VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MODIFIKACE GENETICKÝCH ALGORITMŮ PRO NÁVRH CELULÁRNÍCH AUTOMATŮ

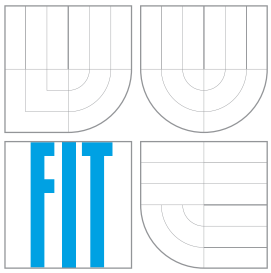
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MATEJ MAGDOLEN

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MODIFIKACE GENETICKÝCH ALGORITMŮ PRO NÁVRH CELULÁRNÍCH AUTOMATŮ

MODIFIED GENETIC ALGORITHMS FOR CELLULAR AUTOMATA DESIGN

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MATEJ MAGDOLEN
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL BIDLO, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce se zabývá evolučním návrhem přechodové funkce celulárního automatu řešícího zvolenou úlohu. Jsou v ní popsány celulární automaty, evoluční algoritmy a alternativní forma zápisu pravidel přechodové funkce vhodná pro evoluční návrh – podmínková pravidla. Dále je zvolen problém řešený celulárním automatem a prezentovány pokusy takový automat navrhnout genetickým algoritmem. Pokračuje se optimalizací parametrů algoritmu, hledáním jeho možných problémů a navržením modifikací řešících je. Pozitivní vlyv těchto modifikací je následně zhodnocen na několika experimentech.

Abstract

This bachelor's thesis aims to examine possibilities of designing a transition function enabling a cellular automaton to solve a given problem using a genetic algorithm. It contains an introduction to cellular automata, evolution algorithms and conditionally matching rules, a method of descripting a transition function suitable for evolutionary development. A set of experiments is conducted using the standard version of genetic algorithms to determine its optimal configuration. Additionally, modifications of this standard algorithm are proposed, effect of which on the algorithm's performance is then evaluated by further experiments.

Klíčová slova

celulární automaty, genetické algoritmy, modifikace, podmínková pravidla

Keywords

cellular automata, genetic algorithms, modification, conditionally matching rules

Citace

Matej Magdolen: Modifikace genetických algoritmů pro návrh celulárních automatů, bakalářská práce, Brno, FIT VUT v Brně, 2015

Modifikace genetických algoritmů pro návrh celulárních automatů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Bidla, Ph.D.

.....
Matej Magdolen
20. května 2015

Poděkování

Děkuji svému vedoucímu Ing. Michalovi Bidlovi, Ph.D za poskytnuté odborné vedení, konzultace, zkušenosti a projevenou trpělivost a vstřícnost během vypracování tohoto projektu.

Obsah

Kapitola 1

Úvod

Už od počiatkov existencie informačných technológii existoval záujem skúmať vlastnosti rôznych výpočtových modelov, často odlišných od zaužívaných. V dnešnej dobe, keď sa čoraz častejšie naskytá potreba riešiť problémy, u ktorých sa naráža na obmedzenia bežných postupov, sa stáva potreba alternatívnych výpočtových modelov stále aktuálnejšou.

Jedným z takýchto netradičných výpočtových modelov sú celulárne automaty. Podobne ako príbuzné prístupy, sú inšpirované prírodnými dejmi, ich chovanie sa podobá vzájomnej interakcii živých buniek. Ide tak o vhodný nástroj nie len na simuláciu podobných dejov, ale aj napríklad k vykonávaniu výpočtov.

Dôležitým princípom činnosti celulárnych automatov je komunikácia medzi bunkami, ktorá prebieha len na lokálnej úrovni, na napriek tomu môže na globálnej úrovni celého automatu vykazovať veľmi komplexné chovanie. Z toho plynú výhody, akými sú možnosť hardvérovej implementácie buniek celulárneho automatu umožňujúcej masívny paralelizmus a dobrú škálovateľnosť pridávaním ďalších jednotiek s bunkami.

Komunikácia len na lokálnej úrovni však sťažuje návrh implementácie požadovanej globálnej úlohy v prostredí automatu. Manuálny vývoj celulárneho automatu s požadovaným chovaním býva veľmi náročný. Preto je v prípade tohoto výpočtového modelu obzvlášť žiadúce umožniť automatizáciu vývoja. Jednou s použiteľných metód sú genetické algoritmy. Pri jej využití je kľúčové vhodne zvoliť reprezentáciu chovania automatu, aby ho bolo možné efektívne zdokonaľovať. Jednou z takýchto reprezentácii sú podmienkové pravidlá.

Motiváciou tejto práce je pokúsiť sa prispieť k zdokonaleniu parametrov procesu automatizovaného hľadanie riešení problémov celulárnymi automatmi a priblížiť tak reálnym podmienkam možnosť riešenia náročnejších problémov, než tomu bolo doteraz.

Cieľom práce je implementovať vývoj celulárneho automatu pomocou genetického algoritmu, pokúsiť sa navrhnúť a implementovať modifikácie tohoto algoritmu zlepšujúce priebeh evolúcie a overiť ich vplyv na zvolených úlohách.

Kapitola 2 popisuje problematiku celulárnych automatov, ich definíciu, vizualizáciu, kategórie a princípy, ktoré sa v nich vyskytujú. Kapitola 3 oboznamuje s metódami evolučného riešenia problémov, genetickými algoritmami, ich vlastnosťami a súčasťami. V kapitole 4 sú popísané podmienkové pravidlá, inovatívna možnosť zápisu chovania celulárneho automatu, ktorá bude využívaná pri ich evolučnom návrhu.

Kapitola 5 uvádza popis implementácie programu využívaného k experimentovaniu. V kapitole 6 je predvedený pokus vykonávajúci vývoj celulárneho automatu riešiaceho zvolenú úlohu pomocou genetického algoritmu. Je v nej preskúmaný vplyv vybraných parametrov genetického algoritmu na jeho vlastnosti. Ďalej sú identifikované niektoré potenciálne problémy genetického algoritmu a navrhnuté modifikácie za účelom ich riešenia. Ďalšími

pokusmi je preverený ich vplyv na výsledky evolúcie v hľadaní riešenia niekoľkých úloh.

Kapitola 7 poskytuje záverečné zhrnutie dosiahnutých výsledkov a návrh možností pokračovania práce.

Kapitola 2

Celulárne automaty

Celulárny automat (CA) je diskrétny výpočtový model pozostávajúci zo systému buniek, ktoré v diskrétnom čase menia stav. Bunky sú obvykle usporiadané do štvorcovej mriežky v jednom alebo dvoch rozmeroch. Je možné pridať tretí alebo viac rozmerov, alebo bunky iného tvaru usporiadať do iného druhu mriežky (napríklad šesťuholníky), ale tieto ďalšie varianty sú oproti základným málo zdokumentované a s malým množstvom aplikácií.

Každá bunka celulárneho automatu sa môže v danom čase nachádzať v niektorom z konečnej množiny stavov. Tie sa obvykle zapisujú číslom. Najjednoduchším automatom s ohľadom na množinu stavov je binárny automat s dvoma stavmi. Bunkám v stave 0 hovoríme mŕtve bunky, v stave 1 živé. Viacstavové automaty majú množinu stavov s väčšou kardinalitou.

Matica stavov celulárneho stavu v určitom čase sa nazýva konfigurácia. Prechod z času t do času $t + 1$ je krok automatu.

Pri vizualizácii automatov do podoby mriežky je vhodné číslo stavu zobrazovať farbou príslušnej bunky. Pri binárnych automatoch mávajú živé bunky farbu popredia (typicky čiernu) a mŕtve bunky farbu pozadia (bielu). U viacstavových automatov je potrebné použiť ďalšie farby.

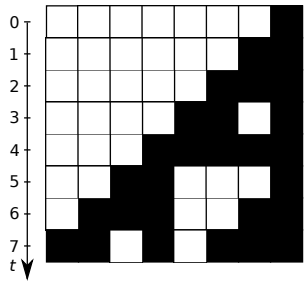
Zobrazenie dvojrozmerného automatu znázorňuje jeho konfiguráciu v určitom čase. U jednorozmerných automatoch sa používa forma, kde jeden obrázok zachytáva vývoj automatu v určitom časovom úseku, pričom jednotlivé stavy sú usporiadané pod seba. Čas je tak prenesený do druhého rozmeru vizualizácie.

2.1 Prechodová funkcia

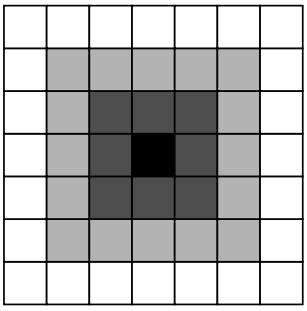
Nasledujúci stav bunky závisí od stavu jej okolia. Predpisom, podľa ktorého sa táto zmena vyhodnocuje, je prechodová funkcia. V prípade synchrónneho automatu sa pri tvorbe no-

(a) binárny (b) viacstavový

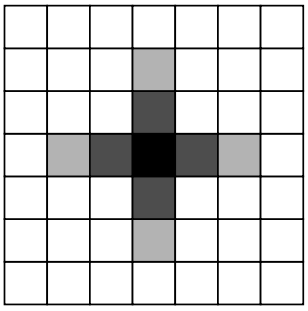
Obrázek 2.1: Vizualizácia dvojrozmerného celulárneho automatu



Obrázek 2.2: Vizualizácia jednorozmerného celulárneho automatu

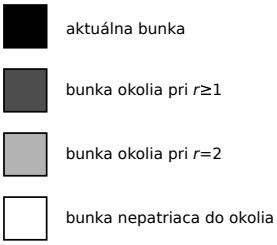


(a) Moorovo okolie v dvojrozmernom automate



(b) Von Neumannovo okolie v dvojrozmernom automate

(c) jednorozmerný automat



(d) legenda

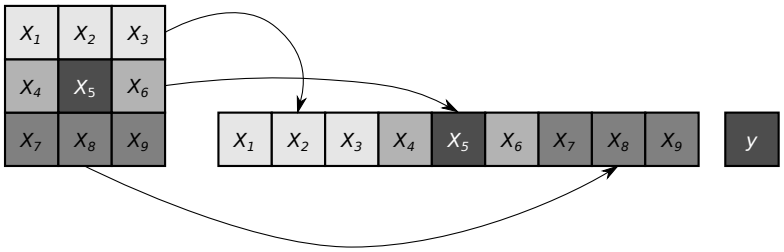
Obrázek 2.3: Okolie bunky pri polomeroch $r = 1$ a $r = 2$

vého stavu v čase $t + 1$ vychádza zo stavu buniek v čase t, ktorý sa teda v priebehu vyhodnotenia nového stavu nemení. Naproti tomu asynchrónny automat využíva nové, v priebehu aktuálneho prechodu nastavené hodnoty, čím sa jeho vývoj stáva nedeterministickým, závislým od implementácie prípadného priebehu paralelných procesov.

