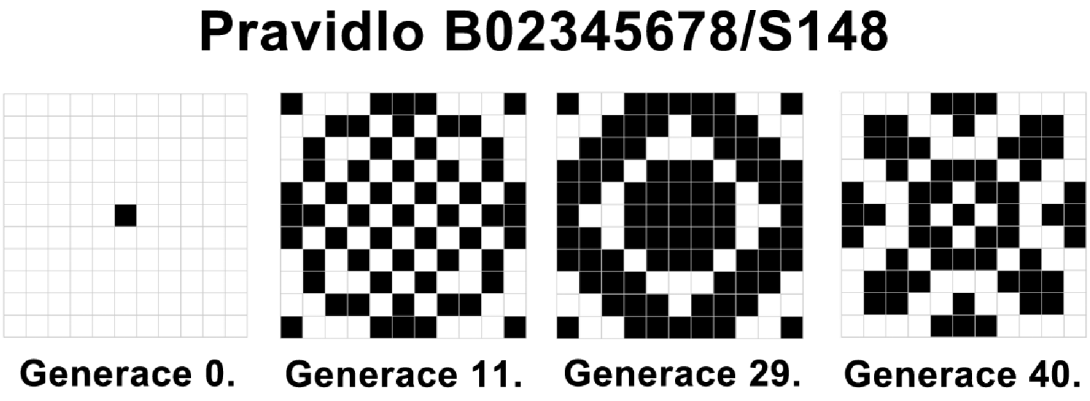
V rámci genetického algoritmu bylo provedeno testování a vyhodnocení a byly nalezeny následující pravidla: „B01234/S2, B01245/S03, B01234/S03, B01234/S02, B01237/S02, B13456/S08, B0123/S1, B01246/S02“

5.3 Symetrie a zajímavé vzory

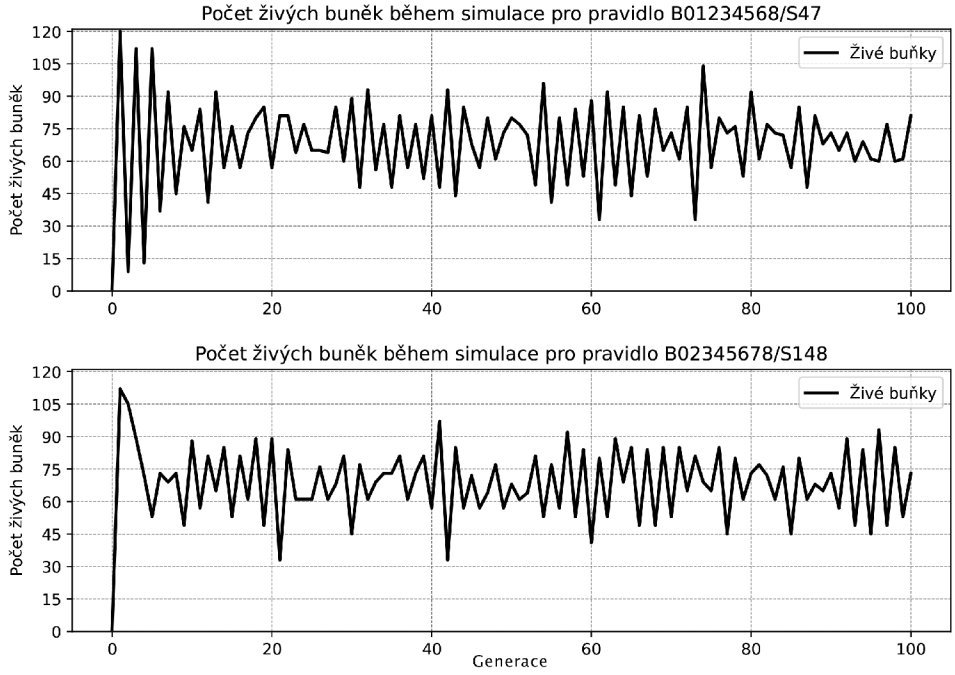
Další hledanou charakteristikou byla symetrie a celková zajímavost vzorů. Pro vyhodnocení fitness byla použita funkce $symetry\_fitness()$. Na rozdíl od předchozích dvou případů je zde počáteční mřížka nastavena pouze na jednoho živého jedince ve středu mřížky.



