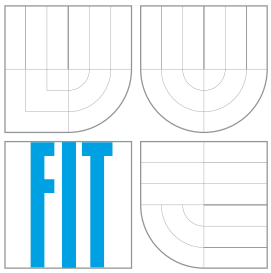
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

TECHNIKY REPREZENTACE PRO EVOLUČNÍ NÁVRH CELULÁRNÍCH AUTOMATŮ

REPRESENTATION TECHNIQUES FOR EVOLUTIONARY DESIGN OF CELLULAR AUTOMATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN KOVÁCS
AUTHOR

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

BRNO 2016

Abstrakt

Tato práce je zaměřena na experimentální testování různých reprezentací přechodové funkce celulárního automatu. V práci je prezentována výpočetní platforma celulárního automatu. Celulární automat má mnoho potenciálních využití při simulacích různých přírodních jevů, fyzikálních systémů, atd. Jeho paralelní výpočet založený na lokálních bunečných interakcích je však náročný na programování, proto je návrh programu automatu často přenechán evolučním technikám. Evoluční techniky založené na Darwinově teorii evoluce byly už mnohokrát využity pro nalezení stejně dobrých nebo lepších než lidsky navržených řešení různých problémů. Evoluční techniky ale vyžadují speciální zakódování řešených problémů, a právě z toho důvodu jsou reprezentace přechodové funkce celulárního automatu zkoumány. Zkoumané reprezentace zahrnují klasickou tabulkovou reprezentaci, podmínková pravidla a kartézske genetické programování. Testovacím problémem pro určení efektivity reprezentací je funkce druhé mocniny.

Abstract

The aim of this thesis is to experimentally evaluate the performance of several distinct representations of transition functions for cellular automata. Cellular automata have many potential applications for simulating various phenomena (e.g. natural processes, physical systems, etc.). Parallel computation of cellular automata is based on local cell interactions. Such computation, however, may prove difficult to program the CA, which is the reason for applying evolutionary techniques for the design of cellular automata in many cases. Evolutionary algorithms, based on Darwin's theory of evolution, have been used to find human-competitive solutions to many problems. In order to perform the evolutionary design of cellular automata, special encodings of the candidate solutions are often necessary. For this purpose the performance testing of various representations of the transition functions will be investigated. In particular, table representation, conditionally matching rules, and genetic programming will be treated. The problem of square calculations in cellular automata will be considered as a case study.

Kľúčové slová

celulární automat, genetický algoritmus, přechodová funkce, podmínková pravidla, kartézske genetické programování

Keywords

cellular automaton, genetic algorithm, transition function, conditionally matching rules, cartesian genetic programming

Citácia

KOVÁCS, Martin. *Techniky reprezentace pro evoluční návrh celulárních automatů*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bidlo Michal.

Techniky reprezentace pro evoluční návrh celulárních automatů

Prehlásenie

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Michala Bidla, Ph.D. Uvedl jsem všechny literátní prameny, ze kterých jsem čerpal.

.....
Martin Kovács
22. mája 2016

Poďakovanie

Děkuji Ing. Michalovi Bidlovi, Ph.D., za jeho odbornou pomoc, férový přístup a cenné rady při psaní této diplomové práce. Tato práce byla podpořena Ministerstvem školství mládeže a tělovýchovy z Národního programu udržitelnosti II (NPU II) v rámci projektu IT4Innovations excellence in science - LQ1602.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Kapitola 1

Úvod

Celulárne automaty majú v dnešnej dobe výkonných a kompaktných počítačov veľký potenciál hlavne pre potreby simulácie prírodných javov, v ktorých figuruje samoorganizácia a zmena globálneho stavu systému na základe lokálnych interakcií jeho jednotlivých prvkov. Avšak jedným z hlavných problémov, ktoré sprevádzajú výpočty na platforme celulárnych automatov je efektívny návrh programu, ktorý by požadovaný výpočet realizoval. Z toho dôvodu sú pre návrh programov skúmané automatické techniky, ktoré by mohli potenciálne nahradiť analytický návrh programu. Motiváciou pre túto prácu je prieskum možností genetických algoritmov ako automatických techník vývoja programov pre viacstavové celulárne automaty. Kľúčovým prvkom genetických algoritmov je zakódovanie programu celulárneho automatu do chromozómu jedincov v populácii. Konvenčné techniky reprezentácie programu pre viacstavové celulárne automaty neboli doteraz v kombinácii s genetickými algoritmami úspešné [3]. Kľúčom k úspešnej aplikácii genetických algoritmov pri automatickej tvorbe programov pre viacstavové celulárne automaty by mohli byť pokročilé reprezentačné techniky prechodovej funkcie ako napríklad genetické programovanie [7], kartézske genetické programovanie [9], prípadne reprezentácia CMR [4].