Okolie bunky, ktoré tvorí vstup prechodovej funkcie má podobu usporiadanej množiny. Jej mocnosť závisí od počtu dimenzií konečného automatu a polomeru okolia, čo je celočíselná hodnota určujúca maximálnu vzdialenosť bunky (rozdiel súradníc) od aktuálnej bunky, pri akej ešte patrí do jej okolia. Pri jednorozmernom automate sa používa 3-okolie ($r = 1$) a 5-okolie ($r = 2$).

V dvojrozmernom automate sa navyše rozlišuje Von Neumannovo okolie (pri $r = 1$ 5-okolie) a Moorovo (pri $r = 1$ 9-okolie). V prípade Von Neumannovho okolia sa za susedné bunky považujú len tie so spoločnou hranou, u Moorovho postačuje spoločný vrchol.

Častým spôsobom zápisu prechodovej funkcie je forma tabuľky obsahujúcej $n+1$ stĺpcov, kde n je počet buniek okolia a každý riadok obsahuje konfiguráciu stavov okolia, ktorej sa týka, a končí novým stavom, do ktorého v takom prípade aktívna bunka prechádza. Kombinácie, pri ktorých sa stav aktívnej bunky nemení sa do tabuľky nezapisujú.



Obrázek 2.4: Vzťah okolia bunky a prechodovej funkcie

X_1	X_2	X_3	y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Tabulka 2.1: Prechodová funkcia binárneho jednorozmerného automatu zapísaná do tabuľky

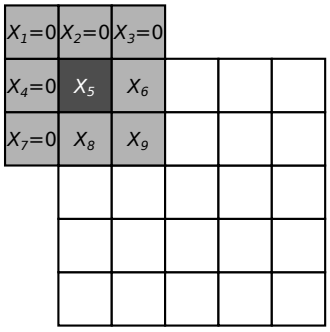
Každá kombinácia sa v tabuľke vyskytuje maximálne jedenkrát. Riadky tabuľky aj poradie v akom sú bunky okolia zoradené do stĺpcov tak môžeme ľubovoľne preusporiadať bez zmeny prechodovej funkcie. Je však konvenciou postupovať po okolí smerom zhora dole a zľava doprava, takže aktívna je v prostrednom stĺpci časti tabuľky popisujúcej okolie. Riadky tabuľky sa zoraďujú podľa číselnej hodnoty postupnosti stavov okolia.

V jednorozmernom automate s 3-okolím existuje $2^3 = 8$ variácií stavov buniek okolia. Pokiaľ pre všetky definujeme nasledujúci stav aktívnej bunky a dodržíme konvenciu o poradí riadkov a stĺpcov tabuľky, je prechodová funkcia jednoznačne určená posledným stĺpcom tabuľky obsahujúcim nový stav. Prechodové funkcie tak môžeme značiť postupnosťou nových stavov, alebo len desiatkovou hodnotou takejto postupnosti. Toto značenie zaviedol Stephen Wolfram, keď sa zaoberal jednorozmernými binárnymi automatmi značenými číslami 0 až 255.

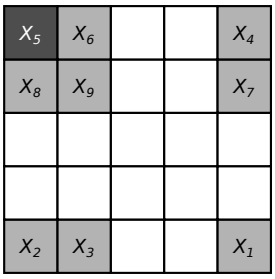
Hoci môže mať mriežka buniek automatu teoreticky neobmedzenú veľkosť, pri implementácií je vhodné definovať jej rozmery. Komplikáciou takéhoto opatrenia je, že okolie buniek na okraji mriežky zasahuje mimo nej a vzniká tak otázka, ako vyhodnocovať ich nový stav. Riešením býva stanovenie okrajových podmienok. Najčastejším druhom bývajú:

- nulové – stav buniek v časti okolia zasahujúcej mimo mriežky je pevne nastavený na nulu, akoby bola mriežka ohraničená trvalo mŕtvymi bunkami a väčšia o dvojnásobok polomeru okolia,
- cyklické – do okolia sa doplnia stavy buniek z opačnej strany mriežky, akoby sa mriežka opakovala [11].

Všetky bunky uniformného celulárneho automatu nadobúdajú stav s rovnakej množiny stavov a prechody medzi ich stavmi sa riadia jednou prechodovou funkciou. V prípade neuniformného automatu sa môže jedna z týchto vlastností alebo obe líšiť pre jednotlivé bunky alebo ich skupiny.



(a) nulové



(b) cyklické

Obrázek 2.5: Okrajové podmienky

Najbežnejším prípadom sú však synchrónne uniformné celulárne automaty, ktorými sa bude zaoberať táto práca.

2.2 História

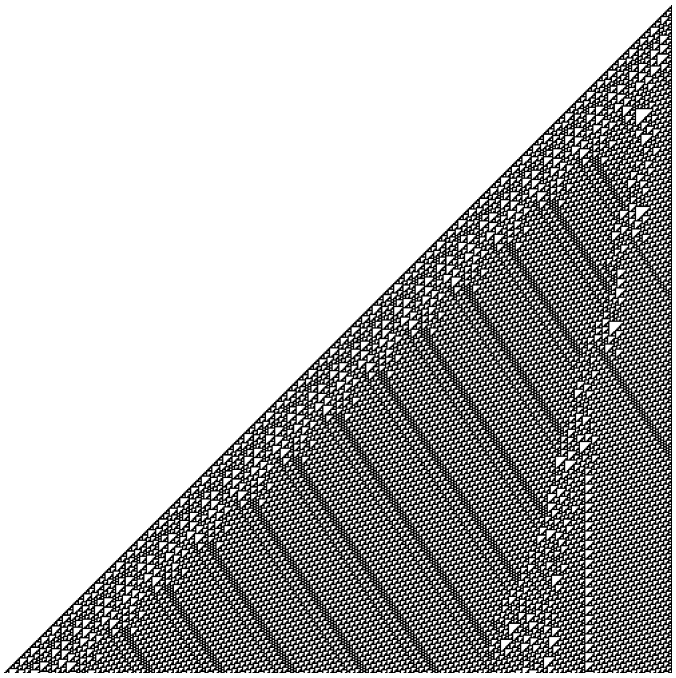
Autormi prvej verzie celulárneho automatu sú Stanislaw Ulam and John von Neumann[10], ktorí v 40. rokoch 20. storočia skúmali sebareplikujuce organizmy a systémy. Von Neumann navrhol univerzálny replikátor – konštrukčné rameno pozostávajúce z 200000 buniek a 29 stavov, pracujúce podľa inštrukcií na páske. Bolo dokázané, že tento systém je výpočtovo úplný. Edgar F. Codd zredukoval univerzálny replikátor na osem stavov.

Ďalším zjednodušením replikátora je Langtonova slučka, ktorá už nie je univerzálna, ale zaberá len 86 buniek [9]. Jej jadro rotuje, pričom dochádza k vytváraniu výhonku, ktorý podľa inštrukcií zakódovaných do buniek jadra rastie a zakrivuje sa, až sa vyvinie do podoby novej slučky. Následne sa pôvodná aj nová slučka replikujú do voľného miesta v automate. Postupnou redukciou Langtonovej slučky odstraňovaním vnútorného a vonkajšieho zapúzdrenia jadra vznikli slučky Byleova a Chou-reggia.

John Horton Conway v roku 1970 navrhol celulárny automat Hra života (Game of Life) modelujúci chovanie živej kultúry mikroorganizmov [5]. Ide o dvojrozmerný binárny automat, ktorého prechodová funkcia je zjednodušená – jej vstupom nie sú stavy buniek okolia ale len počet živých buniek v okolí.

Prechodová funkcia pôvodnej Hry života sa označuje ako B3/S23. Čísla za lomkou sa vzťahujú k živej bunke, ktorá pri dvoch alebo troch živých bunkách v okolí prežije do ďalšieho kroku. Pri menšom počte zomrie na osamelosť, pri väčšom naopak na premnoženie. Prvá časť označenia definuje, že ak sú v okolí mŕtvej bunky práve tri živé bunky, dôjde k reprodukcii a bunka ožije.

Napriek svojej jednoduchosti tento automat vykazuje komplexné chaotické chovanie a v závislosti od počiatočnej konfigurácie sa v ňom vytvárajú štruktúry, či už statické, oscilátory cyklujúce medzi niekoľkými rôznymi podobami alebo klzáky pohybujúce sa po bunkách automatu[12]. Toto viedlo k záujmu o celulárne automaty, hľadanie zaujímavých počiatočných konfigurácií, či tvorbu iných variantov prechodovej funkcie.



Obrázek 2.6: Prvých 500 krokov pravidla 110 inicializovaného jednou živou bunkou [13]

2.3 Základné celulárne automaty

Stephen Wolfram skúmal dvojrozmerné binárne automaty automaty s trojokolím s nekonečným počtom buniek. Zaviedol pre ne vyššie vysvetlené značenie číslami od 0 po 255. Takéto automaty sa nazývajú Základné celulárne automaty. Skúmaním chovania rôznych pravidiel pri rôznych počiatočných stavoch dospel Wolfram k rozdeleniu na 4 triedy[13]:

- Trieda I – automaty, ktoré postupom času dospejú do stabilnej a homogénnej konfigurácie pre takmer všetkých počiatočných stavoch,
- Trieda II – spravidla sa časom ustália do podoby periodicky sa meniacich a opakujúcich vzorov,
- Trieda III – neprestajne sa ich stav chaoticky mení aj pri jednoduchých počiatočných konfiguráciách,
- Trieda IV – kombinácia vlastností predošlých dvoch tried, vyvíjajú sa lokálne štruktúry, ktoré môžu dlhodobo prežívať, alebo sa zložito vzájomne ovplyvňovať.

Jedným z obzvlášť zaujímavých základných celulárnych automatov je takzvané Pravidlo 110 (Rule 110). Ide o automat patriaci do Triedy IV, je možné v ňom pozorovať tri rôzne interagujúce štruktúry[13]. Bola dokázaná turingovská úplnosť tohto automatu[4].