Obr. 31: Symetrické tvary tvořené generovaným pravidlem.



Obr. 32: Symetrické tvary tvořené generovaným pravidlem.



Obr. 33: Vývoj počtu živých jedinců pro pravidla symetrie.

Bylo odpozorováno, že jednotlivá pravidla jsou efektivní jen pro daný rozměr mřížky. Proto byl vytvořen seznam, obsahující ukázky pravidel pro jednotlivé velikosti mřížek, viz tab. 1.

9 x 9	**11 x 11**	**13 x 13**
B034567/S0368	B0123456/S147	B023456/S048
B013456/S1468	B01234568/S47	B0234567/S135
B13456/S0368	B0345678/S36	B0234567/S47

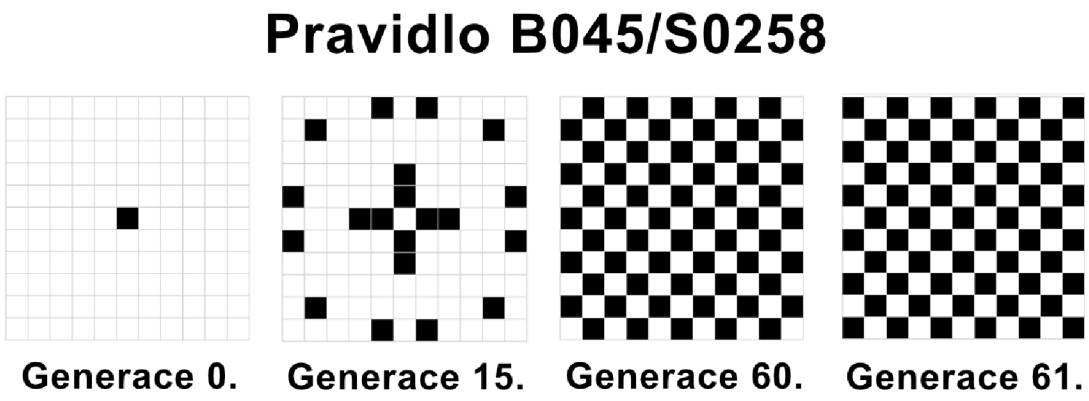
10 x 10	**12 x 12**	**14 x 14**
B02345678/S012467	B0345678/S0368	B02345678/S238
B02345678/S46	B0345678/S368	B0345678/S37
B01234568/S258	B02345678/S347	B02345678/S047

Tab. 1: Nalezená pravidla symetrických vzorů pro dané mřížky.

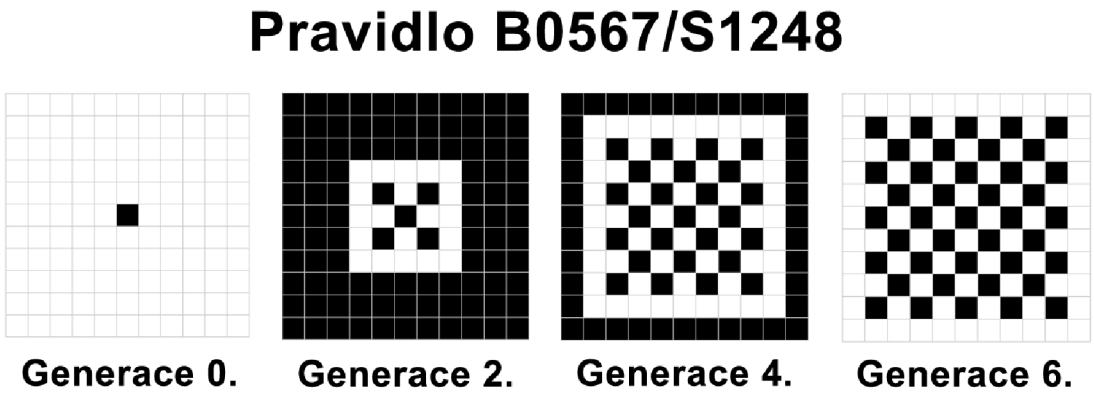
5.4 Šachovnicový vzor

Poslední hledanou charakteristikou je střídající se vzor, často také nazývaný „šachovnicový vzor“. Ten vykazuje emergentní chování, kdy se z jednoduchých pravidel vyvinou neočekávané a složité vzory, kde interakce na mikro úrovni (jednotlivé buňky a jejich sousedi) vedou k makro úrovni jevů (vzor na celé mřížce). Je hledán fitness funkcí *alternating_pattern*(). Stejně jako předchozí případ, je zde počáteční nastavení rovno jedné živé buňce na středu mřížky.