Cieľom tejto práce je vypracovať porovnávaciu štúdiu vybraných techník pre reprezentáciu prechodovej funkcie v celulárnom automate. Hlavná časť modelu vytvoreného za týmto účelom pozostáva z genetického algoritmu. Výpočet implementovaného celulárneho automatu prebieha vo fitness funkcii genetického algoritmu. Populácia, nad ktorou je výpočet automatu vykonávaný, pozostáva z jedincov predstavujúcich zakódovanie jednotlivých reprezentačných techník. Dôležité časti modelu sú v práci detailne analyzované. Nad každou z reprezentácií je spustená sada experimentov. Jednotlivé experimenty sa z hľadiska genetického algoritmu líšia počtom mutovaných prvkov. Ďalej sú analyzované experimenty s rozličnými nastaveniami pre počet stavov, ktoré môže bunka nadobudnúť. Najdôležitejšou časťou práce je analýza výsledkov vykonaných experimentov, ktorá ukáže účinnosť jednotlivých reprezenačných techník v rámci využitia genetickými algoritmami.

Štruktúra práce je nasledujúca. V kapitole 2 je definovaný celulárny automat ako výpočtová platforma pre simuláciu rôznych fyzikálnych a biologických javov. Elementárny celulárny automat, jeho vývoj a triedy správania sú použité ako prototyp celulárnych automatov. Implementáciou Turingovho stroja v automate Game of Life je znázornená ich výpočtová sila. Praktické využitie celulárnych automatov je predvedené na vybraných modeloch z oblasti prírodných vied. Kapitola 3 popisuje problematiku a účinnosť genetických algoritmov pri riešení optimalizačných úloh. Zároveň sú uvedené definície dôležitých pojmov z tejto oblasti. Hlavné algoritmy pre prácu s evolučnými technikami sú detailne analyzované v hlavnej časti tejto kapitoly. Kapitola 4 sa zaoberá technikami reprezentácie

prechodovej funkcie celulárneho automatu. Podrobne sú preskúmané nasledujúce reprezentácie: základná tabuľková reprezentácia, podmienkové pravidlá a kartézske genetické programovanie. Návrh experimentov, prípadové štúdie a rozbor problému druhej mocniny v jednorozmernom celulárnom automate sú uvedené v kapitole 5. Úspešnosť uvedenených reprezentácií je experimentálne preskúmaná v kapitole 6, ktorá obsahuje tabuľky s výsledkami jednotlivých sád experimentov a diskusiu o vhodnosti danej reprezentácie v kombinácii s evolučným návrhom programov pre viacstavové celulárne automaty. V záverečnej kapitole 7 sú zhrnuté poznatky získané z teoretickej aj praktickej časti práce a možný smer pokračovania výskumu, či prípadné rozšírenie o iné experimenty.