2.4 Možnosti aplikácie

K možným využitiam celulárnych automatov patrí modelovanie a simulácie. Existujú prírodné deje, ktoré fungujú ako celulárny automat, alebo je možné ich takto aproximovať.

Ďalej je možné automaty využiť ako výpočtový model, čo umožňuje využiť paralelizmus alebo automat fyzicky implementovať súborom nezávislých lokálne komunikujúcich výpočtových jednotiek s možnosťou škálovateľnosti riešenia, či čiastočnej odolnosti voči poruchám. Problémom, ktorý sa pri vývoji takýchto automatov rieši je, že zatiaľ, čo bunky automatu komunikujú len lokálne, riešený problém býva obyčajne globálny. Zjednodušenie riešení môže byť uskutočniteľné využitím neuniformného automatu.

Kapitola 3

Evolučné algoritmy

Evolučné algoritmy je súhrnné označenie kategórie algoritmov, ktoré hľadajú riešenie problému stochaistickým prehľadávaním stavového priestoru typicky s využitím prírodou inšpirovaných princípov. Je možné ich použiť na tvorbu nových riešení ako aj na zlepšovanie, či dokončovanie existujúcich úplných alebo čiastočných. Charakteristickým znakom výsledkov takýchto postupov je, že nevykazujú znaky bežné pre riešenia nachádzané systematickými postupmi a môžu sa tak stať pre človeka ťažko pochopiteľné.

Základným princípom spoločným s evolúciou v prírode je, že silnejší, lepší jedinci – lepšie riešenia majú väčšiu šancu množiť sa a zabezpečiť tak prenos svojej genetickej informácie do ďalších generácii[6]. V populácii sa tak šíri informácia o vlastnostiach, ktoré ich lepšími spravili. Z toho vyplýva nutnosť porovnávať kvalitu riešení, a to nie len riešenia správne s nesprávnymi, ale aj čiastočné riešenia medzi sebou, aby bolo možné odhadnúť, ktoré sa viac blíži k požadovanému. K tomuto v evolučných algoritmoch slúži fitness funkcia, ktorá priradí každému jedincovi populácie číselnú hodnotu fitness vyjadrujúcu, do akej miery jedinec spĺňa požiadavky kladené na riešenie. S ohľadom na ňu potom prebiehajú operácie selekcie, kríženia a mutácie využívané pri tvorbe novej generácie jedincov[8].

3.1 Fenotyp a genotyp

Každý jedinec, ktorý sa môže vyskytnúť v populácii je charakterizovaný súhrnom jeho vonkajších znakov. Tie určujú jeho vlastnosti pri interakcii s okolím, jeho schopnosť obstáť pri riešení problému. Formálne ich môžeme zaznamenať ako usporiadanú množinu parametrov, ktoré definujú jeho umiestnenie v prehľadávanom stavovom priestore. Takýto vonkajší parameter jedinca sa nazýva alela. V prírode ide napríklad o farbu očí, pri výpočtoch môže ísť o jeden argument funkcie. Súhrn všetkých alel jedinca je jeho fenotyp.

Fenotyp jedinca výsledkom aplikácie akéhosi predpisu na jeho vytvorenie. V prírode je tento predpis obsiahnutý v chromozómoch bunkových jadier. Aj v evolučných algoritmoch sa používa vnútorná reprezentácia fenotypu, ktorá musí byť naň prevedená podľa určeného kódovania. Položka takejto reprezentácie zodpovedajúca fenotypu jedinca je chromozóm, ktorého hodnota je genotyp. Chromozóm sa obvykle skladá z viacerých častí – génov.

Hlavne typom chromozómu a fenotypu sa líšia druhy evolučných algoritmov.

- Genetické programovanie – fenotypom je program premenlivej dĺžky, u ktorého je snaha prispôsobiť ho čo najlepšie danej úlohe. Reprezentovaný môže byť napríklad stromom, grafom či lineárne.

- Evolučné programovanie – program má pevnú dĺžku a štruktúru, algoritmus sa snaží optimalizovať jeho parametre.
- Evolučné stratégie – jedinec je charakterizovaný reálnymi číslami. Mutácia prebieha typicky sčítaním chromozómu s rovnako dlhým vektorom s reálnymi číslami vygenerovanými podľa normálneho rozdelenia pravdepodobnosti. Kríženie môže byť diskrétne, podobne ako u genetických algoritmov, alebo uskutočnené napríklad vyrátaním aritmetického priemeru zodpovedajúcich génov rodičov.
- Genetické algoritmy – fenotyp tvoria celé čísla alebo diskrétne vlastnosti, ktoré je možné celými číslami vyjadriť. Hlavne nim sa bude ďalej venovať táto práca.

Voľba kódovania fenotypu do chromozómu vhodného pre riešenú úlohu je dôležitým krokom v návrhu evolučného algoritmu. Ovplyvňuje chovanie algoritmu pri evolúcii, môže meniť rozsah stavového priestoru a je jej potrebné prispôsobiť implementáciu evolučných operácii. V prípade genetického algoritmu sa často využíva binárna reprezentácia. Usporiadaná množina celočíselných argumentov je vtedy vyjadrená reťazcom bitov potrebnej dĺžky. Obvykle platí, že binárny chromozóm je možné rozdeliť na podreťazce, z ktorých každý zodpovedá jednej alele.

Inou možnosťou sú gény v podobe čísla v desiatkovej sústave. Alely môžu v nich byť vyjadrené priamo, alebo môže jeden dekadický gén zodpovedať viacerým, či naopak, alela viacerým génom.

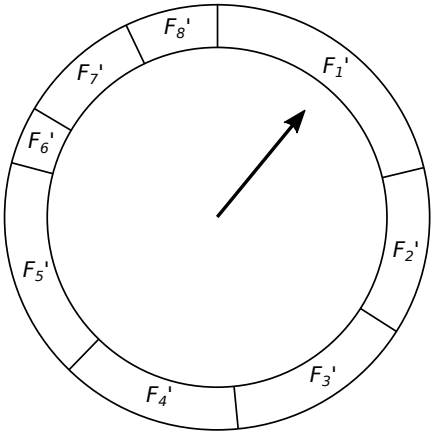
Špecifickejším riešením je permutačné kódovanie. Využíva sa napríklad pri kombinatorických alebo plánovacích úlohách. Každá hodnota génu sa v takto zakódovanom chromozóme vyskytuje práve raz. Napríklad pri hľadaní riešenia problému obchodného cestujúceho genetickým algoritmom možné hodnoty génu zodpovedajú jednotlivým mestám, ktoré sú teda navštívené v takom poradí, v akom sa vyskytujú v chromozóme. Výhodou je, že nakoľko sa hodnoty génov v rámci chromozómu nemôžu opakovať, vylúči sa zo stavového priestoru množstvo ciest, ktoré nezodpovedajú zadaniu, pretože sa rovnaké mesto navštívi viackrát. Algoritmus sa tak môže o niečo lepšie zamerať na samotnú optimalizáciu cesty.

3.2 Evolučné operácie

Operátor selekcie napodobňuje prírodný výber, zabezpečuje lepším jedincom väčšou šancu prežiť do nasledujúcich generácii a množiť sa. Vo všeobecnosti funguje tak, že z populácie náhodne vyberie jedinca, pričom pravdepodobnosť výberu je tým vyššia, čím má jedinec väčšiu hodnotu fitness. Konkrétna pravdepodobnosť sa však pri rovnakej populácii líši podľa spôsobu implementácie.

Ruletový výber je jednou z podôb operátoru selekcie. Je podobný hazardnej hre ruleta, kde pravdepodobnosť výhry zodpovedá veľkosti stredového uhla kruhového výseku prislúchajúceho zvolenej stávke. Platí v nej teda, že šanca, že guľôčka sa zastaví napríklad na niektorom z červených čísel, ktorých kruhové výseky by, ak by ležali vedľa seba, mali stredový uhol takmer 180°, je väčšia, než že skončí na nule, ktorej výsek má stredový uhol $\frac{360°}{37}$. Pomer týchto pravdepodobností je pritom rovný pomeru stredových uhlov. Ruletový výber prideľuje každému jedincovi kruhový výsek so stredovým uhlom zodpovedajúcim jeho fitness hodnote.

Selekcia usporiadaním zotriedi jedincov populácie podľa hodnoty fitness a výsledok náhodne vyberie s tým väčšou pravdepodobnosťou, čím lepšie sa jedinec umiestni v poradí.



Obrázek 3.1: Ruletový výber z populácie obsahujúcej 8 jedincov

Táto závislosť môže byť lineárna alebo napríklad exponenciálna. Rozdielom oproti ruletovému výberu je, že sa nezohľadňuje rozdiel hodnôt fitness, len ich poradie, takže dve miesta v usporiadaní majú rovnaký pomer pravdepodobnosti výberu, bez ohľadu na hodnoty fitness.

Jedným zo spoločných znakov spôsobov selekcie je ich náhodnosť. Z tej vyplýva, že existuje možnosť, že sa do ďalšej generácie dostanú len tie najhoršie jedince alebo sa po mnoho generácii postupne zlepšované riešenie, potenciálne veľmi maximálnej fitness hodnote z populácie stratí. V prípadoch, keď je takáto možnosť nevhodná, možné zabezpečiť monotónnosť evolučného algoritmu použitím tzv. elitizmu. Ten spôsobuje, že je najlepší jedinec populácie vždy bez zmeny prenesený do novej generácie, a nemôže sa tak stratiť.

Turnajová selekcia najprv náhodne vyberie z populácie k jedincov a z nich toho s najvyššou hodnotou fitness. Hodnota k sa nazýva veľkosť turnaja. Výhodou oproti selekcii usporiadaním je, že nie je potrebné jedincov časovo náročne zotrieďovať.