Je zde důležité podotknout, že stejně jako u předchozí varianty tak i zde jsou pravidla závislá na velikosti mřížky. Pokud se pravidlo hledané na mřížce 11×11 použije na mřížku s rozdílnými rozměry, chování v průběhu generací bude diametrálně jiné a nedojde k požadovanému výsledku. Proto jsou zde sestaveny tabulky obsahující příklady pravidel pro různá nastavení rozměrů mřížek, viz tab. 2, 3.

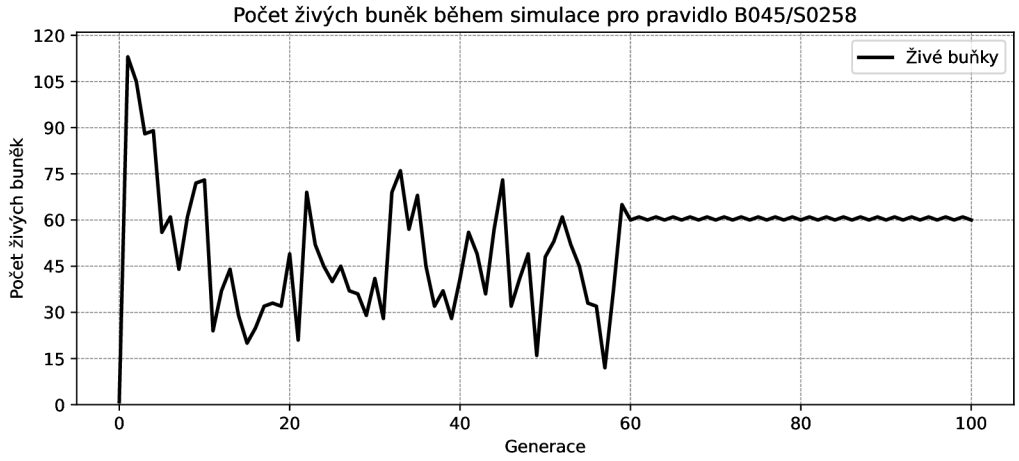


Obr. 34: Šachovnicový periodický vzor tvořený generovaným pravidlem.

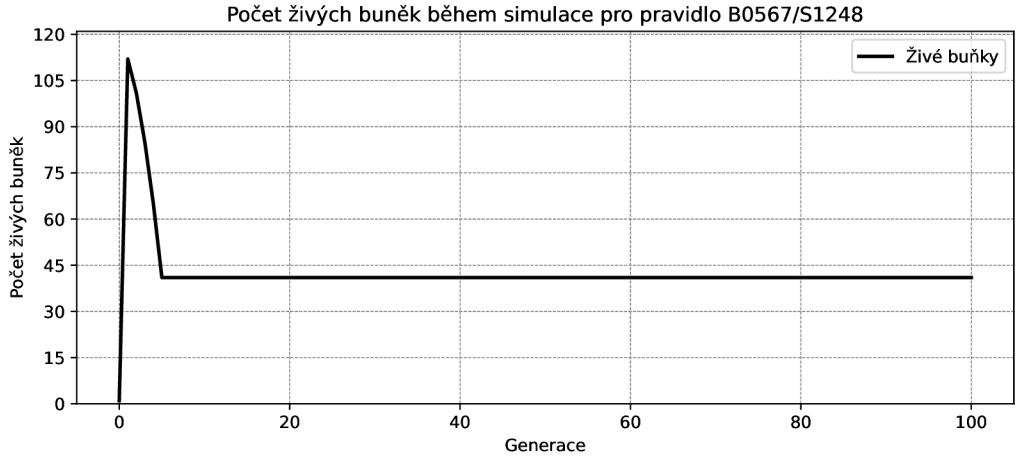


Obr. 35: Šachovnicový stálý vzor tvořený generovaným pravidlem.

Zde se pravidla dělí na dva typy. První typ se po určitém počtu iterací dostane do stavu, kdy cykluje mezi dvěma stavy šachovnicového vzoru do konce simulace, viz obr. 34. Druhý typ se ustálí na jenom šachovnicovém vzoru a ten přetrvává až do konce simulace, viz obr. 35. Tento jev lze pozorovat na vývoji živých buněk v průběhu generací v grafech 36, 37.