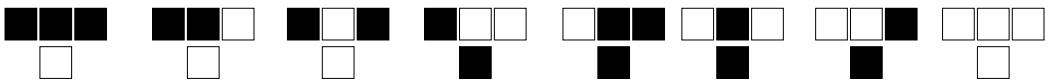
Kapitola 2

Celulárne automaty

Celulárny automat možno chápať ako idealizovaný matematický model, dostatočne abstraktný na popis rôznych fyzikálnych systémov. V celulárnom automate sú priestor a čas diskrétnymi veličinami a skúmané fyzikálne veličiny nadobúdajú konečný počet diskrétnych hodnôt. Obrazové znázornenie celulárneho automatu typicky pozostáva z nekonečnej n-rozmernej mriežky premenných (buniek), z ktorých každá nadobúda konečný počet diskrétnych hodnôt. Konfigurácia celulárneho automatu v jednotlivých diskrétnych momentoch je úplne definovaná stavom každej bunky v danom diskrétnom momente. Priestorová diskretizácia celulárneho automatu spočíva v rozdelení priestoru na n-rozmernú mriežku buniek a jeho časová diskretizácia je daná potenciálne nekonečnou postupnosťou konfigurácií. Každá konfigurácia vzniká z predošlej po aplikovaní prechodovej funkcie celulárneho automatu na každú bunku. Pri zmene konfigurácie je nasledujúci stav určený z aktuálneho stavu bunky a jej okolia, pričom je stav každej bunky aktualizovaný simultánne. Okolie je zvyčajne definované všetkými bunkami vo všetkých rozmeroch bližšími, alebo rovnako vzdialenými od aktuálnej bunky ako vopred stanovený polomer. Polomer zostáva počas výpočtu konštantný. [14]

2.1 Elementárny celulárny automat

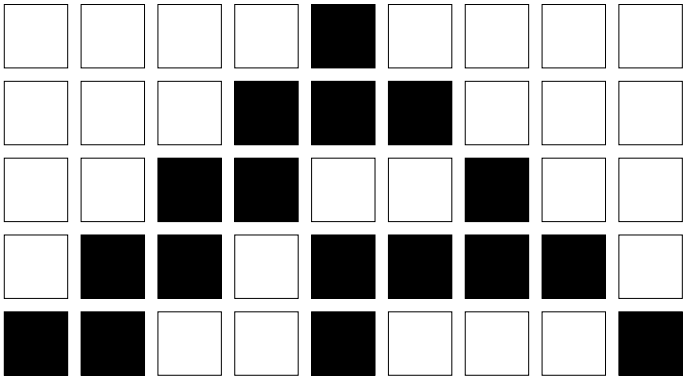
Za elementárny celulárny automat je možné považovať jednorozmerný binárny celulárny automat s okolím s polomerom 1. Okolie takého automatu môže nadobúdať práve osem rôznych konfigurácií Existuje teda $2^8 = 256$ elementárnych celulárnych automatov. Prechodová funkcia každého elementárneho celulárneho automatu je označená unikátnym číslom z rozsahu 0 až 255, ktoré reprezentuje spôsob akým daná funkcia vyhodnotí jednotlivé konfigurácie okolia bunky. Obrázok 2.1 znázorňuje elementárny celulárny automat s prechodovou funkciou 30 (čierny štvorček predstavuje stav 1 a naopak biely stav 0).



Obr. 2.1: Prechodová funkcia 30 [15]

Prechodová funkcia je definovaná ako súbor všetkých možných kombinácií stavov okolia bunky (prvý riadok) a jej nasledujúci stav na základe danej kombinácie (druhý riadok). Názov prechodovej funkcie je odvodený od postupnosti hodnôt nasledujúceho stavu bunky,

ktorá môže byť reprezentovaná ako číslo v binárnej sústave. Na obrázku 2.2 je zobrazených prvých päť krokov vývoja elementárneho celulárneho automatu s prechodovou funkciou 30. Zobrazená je len tá vybraná časť nekonečnej mriežky, ktorú vývoj automatu behom týchto krokov mení.



Obr. 2.2: Vývoj celulárneho automatu s prechodovou funkciou 30 [15]

Každý z riadkov odpovedá jednému diskrétnemu časovému okamihu vývoja celulárneho automatu. Časový okamih každého riadku je bezprostredným následovníkom časového okamihu predchádzajúceho riadku. Na prvom riadku je zobrazená počiatočná konfigurácia automatu, teda akýsi počiatočný časový okamih systému. Uvedený vývoj celulárneho automatu je dosiahnutý synchronizovanou (simultánnou) aplikáciou prechodovej funkcie 30 na každú bunku v automate v danom časovom okamihu. [15]