3.2.1 Kríženie

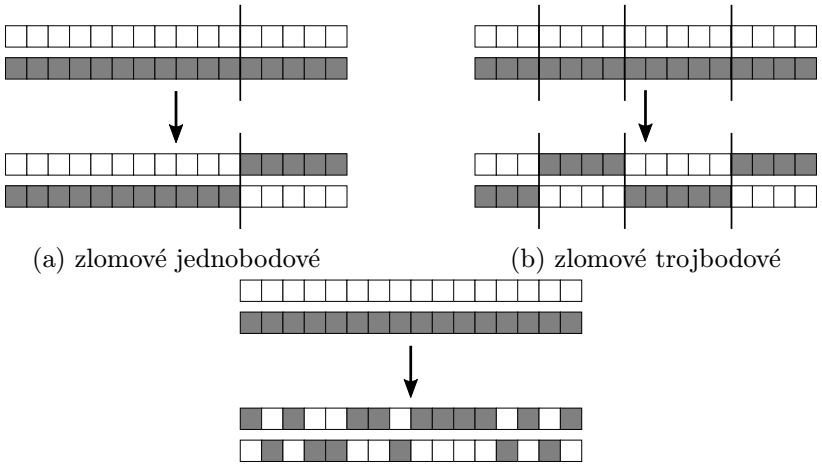
Operátor kríženia (niekedy tiež rekombinácie) má obvykle za vstup dvoch rodičov a výstupom sú dvaja potomkovia, pričom každý z nich obsahuje gény oboch rodičov. Úlohou kríženia by malo šíriť po populácii čiastkové informácie o lepších riešeniach. V niektorých prípadoch je však problémom, že vzniknuté kombinácie častí jedincov nedávajú dokopy zmysel a zbytočne sa tak rozbíjajú lepšie riešenia.

Zlomové kríženie prebieha tak, že sa náhodne určí pozícia zlomu, čo je miesto, kde sa vymenia rodičia génov odovzdávajúci gény potomkom. Pokiaľ je takýto zlom jeden, ide o kríženie jednobodové, pri viacerých hovoríme o viacbodovom krížení[8].

Ďalším druhom je uniformné kríženie, ktoré nevyužíva zlomy ale pre každý gén sa náhodne s danou pravdepodobnosťou určí, či sa bude krížiť, teda, či sa z rodičov do potomkov prekopíruje v pôvodnom poradí alebo naopak.

3.2.2 Mutácia

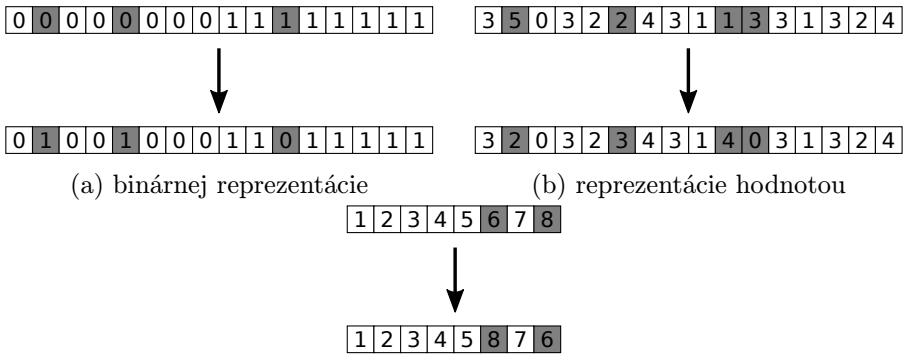
Operátor mutácie stochaistickým spôsobom mení obsah chromozómu [1]. Do populácie sa tým vnáša nová genetická informácia, ktorá má následne, ak sa ukáže jej dobrý vplyv na riešenie problému, šancu zostať a rozšíriť sa po populácii. Zväčšuje sa tým prehľadaná



(a) zlomové jednobodové (b) zlomové trojbodové

(c) uniformné

Obrázek 3.2: Kríženie



(a) binárnej reprezentácie (b) reprezentácie hodnotou

(c) permutačnej reprezentácie

Obrázek 3.3: Mutácia

časť stavového priestoru. To, ktoré gény chromozómu budú zmutované sa určuje náhodne, pričom pravdepodobnosť musí byť vhodne zvolená. Býva malá, aby príliš veľký počet mutácii nepribližoval evolučný algoritmus náhodnému prehľadávaniu a nerobil tak jeho použitie zbytočným. Na druhej strane, ak je zvolená príliš nízka pravdepodobnosť, vplyv mutácie sa vytráca a algoritmus neprehľadá dostatočne veľkú časť stavového priestoru. Optimálne nastavenie je pritom rôzne v závislosti od riešenej úlohy, veľkosti chromozómu či použitej reprezentácie.

Od reprezentácie závisí aj samotná implementácia mutácie. Pri binárnej reprezentácii sa obvykle zneguje hodnotu príslušného génu. V prípade číselnej reprezentácie môže byť hodnota mutovaného génu nahradená nanovo vygenerovanou hodnotou. V permutačnom kódovaní dôjde k výmene dvoch náhodne zvolených génov.

Kapitola 4

Podmienkové pravidlá

Ako bolo popísané v kapitole 2, základným spôsobom, akým je možné popísať prechodovú funkciu celulárneho automatu sú tabuľkové pravidlá. Každý riadok tabuľky v tomto zápise zodpovedá jednej konkrétnej konfigurácii okolia bunky, teda jednej variácií stavov buniek okolia. Pre každý takýto riadok tabuľka predpisuje nasledujúci stav aktívnej bunky, do ktorého bunka prejde, ak jej okolie bude v konfigurácii zodpovedajúcej riadku. Prípady, keď sa stav aktívnej bunky nezmení je možné z tabuľky vynechať.

Počet variácií stavov buniek okolia, ktoré sa môžu v tabuľke vyskytnúť je rovný n^k, kde n je kardinalita stavovej množiny a k počet buniek okolia. Pre jednorozmerný binárny automat s trojokolím tak môže mať tabuľka maximálne $2^3 = 8$ riadkov, pre dvojrozmerný s Moorovým okolím $2^9 = 512$, ak zvýšime počet stavov na 4, je to už $4^9 = 262144$ riadkov.

Je teda vidno, že pri návrhu riešenia komplikovanejšej úlohy môže byť tabuľkových pravidiel potrebné značné množstvo. Obzvlášť problematické sú tieto vlastnosti pri snahe o evolučný návrh prechodovej funkcie. Jedným z faktorov je, že sa prehľadáva značne rozsiahly stavový priestor. Možností, ako priradiť nasledujúce stavy 512 riadkom tabuľky je 2^{512}. Okrem toho je možné, že časť pravidiel vygenerovaných evolučným algoritmom sa ani nevyužije, pretože daný stav okolia nebude nastávať, čím sa stáva ťažšie urobiť v predpise zmysluplnú zmenu. V komplexnejších úlohách je pravdepodobné, že na dosiahnutie istého požadovaného čiastkového chovania bude potrebná súhra viacerých riadkov tabuľky, čo môže spôsobovať problém genetickým operátorom, ktoré takéto súvislosti nerešpektujú.

Uvedené nevýhody tabuľkových pravidiel viedli k snahe navrhnúť vhodnejší a úspornejší spôsob zápisu prechodovej funkcie. Jednou zo skúmaných možností bola reprezentácia založená na inštrukciách, kde okolie aktívnej bunky a jej nový stav sú vstupom a výstupom programu [2].

4.1 Popis

Ďalšou formou sú podmienkové pravidlá (anglicky condition matching rules), ktoré prinášajú isté zovšeobecnenie tabuľkových pravidiel [3]. Tie sú aplikované, keď sa konfigurácia okolia rovná stavom zapísaným v pravidle. Podmienkové pravidlá miesto tohoto implicitného operátoru `==`, prinášajú možnosť zvoliť pre každý stĺpec tabuľky z výberu funkcii. Najpoužívanejšími sú binárne operátory „==“,„!=“,„¿=“,„¡=“ a unárne „*“, negácia a identita, zväčša sa používa ich podmnožina. Dvojica funkcie a operandu pre každú bunku okolia tvorí podmienkovú časť, okrem ktorej podmienkové pravidlo obsahuje nasledujúci stav v prípade jeho aplikácie. Za druhý operand sa dosadzuje stav príslušnej bunky okolia.

Podmienkové pravidlo môže byť aplikované, ak platia všetky podmienky uvedené v jeho podmienkovej časti. Vzhľadom k tomu, že jednotlivým funkciám môže vyhovovať viacero stavov, než je tomu pri testovaní na rovnosť, môže podmienkové pravidlo vyjadrovať viacero tabuľkových pravidiel. Dochádza tak k skráteniu zápisu prechodovej funkcie.

Celá prechodová funkcia celulárneho automatu je vyjadrená usporiadanou množinou podmienkových pravidiel. Vzhľadom k všeobecnosti, ktorú forma podmienkových pravidiel umožňuje je pravdepodobné, že sa v množine vyskytne pre niektoré konfigurácie okolia viacero vyhovujúcich pravidiel. Aby bola zachovaná deterministickosť vývoja automatu, testuje sa aplikovateľnosť podmienkových pravidiel na aktuálne okolie v poradí, v akom sú uvedené. Použije sa ďalší stav z prvého nájdeného vyhovujúceho pravidla. Zvyšné pravidlá tak už nemusia byť kontrolované. Preto je pre podobu prechodovej funkcie dôležitý nie len obsah, ale aj poradie podmienkových pravidiel.

Popísaná vlastnosť umožňuje napríklad obmedziť časť platnosti neskoršieho podmienkového pravidla umiestnením konkrétnejšie definovaného pravidla na prednejšiu pozíciu v usporiadaní. Možnou nevýhodou je, že podmienkové pravidla, ktorých všetky možnosti aplikácie už boli pokryté skôr uvedenými pravidlami sa stávajú zbytočnými. Pokiaľ sa napríklad blízko začiatku usporiadanej množiny vyskytne pravidlo obsahujúce viacero funkcii `*`, bude takých väčšina pravidiel.

Prechodová funkcia predpísaná podmienkovými pravidlami býva kratšia než tabuľková podoba, ale vyjadrovacie schopnosti týchto dvoch foriem sú ekvivalentné. Je teda medzi nimi možné zápis obojsmerne prevádzať. Platí, že každú usporiadanú množinu podmienkových pravidiel je možné previesť na práve jednu tabuľku. V opačnom smere prevodu ale tabuľke zodpovedá viacero zápisov podmienkovými pravidlami. Najjednoduchší prevod týmto smerom, ak je dostupná funkcia `==`, je doplniť ju do všetkých stĺpcov popisujúcich stav okolia v tabuľke.