Obr. 36: Vývoj živých buněk v průběhu generací pro periodické pravidlo.



Obr. 37: Vývoj živých buněk v průběhu generací pro stálé pravidlo.

Pravidla generovaná GA byla testována na různých mřížkách, s různými nastaveními počátečních stavů a bylo vypozorováno, že počáteční stav jedné živé buňky na středu mřížky je nutným předpokladem pro dosáhnutí výsledného šachovnicového vzoru.

Periodické šachovnicové vzory

5 x 5	**6 x 6**	**7 x 7**	**8 x 8**
B145/S0267	B134/S0168	B045/S1256	B0246/S0136
B0145/S1267	B1245/S368	B0245/S256	B1246/S236

9 x 9	**10 x 10**	**11 x 11**	**12 x 12**
B145/S1256	B1248/S158	B045/S0258	B0468/S067
B12678/S012348	B0248/S0357	B012678/S3458	B0246/S156

Tab. 2: Nalezená pravidla periodických šachovnicových vzorů pro dané mřížky.

Stálé šachovnicové vzory

5 x 5	**6 x 6**	**7 x 7**	**8 x 8**
B1278/S347	B0257/S48	B0268/S345	B1257/S245
9 x 9	**10 x 10**	**11 x 11**	**12 x 12**
B1367/S3467	B0367/S0147	B0567/S1248	B02567/S467

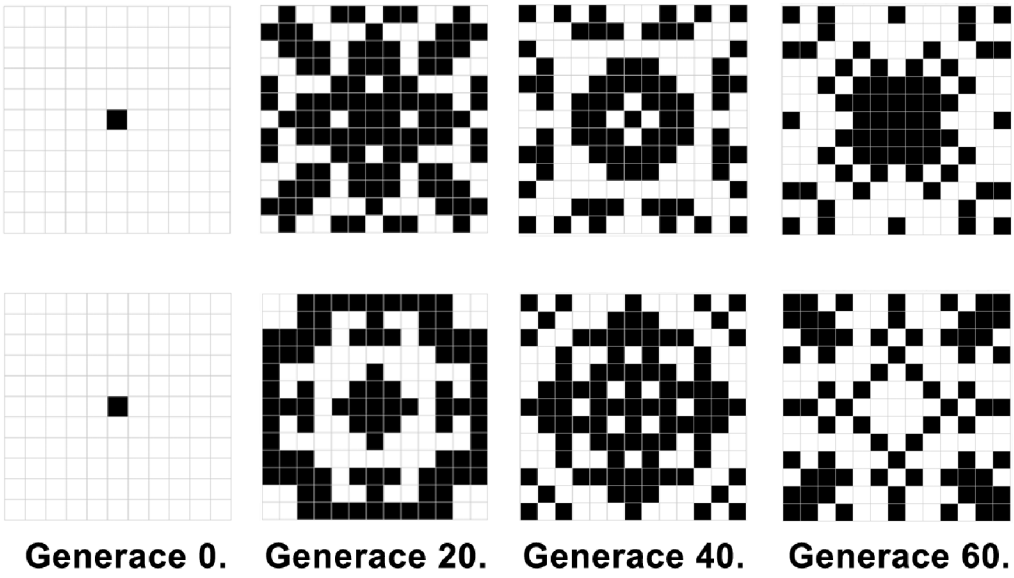
Tab. 3: Nalezená pravidla stálých šachovnicových vzorů pro dané mřížky.

Velmi zajímavým nálezem a výjimkou je pravidlo „B0567/S1248“, které lze vidět na obr. 35. To je schopno produkce šachovnicového stálého vzoru na liché čtvercové mřížce jakékoliv velikosti. Je ale nutno pracovat s periodickými okrajovými podmínkami. Tato schopnost se bohužel ztrácí při přechodu na čtvercovou mřížku sudou.