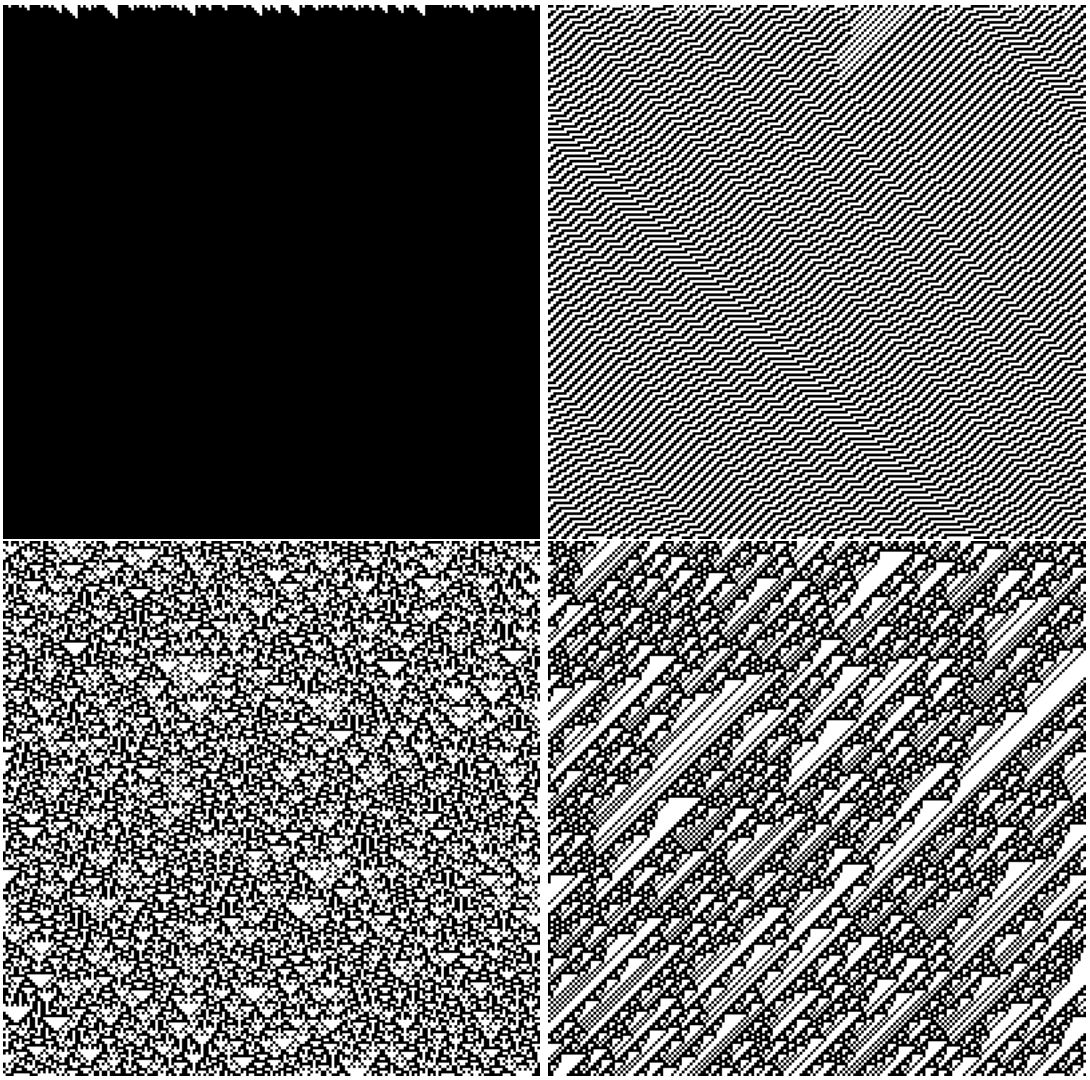
2.2 Triedy správania

Pri pozorovaní jednotlivých elementárnych celulárnych automatov počas vývoja z náhodných počiatočných konfigurácií je možné vidieť štyri druhy ich správania. Tieto klasifikoval Wolfram ako štyri triedy správania [15]:

1. správanie je veľmi jednoduché a takmer všetky počiatočné konfigurácie rýchlo dosiahnu rovnakú koncovú konfiguráciu,
2. koncových konfigurácií je veľa, no každá z nich je zložená z jednoduchých vzorov a po čase sa ustáli alebo sa cyklicky opakuje,
3. koncová konfigurácia nie je ustálená, často vyzerá ako náhodný šum a ak sa nejaký vzor objaví, tak je v nasledújucich krokoch rozbitý,
4. po čase v automate vznikajú náhodne vyzerajúce lokalizované štruktúry, ktoré medzi sebou môžu navzájom interagovať a tak tvoriť zaujímavé a komplexné výsledky.