Pri opačnom prevode sa pre všetky riadky tabuľky vyhľadá obvyklým postupom zodpovedajúce podmienkové pravidlo a v ňom uvedený ďalší stav. Prípadne môže byť vhodné ukladať len riadky tabuľky skutočne využité pri simulácii celulárneho automatu.

Pri použití podmienkových pravidiel bola zistená lepšia schopnosť genetických algoritmov hľadať riešenia problémov než u iných foriem, napríklad evolúcii tabuľkových pravidiel. Takémuto využitiu sa bude venovať aj táto práca.

Kapitola 5

Implementácia

Hlavná časť projektu bola implementovaná v jazyku C++ podľa normy C++11, takže prebehol objektovo orientovaný návrh, kde populácia genetického algoritmu, celulárny automat a ich časti zodpovedajú triedam v programe. Využité boli štandardné knižnice, ktoré sú súčasťou jazyka.

K vizualizácii vývoja celulárnych automatov sa používal program BiCAS, ktorého autorom je Ing. Michal Bidlo, Ph.D. Oproti pôvodnej verzii bola do programu doplnená podpora nulových okrajových podmienok, Moorovho okolia, vizualizácie jednorozmerného celulárneho automatu a boli upravené farby výstupu. Tento program a niektoré pomocné skripty využívané pri experimentovaní boli implementované v jazyku Python. Pri tvorbe technickej správy sa využíval program LaTeX, na kreslenie obrázkov Inkscape.

5.1 Celulárny automat

Pri návrhu celulárnych automatov genetickým algoritmom fitness hodnota jedincov spravidla zodpovedá miere, akou konfigurácia zodpovedajúceho automatu po nejakom počte krokov zodpovedá zadaniu. Bolo teda potrebné implementovať prevod genotypu chromozómov na fenotyp prechodovej funkcie celulárneho automatu (počiatočný stav bol daný zadaním úlohy) a simuláciu vývoja automatu. Vo vytvorenom programe je možné simulovať synchrónne uniformné automaty. Na výber sú dvojrozmerné automaty obdĺžnikového tvaru s nastaviteľnými rozmermi a počtom stavov, Von Neumannovým alebo Moorovým okolím. Taktiež sa dajú simulovať jednorozmerné automaty.

Okrajové podmienky je možné zvoliť nulové alebo cyklické, čo je implementované zväčšením poľa uchovávajúceho konfiguráciu automatu o položky na okrajoch, kde sú počas simulácie udržované zodpovedajúce hodnoty stavov, ale nie sú vyhodnocované ani pri iných operáciách považované za súčasť automatu.

5.2 Podmienkové pravidlá

Ako spôsob zápisu prechodovej funkcie CA sa využívajú podmienkové pravidlá. Pri ich objektovej reprezentácii bol využitý princíp kompozície. Usporiadaná množina podmienkových pravidiel vyjadrujúca prechodovú funkciu je uložená ako pole podmienkových pravidiel. Podmienkové pravidlo je zložené z nového stavu a poľa inštancii triedy podmienky s dĺžkou zodpovedajúcou použitému okoliu. Tie uchovávajú zvolenú podmienkovú funkciu a

hodnotu, a obsahujú metódu, ktorá pre vstup, ktorým je stav bunky okolia, zistí, či podmienka platí alebo nie. Túto metódu zas využíva metóda podmienkového pravidla, ktorá testuje platnosť konjukcie platnosti všetkých jej podmienok a za vstup má celú konfiguráciu okolia aktuálnej bunky. Na najvyššej úrovni je v poli podmienkových pravidiel hľadané prvé, ktoré platí pre aktuálne okolie. Z neho je potom zistený nový stav.

Kvôli optimalizácii nárokov na čas výpočtu sa každá konfigurácia okolia, ktorá sa vyskytne zapisuje do asociatívneho poľa spolu s hodnotou nového stavu, ktorá bola vyhodnotením podmienkových pravidiel zistená, alebo príznakom, že ani jedno podmienkové pravidlo nezodpovedá. Pri určovaní ďalšieho stavu bunky je tak najprv zisťovaná prítomnosť jej okolia v tomto asociatívnom poli a vyhodnocovanie podmienkových pravidiel sa deje len pri prvom výskyte konfigurácie okolia. Takto sa vlastne vytvára reprezentácia prechodovej funkcie tabuľkovými pravidlami.

5.3 Kódovanie fenotypu

Aby bolo možné prechodovú funkciu vyvíjať genetickým algoritmom, je nutné jej fenotyp – usporiadanú množinu podmienkových pravidiel zakódovať do genotypu – obsahu chromozómu jedinca populácie. Bola zvolená reprezentácia poľom celých čísel, z ktorých každé vyjadruje jednu podmienku podmienkového pravidla alebo ďalší stav. Ďalší stav je uložený priamo ako jeho číselná hodnota, ale pri podmienke je jedným číslom reprezentovaná podmienková funkcia aj operátor. To je uskutočnené tak, že hodnota génu zodpovedá súčtu indexu funkcie a hodnoty operátoru. Dekódovanie do podoby podmienky podmienkového pravidla teda prebieha tak, že index funkcie je výsledok celočíselného delenia hodnoty génu počtom stavov a operand je zvyšok po delení.

Mutácia génu prebieha vygenerovaním novej náhodnej valídnej hodnoty. Z celého chromozómu sa náhodne vyberie nastavený počet mutovaných génov a u každého z nich mutácia nezávisle buď prebehne s nastavenou pravdepodobnosťou alebo neprebehne.

Je implementované jednobodové kríženie, ktoré prebieha z určenou pravdepodobnosťou. Selekcia je turnajová. Je možné zvoliť použitie elitizmu.

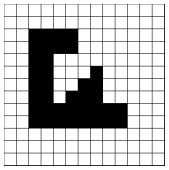
Kapitola 6

Experimenty

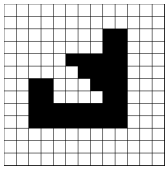
Prvou zo zvolených úloh je transformácia vzoru v dvojrozmernom binárnom CA. Cieľom bude otočiť zadaný vzor o 90° proti smeru hodinových ručičiek. Experiment je prebratý z [3]. Počiatočná konfigurácia automatu má podobu štvorca so stranou 16 buniek, pričom všetky bunky sú mŕtve okrem uprostred umiestneného vzoru. Cieľom je dosiahnuť rovnaký stav len s otočeným vzorom, nemali by sa teda mimo neho objaviť žiadne prebytočné živé bunky. Určenie fitness hodnoty teda prebieha porovnaním všetkých buniek cieľovej konfigurácie so skutočnou konfiguráciou CA. Ak sa stavy zodpovedajúcich buniek zhodujú hodnota fitness sa zvýši. Najvyššia hodnota fitness tak zodpovedá počtu buniek automatu a nastáva pri dokonalej zhode konfigurácii. Nulová hodnota je pri opačnej konfigurácii. Počiatočná konfigurácia alebo konfigurácia obsahujúca len mŕtve bunky tak už má hodnotu pomerne vysokú, čo však neprekáža, pokiaľ je dodržané, že lepšie riešenia dosiahnu vyššiu hodnotu.

Nakoľko vopred nevieme, koľko krokov bude potrebných k rotácii, odsimuluje sa vhodne zvolený maximálny počet krokov, počas ktorých sa hľadá maximum hodnoty fitness. Aby sa zbytočne, neporovnávali ani prvé kroky, kedy je nepravdepodobné dokončenie rotácie, nastavuje sa aj dolná hranica, sledovaného úseku. V rámci neho stačí, aby bola požadovaná konfigurácia dosiahnutá raz, a nemusí byť stabilná.

Najprv bol experiment uskutočnený s parametrami snažiacimi sa napodobniť výsledky v článku. Nastavenie genetického algoritmu je zhrnuté v tabuľke 6.1. Cieľom pokusu bolo zistiť, či budú dosiahnuté výsledky zodpovedať doterajším a preveriť tak funkčnosť implementácie genetického algoritmu, celulárneho automatu a experimentu. Porovnanie štatistík je v tabuľke 6.2. Ako vidno, výsledky sú pomerne podobné. Odlišnosti môžu byť spôsobené rozdielmi vo fitness funkcii, polovičným počtom generácii, drobnými rozdielmi v implementácii algoritmov a ich súčastí (napr. genetické operácie), ako aj samotnou stochaistickou povahou algoritmu. Sú však natoľko podobné, že môžeme implementáciu považovať za funkčnú a bez závažných chýb, a prikročiť k ďalším experimentom.



(a) zdroj



(b) cieľ

Obrázek 6.1: Rotácia vzoru

Parameter	Hodnota
Počet mutovaných génov	6
Pravdepodobnosť mutácie	50 %
Pravdepodobnosť kríženia	50 %
Veľkosť populácie	8
Veľkosť turnaja	4
Maximálny počet generácií	500000
Použitý elitizmus	nie
Podmienkové funkcie	$==, !=, <=, >=, *$
Počet stavov CA	2
Počet podmienkových pravidiel	22
Okolie	Moorovo
Okrajové podmienky	nulové
Najnižší kontrolovaný krok CA	5
Najvyšší počet krokov CA	15
Počet opakovaní experimentu	50

Tabulka 6.1: Nastavenie genetického algoritmu podľa článku

	Úspešnosť evolúcie	Priemerný počet generácii
Pôvodný výsledok	64 %	158919
Nový experiment	62 %	245627

Tabulka 6.2: Porovnanie výsledkov experimentu s doterajšími

6.1 Optimalizácia parametrov

Výsledky evolúcie nie sú ovplyvňované len spôsobom jej implementácie ale aj nastavením parametrov, ktorými sa riadia jednotlivé časti algoritmu, hlavne evolučné operácie. Cieľom nasledujúcich experimentov je zistiť ich vplyv a preveriť vhodnosť ich voľby v predošlom pokuse.

Tabuľka 6.3 ukazuje výsledky pri rôznych maximálnych počtoch génov, ktorých sa dotkne jedna operácia mutácie. Oproti predošlému experimentu bol znížený maximálny počet generácii evolúcie na 100000, ostatné parametre zostali rovnaké. Je vidieť, že vplyv skúmaného parametru na úspešnosť evolúcie je značný. Nasvedčuje to tomu, že operácia mutácie nové genetické informácie ňou získané sú v tomto prípade hlavným spôsobom prehľadávania stavového priestoru a hybnou silou evolúcie. Ukázalo sa, že pôvodne nastavených 6 génov, u ktorých sa môže uskutočniť mutácia nie je, aspoň pri tejto riešenej úlohe a nastavení ostatných parametrov, úplne optimálnou voľbou. Lepšia úspešnosť bola dosiahnutá pri nižších nastaveniach, čo zodpovedá viac systematickému zisťovaniu vplyvu menších zmien na kvalitu riešenia. Naproti tomu pri ďalšom zvýšení parametru úspešnosť prudko klesá, nakoľko sa evolúcia začína chovať viac náhodne. Do ďalšieho experimentu bude teda hodnota upravená na 4.