5.5 Výsledné zhodnocení

Genetický algoritmus dokáže efektivně hledat pravidla, která vyhovují požadovaným charakteristikám chování celulárního automatu. Délka průběhu je značně zkrácena díky paralelnímu simulování CA. Pro menší rozměry mřížek je efektivita GA větší, s roustoucí velikostí mřížky avšak klesá. To je způsobeno několika faktory:

- *Délka simulace* - Větší mřížky vyžadují více času na simulaci každé generace CA, protože každý krok musí být aplikován na větší počet buněk. To vede k delšímu času potřebnému k vyhodnocení fitness každého jedince v populaci genetického algoritmu.
- *Složitost vzorů* - Ve větších mřížkách mohou vznikat komplexnější a rozmanitější vzory. Tyto vzory mohou být obtížnější analyzovat a hodnotit, což vyžaduje sofistikovanější fitness funkce pro správné vyhodnocení jejich kvality.
- *Paměťové a výpočetní nároky* - Simulace větších mřížek vyžaduje více paměti a výpočetního výkonu. Paralelní simulace může tento problém částečně řešit, ale stále existují limity dané hardwarovými možnostmi.
- *Okrajové podmínky* - Při použití periodických okrajových podmínek, buňky na malé mřížce rychleji dosáhnou okraje a interagují s buňkami na protější straně. To může vést k rychlejším a méně předvídatelným změnám ve vzorcích chování buněk. Na větších mřížkách je tento efekt méně výrazný, což může vést k pomalejší evoluci a komplikovanějšímu vyhodnocování fitness funkcí.



Obr. 38: Symetrie pravidel B0234567/S47 a B023456/S048.

6 ZÁVĚR

Tato bakalářská práce se zabývala propojením celulárních automatů a genetických algoritmů, přičemž jejím cílem bylo zkoumat synergii mezi těmito dvěma oblastmi a vytvořit systém generování pravidel pro dvourozměrný celulární automat pomocí genetického algoritmu. Splněním prvních dvou bodů zadání, tedy seznámením se s CA a GA, jsem získal důležité informace o jejich aplikacích a možnostech využití v různých oborech.

Teoretická část práce se nejprve zaměřila na historii a základní principy celulárních automatů. Byly popsány různé druhy CA, včetně jednorozměrných, dvourozměrných a vícerozměrných celulárních automatů. Dále se zabývala klasifikací CA podle jejich chování a různými typy okrajových podmínek. V této části bylo také uvedeno několik příkladů využití CA, například model šíření epidemie a model růstu měst.

Další část rešerše se věnovala přírodním optimalizačním algoritmům, s důrazem na genetické algoritmy. Popis zahrnoval historii GA, jejich principy a klíčové komponenty, jako je genetická reprezentace problému, selekce, křížení a mutace. Tento přehled poskytl nezbytný teoretický základ pro praktickou část práce.

V praktické části byl navržen a implementován dvourozměrný celulární automat s pravidly generovanými genetickým algoritmem. Proces zahrnoval nastavení počátečního stavu mřížky, výpočet živých buněk v sousedství, aplikaci pravidel a simulaci CA. Návrh genetického algoritmu zahrnoval vytvoření počáteční populace jedinců, vyhodnocení fitness, proces selekce, křížení a mutace, a nakonec celkový proces evoluce.

Výsledky praktické části ukázaly, že genetické algoritmy jsou účinným nástrojem pro nalezení pravidel, která vedou ke vzniku zajímavého a komplexního chování celulárních automatů. S většími rozměry mřížek celulárního automatu přichází ale komplikace, jako je zpomalení procesu hledání a snížení pravděpodobnosti nalezení pravidla s požadovaným chováním.

Z celkového pohledu tato práce prokázala, že propojení celulárních automatů a genetických algoritmů nabízí široké možnosti pro výzkum. Každý z prezentovaných modelů může být dále rozvíjen a upravován, což umožňuje jejich aplikaci na složitější a komplexnější případy. Zajímavou cestou by pak mohlo být například sledování růstu života v horizontálních a vertikálních osách. Z uvedených výsledků a závěrů lze vyvodit, že kombinace CA a GA má potenciál při modelování a simulaci komplexních systémů.

SEZNAM POUŽITÉ LITERATURY

[1] Weisstein, E. W. *Cellular Automaton*. Dostupné z: https://mathworld.wolfram.com/.

[2] Wolfram, S. *A new kind of science*. Champaign, IL: Wolfram Media, 2002. ISBN 9781579550080.