Obrázok 2.3 znázorňuje vybrané prvky jednotlivých tried správania elementárnych celulárnych automatov. Ako zástupca prvej triedy je na obrázku znázornený elementárny automat s prechodovou funkciou 252 (vľavo hore). Tento automat už po niekoľko málo krokoch dosiahol svoju koncovú konfiguráciu, ktorá sa jeho vývojom už ďalej nemenila. Zástupcom druhej triedy správania je na obrázku automat s prechodovou funkciou 11 (vpravo hore).

Jeho konečná konfigurácia sa cyklicky posúva. Automat s prechodovou funkciou 90 (vľavo dole) zastupuje tretiu triedu správania elementárnych celulárnych automatov a jeho konečná konfigurácia pripomína náhodný šum občas prerušený nejakou jednoduchou štruktúrou. Posledným automatom na obrázku je automat s prechodovou funkciou 106 (vpravo dole). V jeho konečnej konfigurácií je jasne vidieť jednoduché, ale aj komplexné štruktúry vyplnené náhodným šumom.



Obr. 2.3: Vybrané prvky tried správania elementárnych cel. automatov. Zdroj obrázkov: www.wolframalpha.com

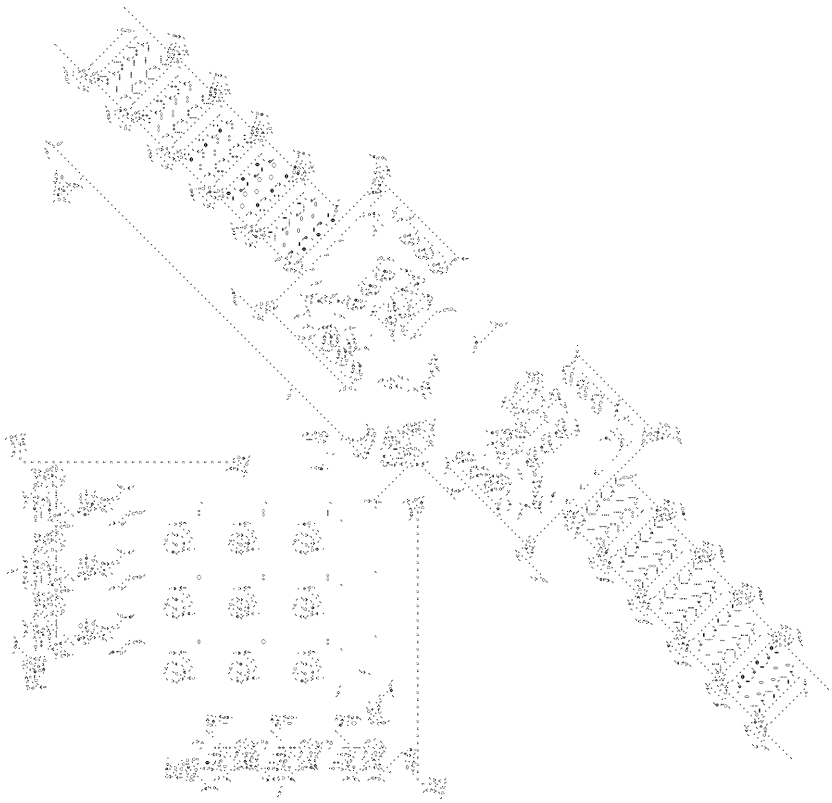
2.3 Využitie celulárnych automatov v informatike