Ďalšou využívanou operáciou je kríženie. Teoreticky by malo pomáhať k výmene častí lepších riešené medzi jedincami prospievať tak evolúcii. Kríženie je pri tvorbe potomstva využívané s nastavenou pravdepodobnosťou. Zmeny tejto hodnoty by mali ukázať nakoľko je operácia kríženia priebehu a výsledkom algoritmu skutočne prospešná.

Hodnoty v tabuľke 6.4 naznačujú, že kríženie evolúcii nijako neprospieva, dokonca, ako

Počet mutovaných génov	Úspešnosť evolúcie	Priemerný počet generácii
2	52 %	49934
3	68 %	42569
4	68 %	37647
5	66 %	44355
6	20 %	61673
7	6 %	31120

Tabulka 6.3: Vplyv počtu mutovaných génov na úspešnosť evolúcie pri maximálnom počte 100000 generácií

Pravdepodobnosť kríženia	Úspešnosť evolúcie	Priemerný počet generácii
0 %	76 %	33548
10 %	74 %	39443
20 %	74 %	37905
30 %	74 %	38741
40 %	68 %	40012
50 %	68 %	37647
60 %	74 %	36795
70 %	72 %	38251

Tabulka 6.4: Vplyv pravdepodobnosti kríženia na úspešnosť evolúcie

naznačuje najvyššia úspešnosť dosiahnutá pri jeho úplnej neprítomnosti, jej mierne škodí. Môže to znamenať, že návrh celulárnych automatov a využívaná reprezentácia ich prechodovej funkcie podmienkovými pravidlami je pre túto operáciu celkovo nevhodná. Napríklad sa môžu pri snahe o dosiahnutie požadovaného chovania CA výhodne dopĺňať podmienkové pravidlá umiestnené na začiatku a blízko konca usporiadanej množiny. Ak sa nad chromozómom reprezentujúcim takúto prechodovú funkciu vykoná operácia kríženia, je pravdepodobné, že sa táto súvislosť naruší. Novovytvorené riešenie tak nebude dávať zmysel a bude menej kvalitné, než jeho rodičia, takže jeho prežitie v populácii bude málo pravdepodobné a evolúcia jeho vytvorením nebude napredovať.

Na druhej strane sa nedá vylúčiť, že ide len o nevhodnosť implementovaného druhu kríženia a bolo by možné vytvoriť operáciu kríženia vhodnú a prospešnú pre tento typ úloh. Z nasledujúcich experimentov bolo kríženie ako zbytočná záťaž vypustené.

6.2 Modifikácia mutácie

Dosiaľ uvedené pokusy boli zamerané na zistenie vplyvu vybraných parametrov základnej verzie genetického algoritmu. Na základe zistených štatistických dát sa podarilo ich úpravou vylepšiť chovanie algoritmu pri riešení zvolenej úlohy. Ďalšie experimenty budú zamerané na skúmanie javov ovplyvňovaných samotnou podobou algoritmu a skúmanie vplyvu jej modifikácii na výsledky.

Jednou z vlastností charakterizujúcich mnohé časti genetického algoritmu je jeho stochaickosť. Aj vďaka nej je schopný prinášať dosiaľ neznáme riešenia problémov často lepšie než iné metódy. Na druhej strane je vhodné tieto charakteristiky udržovať v takom rozsahu, ktorý nebude konvergenciu k výsledku negatívne ovplyvňovať. Príkladom hľadania takejto rovnováhy je už opísaná optimalizácia parametru počtu mutovaných génov.

	Úspešnosť evolúcie	Priemerný počet generácii
Bez modifikácie	76 %	33548
S modifikáciou	22 %	37969

Tabulka 6.5: Vplyv modifikovanej mutácie

Pravdepodobnosť	Úspešnosť evolúcie	Priemerný počet generácii
0 %	76 %	33548
10 %	76 %	43479
25 %	72 %	38792
50 %	58 %	40311
100 %	22 %	37969

Tabulka 6.6: Vplyv pravdepodobnosti použitia modifikovanej mutácie

Aj napriek tomu však rovnako, ako je operácia mutácie dôležitým a jediným zdrojom novej genetickej informácie vedúcej k zdokonaľovaniu populácie, môže takisto (alebo ešte s väčšou pravdepodobnosťou) kvalitu jedincov znižovať. Nekvalitné riešenia majú malú šancu prejsť selekciou a v populácii sa tak dlho neudržia, ale v extrémnejších prípadoch môže nastať, že kvalitné riešenie, ku ktorého vytvoreniu viedlo viacero náhodných mutácií, je v ďalšej generácii veľmi poškodené a nezachová sa žiaden jedinec vykazujúci jeho priaznivé znaky.

Zabrániť tomuto javu by mohla modifikácia, ktorá by pri každej mutácii porovnávala kvalitu rodiča a potomka a do novej generácie zaradila toho lepšieho z nich. Takáto úprava bola implementovaná a jej vplyv na vývoj pri dosiaľ používaných parametroch obsahuje tabuľka 6.5. Ako vidno ukázalo sa, že aj napriek predpokladu, že by povoľovanie len mutácii s pozitívnym efektom na jedinca priebehu algoritmu nemalo uškodiť, sú pre jeho priebeh potrebné aj ostatné mutácie a náhodný prvok, ktorý do jeho priebehu prinášajú. Ich zablokovanie touto zmenou v algoritme sa tak na jeho výsledkoch podpísalo veľmi nepriaznivo.

Takýto neúspech však nemusí znamenať úplnú nesprávnosť predloženej hypotézy. Problémom môže byť prílišná tvrdosť pravidla povoľujúceho alebo blokujúceho mutáciu. Preto bola modifikácia doplnená o náhodne rozhodnutie, kedy sa z nastavenou pravdepodobnosťou buď použije vždy zmutovaný potomok, alebo lepší jedinec z dvojice potomka a rodiča. Vplyv tejto pravdepodobnosti na evolúciu je zhodnotený v tabuľke 6.6. Ako však vidno, aj pri nižšej pravdepodobnosti použitia modifikovanej mutácie, funguje algoritmus menej úspešne, než bez modifikácie.

Zámerom modifikácie by malo byť do istej miery podľa nastavenej pravdepodobnosti usmerňovať evolúciu vo väčšej miere ku kvalitnejším riešeniam, než je to u základného genetického algoritmu. To by mohlo byť možné kompenzovať vyšším počtom mutovaných génov a hľadanie riešenia tak urýchliť. Úspešnosť genetického algoritmu pri 7 mutovaných génoch a pri rôznych pravdepodobnostiach použitia modifikácie mutácie je zaznamenaná v tabuľke 6.7. Úspešnosť genetického algoritmu pri takomto nastavení modifikovanej mutácie sa podarilo zvýšiť na 90 %. Okrem toho by mohla, ako sa zatiaľ zistenými štatistikami ukázalo, znížiť vysokú citlivosť algoritmu na nastavenie počtu mutovaných génov, čo môže mať priaznivý účinok pri riešení neznámych problémov.

Pravdepodobnosť	Úspešnosť evolúcie	Priemerný počet generácii
25 %	72 %	33548
40 %	90 %	43479
50 %	82 %	36302

Tabulka 6.7: Vplyv pravdepodobnosti použitia modifikovanej mutácie pri 7 mutovaných génoch

6.3 Modifikácia selekcie

Iným možných problémom pre genetický algoritmus je uviaznutie v lokálnom maxime fitness funkcie. Dôjsť k nemu môže ak je v stavovom priestore oblasť, ktorá síce neobsahuje hľadané riešenia, ale aj napriek tomu v nej výraznejšie stúpa hodnota fitness. Selekciou sú takéto čiastočné riešenia preferované, až sa môže stať, že sa z populácie vytratia jedince z iných častí stavového priestoru. Potom je už málo pravdepodobné, že by operácia mutácie dokázala vytvoriť jedinca, ktorý by zároveň nebol z oblasti lokálneho maxima a zároveň by mal hneď v prvých generáciách dostatočnú hodnotu fitness, aby sa dokázal oproti hodnotám z okolia lokálneho maxima presadiť v selekcii. Algoritmus tak môže v takomto prípade skončiť neúspechom.

Ako je z uvedeného možné usúdiť, takýto neželaný vývoj je spôsobený operáciou selekcie, ktorá preferuje okamžité zvýšenie alebo udržanie hodnoty fitness bez ohľadu na dlhodobý vývoj evolúcie. Navrhovaným riešením je tak snažiť sa do istej miery obmedziť jej vplyv, aby dostali šancu na zdokonalenie aj riešenia, ktoré sa na prvý pohľad nezdajú sľubné.

Používané druhy selekcia sa vo všeobecnosti dajú zaradiť niekde medzi extrémy elitizmu a rovnomerného náhodného výberu. V prípade elitizmu je vždy zvolené jedinec s najvyššou hodnotou fitness a ostatné jedince majú nulovú pravdepodobnosť výberu. Naproti tomu pri rovnomernom náhodnom výbere má jedinec s minimálnou hodnotou fitness rovnakú šancu na prežitie ako najlepší jedinec. Princípom navrhovanej úpravy by malo byť posunúť využívanú ruletovú selekciu o bližšie k rovnomernému výberu.

To bolo uskutočnené tak, že pri selekcii sa podľa nastavenej pravdepodobnosti vybraný jedinec buď vymaže z populácie rodičov, takže sa už viac nerozmnoží, alebo prebehne selekcia klasickým spôsobom. Ak bude pravdepodobnosť nastavená na 100 %, znamená to, že každý jedinec bude mať v novej populácii práve jedného potomka.