[3] Sarkar, P. A brief history of cellular automata. *ACM Comput. Surv.* New York, NY, USA: Association for Computing Machinery. mar 2000, sv. 32, č. 1, s. 80–107. DOI: 10.1145/349194.349202. ISSN 0360-0300. Dostupné z: https://doi.org/10.1145/349194.349202.

[4] *The Making of A New Kind of Science*. Květen 2022. Dostupné z: https://writings.stephenwolfram.com/2022/05/the-making-of-a-new-kind-of-science/.

[5] Weisstein, E. W. *Elementary Cellular Automaton*. Dostupné z: https://mathworld.wolfram.com/.

[6] Packard, N. H. a Wolfram, S. Two-dimensional cellular automata. *Journal of Statistical Physics*. březen 1985, sv. 38, 5-6, s. 901–946. DOI: 10.1007/BF01010423. ISSN 0022-4715, 1572-9613. Dostupné z: http://link.springer.com/10.1007/BF01010423.

[7] Davies, P. 3D cellular automata. In: *Proceedings 5th Annual BPC Research Conference 5th July 2009*. Bournemouth, England: Bournemouth and Poole College, July 2009, č. 5. BPC Research Conference Proceedings. Proceedings 5th Annual BPC Research Conference 5th July 2009. Dostupné z: http://eprints.bournemouth.ac.uk/17408/.

[8] Gutowitz, H., ed. *Cellular automata: theory and experiment*. 1st MIT Press ed. Cambridge, Mass: MIT Press, 1991. Special issues of physica D. ISBN 9780262570862.

[9] Burzyński, M., Cudny, W. a Kosinski, W. K. Cellular automata: structures and some applications. *Journal of Theoretical and Applied Mechanics*. 2004, sv. 42, s. 461–482. Dostupné z: https://api.semanticscholar.org/CorpusID:55638228.

[10] Ahmed, E. a Elgazzar, A. On some applications of cellular automata. *Physica A: Statistical Mechanics and its Applications*. 2001, sv. 296, č. 3,

s. 529–538. DOI: https://doi.org/10.1016/S0378-4371(01)00182-0. ISSN 0378-4371. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0378437101001820.

[11] Dai, J., Zhai, C., Ai, J., Ma, J., Wang, J. et al. Modeling the Spread of Epidemics Based on Cellular Automata. *Processes*. 2021, sv. 9, č. 1. DOI: 10.3390/pr9010055. ISSN 2227-9717. Dostupné z: https://www.mdpi.com/2227-9717/9/1/55.

[12] Maithani, S. Application of cellular automata and GIS techniques in urban growth modelling: A new perspective. *Institute of Town Planners, India Journal*. 2010, sv. 7, č. 1, s. 36–49.

[13] McGrath, B., Sangawongse, S., Thaikatoo, D. a Barcelloni Corte, M. The architecture of the metacity: Land use change, patch dynamics and urban form in Chiang Mai, Thailand. *Urban Planning*. Cogitatio, Portugal. 2017, sv. 2, č. 1.

[14] Yang, X.-S. *Nature-inspired optimization algorithms*. Academic Press, 2020.

[15] Dorigo, M. Optimization, learning and natural algorithms. *Ph. D. Thesis, Politecnico di Milano*. 1992.

[16] Yang, X.-S. A new metaheuristic bat-inspired algorithm. In: *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, s. 65–74.

[17] Holland, J. H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992.

[18] Saravanan, N. a Fogel, D. B. A bibliography of evolutionary computation & applications. *Computers & Mathematics with Applications*. 1993, sv. 25, č. 5, s. 91–100.

[19] Sangwan, S. Literature Review on Genetic Algorithm. *International Journal of Research*. Červen 2018, sv. 5, s. 1142.

[20] Miček, D. Genetické algoritmy. říjen 2018. Dostupné z: http://hdl.handle.net/11012/11483.

[21] Obitko, M. Introduction to genetic algorithms. *Czech Technical University: Prague, Czech Republic*. 1998.

[22] Matoušek, R. *Elite tournament selection*. Brno: Brno University of Technology, 2005. ISBN 80-214-2961-5.

SEZNAM ZKRATEK A SYMBOLŮ

CA Celulární automat – Cellular automaton

GA Genetický algoritmus – Genetic algorithm

GIS Geografický informační systém – Geographical information system

ACO Optimalizace mravenčí kolonií – Ant colony optimization

BA Algoritmus netopýra – Bat algorithm

SEZNAM PŘÍLOH