Celulárne automaty je možné považovať za abstraktný model počítania. Počiatočná konfigurácia automatu v takom prípade predstavuje vstupné dáta výpočtu. Po niekoľkých krokoch evolúcie automatu je potom dosiahnutá koncová konfigurácia interpretovateľná ako

výsledok výpočtu. Je formálne dokázaná existencia celulárnych automatov, ktoré majú výpočtové schopnosti univerzálneho Turingovho stroja, čo im umožnuje pracovať ako počítače obecného využitia. Prvý takýto automat prezentoval von Neumann [10]. Jeho automat je dvojrozmerný. Má bunky, ktoré dosahujú 29 stavov a počíta s von Neumannovým okolím bunky (okolie tvorí bunka samotná a jej najbližšia východná, západná, severná a južná susediaca bunka). Tento automat sa neskôr podarilo zredukovať na osem a štvor-stavový. Ak uvážime Moorovo okolie (von Neumannovo rozšírené o najbližšiu severovýchodnú, severozápadnú, juhovýchodnú a juhozápadnú susediacu bunku), celulárny automat je schopný univerzálneho výpočtu už pri bunkách s dvomi stavmi. [2] Najznámejší taký automat je Conwayov Game of Life. Game of Life je dvojrozmerný binárny celulárny automat s Moorovým okolím a nasledujúcimi pravidlami:

- každá živá bunka s dvomi alebo tromi živými susediacimi bunkami ostáva žiť,
- každá živá bunka s iným počtom živých susediacich buniek umiera,
- každá neživá bunka s práve tromi susediacimi bunkami ožíva.

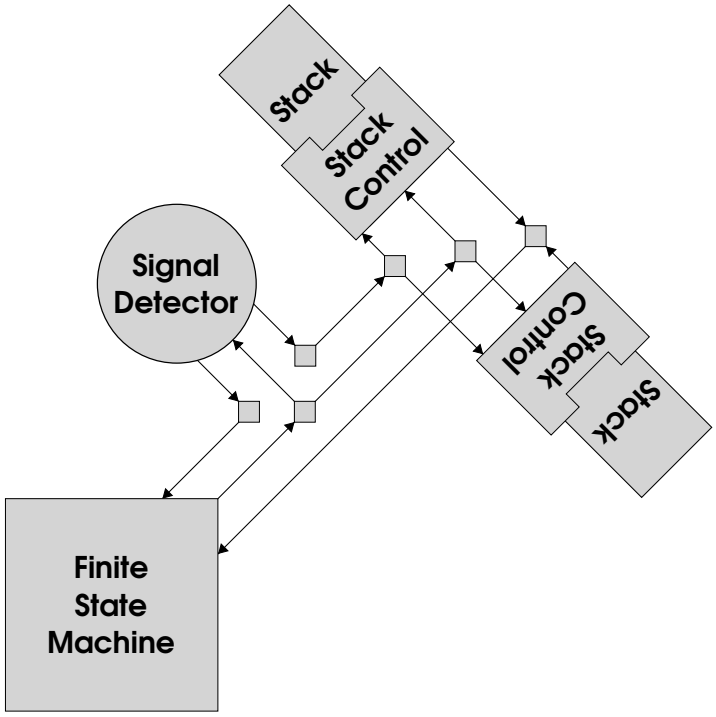
Jeho univerzalitu dokázal Rendell konštrukciou univerzálneho Turingovho stroja v automate Game of Life. Obrázok 2.4 znázorňuje univerzálny Turingov stroj v tomto automate. [11]



Obr. 2.4: Implementácia univerzálneho Turingovho stroja v automate Game of Life [11]

Na obrázku 2.5 je zobrazený diagram k univerzálnemu Turingovmu stroju z obrázku 2.4. Tento Turingov stroj je trojstavový a na jeho páske sa objavujú tri rôzne symboly. Komponent konečného stavového riadenia obsahuje adresový mechanizmus v ľavej (hodnota stavu) a spodnej (hodnota symbolu) časti. Uprostred komponentu sa nachádza deväť

pamäťových blokov, v ktorých je uložená tabuľka popisujúca činnosť automatu pri jednotlivých kombináciách vnútorného stavu a vstupného symbolu. Páska Turingovho stroja je zložená z dvoch zásobníkov, ktoré sa ťahajú diagonálne do ľavého horného a pravého dolného rohu automatu. Zásobníky sú spojené mechanizmom, ktorý zaisťuje aby, práve jeden z nich vykonával operáciu *push* a druhý operáciu *pop*. [11]