Aby sa ešte viac posilnila diverzita populácie, bola pridaná nová operácia vloženia náhodne generovaného jedinca počas evolúcie. Zámerom je aby sa táto operácia vykonala niekoľkokrát za evolúciu a novou genetickou informáciou zvyšovala šancu na úspech. Pri tom by sa využili vlastnosti modifikovanej selekcie, vďaka ktorej by generovaný jedinec dostal dostatočnú príležitosť na zdokonalenie a vyrovnanie sa hodnotou fitness ostatku populácie, či jeho predstihnutie.

Implementovaná bola táto operácia tak, že pred každou selekciou sa podľa nastavenej pravdepodobnosti vyhodnocuje, či nebude nahradená vložením nového jedinca. V súhre s modifikovanou selekciou by to malo znamenať, že najväčšiu šancu na nahradenie má najhorší jedinec, pretože, ak je niektorá zo selekcii v rámci kroku evolúcie nahradená, je pravdepodobné, že naň v novej populácii neostane miesto.

Vplyv nastavení novovzniknutých parametrov (pravdepodobností) pri piatich mutovaných génoch je zhrnutý v tabuľke 6.8. Ak sa so 100% pravdepodobnosťou vykonáva modifikovaná mutácia aj modifikovaná selekcia, algoritmus má až 100% úspešnosť, čo je dosiaľ najlepší výsledok. Pri len čiastočnom použití modifikovaných operácii, čo bolo pri poku-

P_{mutmod}	P_{selmod}	P_{new}	Úspešnosť evolúcie	Priemerný počet generácii
100 %	100 %	0,1 %	100 %	27619
100 %	100 %	0,01 %	96 %	25282
100 %	100 %	0 %	82 %	29044
100 %	90 %	0,1 %	45 %	30964
100 %	90 %	0,01 %	45 %	39226
100 %	90 %	0,1 %	15 %	42496
50 %	100 %	0,1 %	0 %	-

Tabulka 6.8: Vplyv pravdepodobnosti modifikovanej mutácie P_{mutmod}, pravdepodobnosti modifikovanej selekcie P_{selmod} a pravdepodobnosti vloženia nového jedinca P_{new}

soch len s modifikovanou mutáciou prospešné, sa úspešnosť rýchlo znižuje. Malo by teda byť vhodné pri použití takéhoto typu algoritmu pevne nastaviť použitie modifikovaných operácii.

Môže sa zdať, že pri nízkych pravdepodobnostiach s akými sa aplikuje vkladanie nového jedinca do populácie táto operácia nebude mať na priebeh evolúcie vplyv. Ako je vidieť porovnaním prvých troch riadkov tabuľky 6.8, opak je pravdou. Je potrebné si uvedomiť, že napríklad za 100000 krokov evolúcie populácie ôsmich jedincov sa pri pravdepodobnosti 0,1 % operácia vykoná priemerne 800 krát. Oproti stavu, keď sa náhodné jedince generujú len na začiatku evolúcie tak ide o stonásobné množstvo takéhoto typu genetickej informácie.

Podobne je možné sa domnievať, že v situácii, keď sa vždy použije modifikovaná selekcia, teda bude mať každý jedinec v novej generácii práve jedného potomka, je možné vypustiť implementáciu turnaja. Táto domnienka bola preverená nastavením veľkosti turnaja na 1, čím sa vlastne selekcia stala náhodnou. Ako vidieť v tabuľke 6.9, nie je tomu tak. Dôvodom je, že turnaj zabezpečuje, že náhodne vkladané nové jedince štatisticky nahrádzajú v populácii zväčša menej kvalitné riešenia.

6.4 Transformácia upraveného vzoru

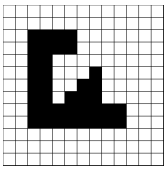
Predošlým pokusmi sa úspešne podarilo zlepšiť priebeh evolúcie CA rotujúceho predpísaný vzor. Výhodou genetického algoritmu je, že pracuje pre široký obor úloh. Navrhnuté modifikácie by tak neboli účinné, keby zlepšovali výsledky len pri problémoch rovnakej obtiažnosti, či dokonca boli špecificky účinné len pri konkrétnej úlohe. Nasledujúce experimenty si kladú za cieľ preveriť, či tomu tak nie je.

Na obrázku 6.2 je mierne zväčšená verzia doteraz využívaného vzoru. Rozdiel tvorí len niekoľko pridaných živých buniek, čo by pre bežne používané postupy transformácie obrázkov nepredstavovalo problém. Pri snahe vyjadriť túto transformáciu lokálnymi pravidlami ale už malé skomplikovanie vzoru môže zvýšiť náročnosť.

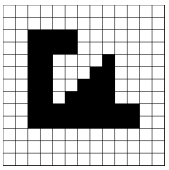
Tabuľka 6.10 porovnáva úspešnosť rôznych modifikácii algoritmu. Parametre boli rovnaké ako v doterajších experimentoch. Rozdiely medzi nastavením jednotlivých modifikácii plynuli z toho, čo sa v nich najviac osvedčilo. Algoritmus v základnej verzii mutuje 4 gény a nevykonáva kríženie, s modifikovanou mutáciou sa mutuje 7 génov a pravdepodobnosť jej aplikácie je 40 %, algoritmus zo všetkými modifikáciami ich používa s pravdepodobnosťou 100 %, náhodného jedinca vkladá s pravdepodobnosťou 0,01 % a mutuje sa maximálne 5 génov. Tento prístup kopíruje predpokladané praktické využitie, kedy by bolo nevhodné resp. nemožné pre novoriešenú úlohu robiť optimalizáciu parametrov.

N_{tour}	P_{mutmod}	P_{selmod}	P_{new}	Úspešnosť evolúcie	Priemerný počet generácii
1	100 %	100 %	0.01 %	75.0 %	31205.066666666666
4	100 %	100 %	0,01 %	96 %	25282

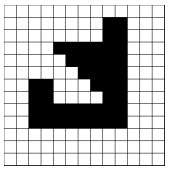
Tabulka 6.9: Vplyv vyradenia selekcie nastavením veľkosti turnaja N_{tour} na 1



(a) pôvodný vzor pre porovnanie



(b) zdroj



(c) cieľ

Obrázek 6.2: Rotácia väčšieho vzoru

Výsledky v tabuľke 6.10 ukazujú podstatné zlepšenie činnosti genetického algoritmu, zvlášť v prípade využitia všetkých modifikácii. Potvrdzuje sa tým, že modifikácie sú schopné obstáť pri riešení úloh rôznej obtiažnosti.

6.5 Výpočet mocniny

Cieľom posledných experimentov bude preveriť schopnosť navrhnutých optimalizácii pozitívne ovplyvňovať aj vývoj celulárnych automatov riešiacich iné úlohy než transformáciu vzoru. Za týmto účelom bola zvolená úloha počítania mocniny čísla v jednorozmernom viacstavovom CA. Vstupom algoritmu počítajúceho mocninu je počiatočná konfigurácia CA, ktorej jediné živé bunky sú v súvislom bloku buniek v stave 1, ktorých počet zodpovedá argumentu mocniny. Výsledok výpočtu je možné odčítať po ustálení CA, teda keď má vo dvoch nasledujúcich krokoch rovnakú konfiguráciu. Jedinými živými bunkami CA je vtedy súvislý blok buniek ľubovoľného rovnakého stavu, ktorých počet zodpovedá výsledku mocniny.

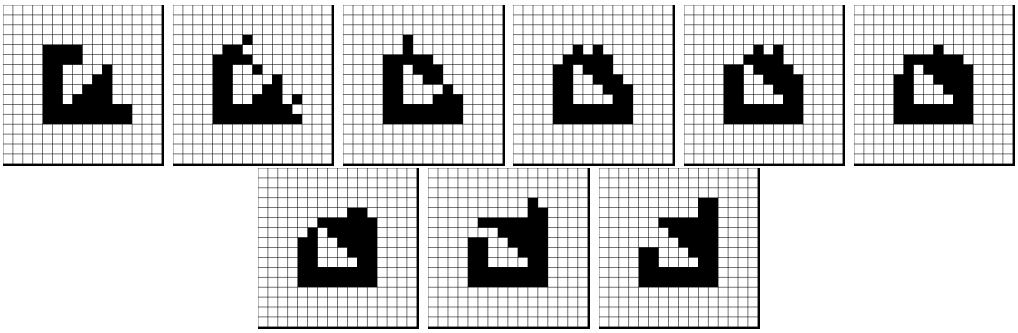
Pri implementácii fitness funkcie sa postupovalo podľa skúseností v [7]. Simulácia automatu prebieha, kým sa neustáli, alebo do maximálneho počtu krokov, ktorý závisí od vstupu. Fitness hodnota je znižovaná odchýlkou veľkosti najväčšieho súvislého bloku buniek jedného nenulového stavu od správneho výsledku výpočtu a tiež mierou výskytu živých buniek mimo najväčšieho bloku.

Čiastočné riešenia sú testované vstupmi 0 až 6. U úspešných riešení sa navyše zisťuje všeobecnosť, podľa toho, či dávajú správny výsledok v intervale 7 až 100. Parameter všeobecnosti je vhodným ukazovateľom, podľa ktorého zistíme vplyv modifikácii genetického algoritmu na kvalitu a vlastnosti riešení.

Parametre genetického algoritmu pri jednotlivých stupňoch modifikácie boli rovnaké

Algoritmus	Úspešnosť evolúcie	Priemerný počet generácii
V základnej verzii	14 %	49694
S modifikovanou mutáciou	18 %	52903
So všetkými modifikáciami	42 %	48205

Tabulka 6.10: Porovnanie úspešnosti pri rotácii väčšieho vzoru

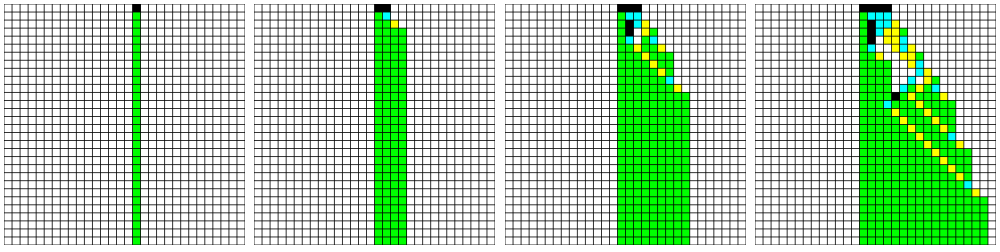


Obrázek 6.3: Príklad nájdeného riešenia rotácie

Algoritmus	Úspešnosť evolúcie	Podiel všeobecných riešení	Podiel všeobecných riešení zo všetkých pokusov	Priemerný počet generácii
V základnej verzii	4 %	100 %	4 %	54232
S modifikovanou mutáciou	22 %	83 %	18 %	47987
So všetkými modifikáciami	52 %	35 %	18 %	57268

Tabulka 6.11: Porovnanie úspešnosti pri počítaní mocniny

ako v predošlom pokuse. CA má 5 stavov. Výsledky sú zhrnuté v tabuľke 6.11. Zvýšenie úspešnosti genetického algoritmu je opäť pomerne značné. Zaujímavé je, že s modifikáciami genetického algoritmu klesá medzi nájdenými riešeniami podiel všeobecných. Ťažko ale tento vývoj algoritmu vytýkať, hlavne keď aj napriek tomu rastie celková pravdepodobnosť nájdenia všeobecného riešenia v rámci daného počtu behov experimentu. Môžeme naopak považovať za výhodu algoritmu, že nachádza rozmanité riešenia, ktoré majú zodpovedajúcu fitness hodnotu. Jeho zvýšenú účinnosť by bolo možné využiť špecifickejšou fitness funkciou berúcou aspoň čiastočne ohľad aj na všeobecnosť riešenia.



Obrázek 6.4: Príklad nájdeného riešenia mocniny

Kapitola 7

Záver

V práci bola vysvetlená problematika celulárnych automatov, evolučných algoritmov a podmienkových pravidiel. Ďalej bola popísaná implementácia genetického algoritmu a simulátora celulárnych automatov s prechodovou funkciou vyjadrenou podmienkovými pravidlami, ktorý slúži ako základ pre výpočet fitness hodnoty, udávajúcej, nakoľko CA plní zadanú úlohu.

Na probléme rotácie vzoru v dvojrozmernom binárnom automate bola potvrdená správnosť implementácie a schopnosť genetického algoritmu hľadať riešenia s efektivitou porovnateľnou s predošlými publikáciami. Následne bola úpravou počtu mutovaných génov a vypustením neúčinnej operácie kríženia zvýšená úspešnosť pri obmedzení na 100000 generácii z 20 % na 76 %.

Potom boli popísané predpokladané problémy straty riešení mutáciou a uviaznutia v lokálnom maxime. Ako možnosť zmiernenia týchto problémov boli navrhnuté modifikácie. Upravená operácia mutácie povolí zaradenie potomka do populácie miesto rodiča len, ak nedošlo k zníženiu kvality jedinca. Experimenty ukázali, že pri použití tejto úpravy s vhodne zvolenou pravdepodobnosťou alternujúc s klasickou mutáciou sa zníži citlivosť genetického algoritmu na nastavenie počtu mutovaných génov a mierne zvýšenie úspešnosti na 90 %.

Ďalšie navrhnuté modifikácie selekcie každého rodiča najviac jedenkrát a s istou malou pravdepodobnosťou vkladanie náhodného jedinca do populácie priniesli podstatné zvýšenie úspešnosti, ktorý dosiahla 100 %.

Vhodnosť úprav genetického algoritmu pre širší obor úloh, než len rotácia jedného konkrétneho vzoru mali za úlohu overiť experimenty s rotáciou o niečo väčšieho vzoru a počítaním mocniny v jednorozmernom viacstavovom CA. V oboch prípadoch sa potvrdil prínos modifikácii značným zvýšením úspešnosti z 14 % na 42 % a z 4 % na 52 % oproti základnej verzii.

Za potenciálne možnosti pokračovania v práci možno považovať podrobnejšie preskúmanie nedostatkov genetického algoritmu a prípadná náhrada navrhnutých modifikácii ich sofistikovanejšími variantami. Okrem toho by mohlo byť užitočné pokúsiť sa vytvorený účinnejší genetický algoritmus aplikovať na náročnejšie úlohy a využiť jeho úspešnosť na možnú prísnejšiu špecifikáciu funkcie fitness a získavanie kvalitnejších riešení, než tomu bolo so základnou verziou algoritmu, či dokonca pokúsiť sa nájsť dosiaľ neznáme riešenia.

Literatura

[1] Bentley, P.: *Evolutionary Design by Computers*. číslo 1 in Evolutionary Design by Computers, Morgan Kaufman Publishers, 1999, ISBN 9781558606050.
URL `https://books.google.sk/books?id=EgC6LBAH5r8C`

[2] Bidlo, M.; Vasicek, Z.: Evolution of cellular automata using instruction-based approach. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, June 2012, s. 1–8, doi:10.1109/CEC.2012.6256475.

[3] Bidlo, M.; Vasicek, Z.: Evolution of cellular automata with conditionally matching rules. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, June 2013, s. 1178–1185, doi:10.1109/CEC.2013.6557699.

[4] Cook, M.: Universality in Elementary Cellular Automata. *Complex Systems*, ročník 15, č. 1, 2004: s. 1–40.

[5] Gardner, M.: Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". *Scientific American*, ročník 223, 1970: s. 120–123.
URL `http://www.worldcat.org/isbn/0894540017`

[6] Hynek, J.: *Genetické algoritmy a genetické programování*. Průvodce (Grada), Grada, 2008, ISBN 9788024726953.

[7] Jílek, T.: *Optimalizace návrhu celulárních automatů*. Diplomová práce, FIT VUT v Brně, 2014.

[8] Kvasnička, V.; Pospíchal, J.; Tiňo, P.: *Evolučné algoritmy*. Edícia vysokoškolských učebníc / Slovenská technická univerzita, Slovenská technická univerzita, 2000, ISBN 9788022713771.

[9] Langton, C. G.: Self-reproduction in cellular automata. *Physica D: Nonlinear Phenomena*, ročník 10, č. 1, 1984: s. 135–144.

[10] Neumann, J. V.: *Theory of Self-Reproducing Automata*. Champaign, IL, USA: University of Illinois Press, 1966.

[11] Sarkar, P.: A Brief History of Cellular Automata. *ACM Comput. Surv.*, ročník 32, č. 1, Březen 2000: s. 80–107, ISSN 0360-0300, doi:10.1145/349194.349202.
URL `http://doi.acm.org/10.1145/349194.349202`

[12] Sipper, M.: *Evolution of parallel cellular machines: the cellular programming approach*. Lecture notes in computer science, Springer, 1997, ISBN 9783540626138.

[13] Wolfram, S.: *A New Kind of Science*. Wolfram Media, 2002, ISBN 1579550088. URL `http://www.wolframscience.com`

Příloha A

Manuál k implementácii genetického algoritmu

Vzhľadom k experimentálnej povahe programu nevznikla potreba zavedenia používateľského rozhrania. Evolúcia sa spúšťa príkazom `./ga` v programovom adresári. Parametre pokusov sa nastavujú makrami v súbore `config.h`.

- `OPER_COUNT` – hodnota 5 povoľuje všetky podmienkové funkcie, nastavením na 4 je možné zakázať wildcard
- `STATE_COUNT` – počet stavov CA
- `CMR_COUNT` – počet podmienkových pravidiel
- `INIT_FILE` – súbor obsahujúci počiatočnú konfiguráciu CA
- `GOAL_FILE` – súbor obsahujúci cieľovú konfiguráciu CA, vyhodnocovaná je podľa ďalších nastavení
- `ALL_TABLE_RULES` – výpis tabuľkových pravidiel vrátane tých, ktoré sa pri evolúcii nepoužili, vhodné použiť pri výpočtoch v CA
- `VON_NEUMANN, MOORE, ONE_DIMENSIONAL` – zvoliť jedno z makier, nastavenie typu okolia
- `CYCLIC_BORDERS` – cyklické okrajové podmienky
- `UNIT` – jednotky v ktorých sa udávajú pravdepodobnosti, hodnota 100 zodpovedá percentám
- `MUTAGENES` – maximálne množstvo mutovaných génov
- `MUT_PROB` – pravdepodobnosť mutácie
- `TOURNAMENT_SIZE` – veľkosť turnajovej selekcie
- `CPROB` – pravdepodobnosť mutácie
- `MAX_GENERATIONS` – maximálny počet generácii evolúcie
- `ELITISM` – použitie elitizmu (v pokusoch v práci sa nepoužilo)

- `MAXIMUM_FITNESS` – maximálna hodnota fitness. `ca.getGoalSize()` je vhodné pri porovnávaní celej konfigurácia CA, napríklad pri rotácii. Pri vyhľadávaní vzoru v CA v replikačných úlohách je potrebné nastaviť násobok. Pre výpočet mocniny je určená hodnota makra `((MAX_INPUT+1) * UNIT)`
- `FIRST_CHECKED_STEP` – najnižší krok, kedy sa vyhodnocuje fitness
- `MULTIPLE_GOALS` – umožňuje hodnotiť rôznu konfiguráciu vo viacerých krokoch. V práci sa nepoužilo
- `COMPARE_GOAL` – porovnávanie celej konfigurácie, napríklad pri rotácii vzoru
- `COMPUTATION` – počítanie mocniny. Z posledných troch je potrebné zvoliť jedno makro.
- `MAX_INPUT` – najvyšší hodnotený vstup mocniny
- `EXPERIMENTAL` – genetický algoritmus s modifikáciami popísanými v práci
- `MOD_MUT_PROB` – pravdepodobnosť aplikácie modifikovanej mutácie
- `MOD_SEL_PROB` – pravdepodobnosť aplikácie modifikovanej selekcie
- `RANDOM_IND_PROB` – pravdepodobnosť vloženia náhodného jedinca. Vzhľadom k nízkym hodnotám sa zadáva celá podmienka
- `AUTOMATON_STEPS` – maximálny počet krokov CA. Pri výpočte mocniny sa ignoruje.

Genetický algoritmus vypisuje na štandardný výstup buď podmienkové pravidlá uvedené riadkom `CMRs:` a tabuľkové pravidlá uvedené riadkom `TABLE_RULES:` alebo text `fail` v prípade neúspechu.

Pre vizualizáciu je potrebné tabuľkové pravidlá uložiť ako vstupný súbor pre program BiCAS. Pri hromadnom spustení experimentov skriptom `run_tests` je z výstupného súboru `nohup.out` možné súbory tabuľkových pravidiel vygenerovať skriptom `cutupresult.py`. Štatistiky ráta skript `./stats.py`.