Obr. 2.5: Diagram univerzálneho Turingovho stroja v automate Game of Life [11]

2.4 Využitie v iných vedných oblastiach

Celulárne automaty sú s úspechom využívané pri modelovaní biologických systémov. Javy ako vývoj a vývin organizmu často podliehajú sade jednoduchých lokálnych pravidiel a samoorganizácii, čo robí z celulárnych automatov perfektného kandidáta na modelovanie takého systému. Jednoduché správanie a fungovanie organizmov môže byť v celulárnych automatoch modelované bunkami s diskrétnymi stavmi, reprezentujúcimi živé bunky alebo tkanivá. Obdobne môžu celulárne automaty modelovať systém nehybných jedincov (lúku kvetov), kde jednotlivé bunky automatu reprezentujú prítomnosť, prípadne druh jedinca, a lokálna prechodová funkcia popisuje vzťah medzi susednými jedincami (rivalita, symbióza, atp.). [14]

V biologicky zameranej literatúre je najlepším príkladom využitia celulárnych automatov modelovanie excitačných médií, ktoré napodobňujú interakcie medzi nervovými alebo svalovými bunkami, funkcionalitu srdečného tkaniva alebo chemické reakcie. V excitačnom médiu existuje stabilný stav, z ktorého možno dostatočne silným vyrušením dosiahnuť excitovaného stavu. Z excitovaného stavu sa po vyprchaní narušenia jednotlivé komponenty média dostávajú do oddychového stavu a nemožno ich opätovne vyrušiť ľubovoľne silným vyrušením. Nakoniec sa oddýchnuté komponenty vracajú do stabilného stavu a môžu znovu

podliehať vyrušeniam. Pri modelovaní celulárnymi automatmi je snaha o zjednodušenie komponentov excitačného média na bunku s čo najmenším počtom stavov (typicky 0 pre stabilný stav, E - angl. excited - pre vyrušený stav a R - angl. recovery - pre oddychový stav). Prechodová funkcia takého automatu by mohla vyzerať následovne:

- ak je bunka v stave E, v ďalšom kroku bude v stave R,
- ak je bunka v stave R, v ďalšom kroku bude v stave 0,
- ak je bunka v stave 0 a aspoň jedna susedná bunka je v stave E, v ďalšom kroku bude v stave E.

Ďalším častým využitím celulárnych automatov v biológii je modelovanie vzťahu korisť - predátor. Príkladom môže byť systém s migrujúcou kolóniou buniek, predstavujúcou ryby, v ktorej pravidelne dochádza k reprodukcii, teda k zväčšovaniu kolónie. V systéme sa taktiež nachádzajú jednotlivé migrujúce bunky, žraloky, ktoré po istom počte krokov zahynú hladom. Pri strete ryby so žralokom ryba zaniká, životnosť žraloka sa obnoví a zároveň sa narodí nový žralok. [5]

Vo fyzike je využitie celulárnych automatov často spájané s problémom rozptylu a pohybu tekutín. Známe sú takzvané lattice-gas celulárne automaty s bunkami s niekoľkými stavmi, kde jednotlivé stavy predstavujú častice tekutiny s rôznymi vektormi rozptylu. Lattice-gas celulárnemu automatu možno nastaviť počiatočnú konfiguráciu, kde jednotlivé bunky reprezentujú častice tekutiny pohybujúce sa určitým smerom. Pravidlá takého automatu potom určujú akým smerom sa budú jednotlivé častice uberať, keď na jednej bunke dôjde k ich vzájomnému stretu. Problémy ako Couettove, Stokesove alebo Blasiusove prúdenie boli úspešne simulované týmto druhom celulárnych automatov. [8]

Kapitola 3

Genetický algoritmus

V informatike existuje mnoho úloh, v ktorých je žiadúce nájsť čo najlepšie riešenie v čo najkratšom čase, pričom prehľadávaný priestor môže byť veľmi veľký. Čím je prehľadávaný priestor rozsiahlejší, tým obtiažnejšie je nájsť dostatočne kvalitné riešenie zadaného problému. V takýchto prípadoch sa riešitelia často snažia hľadať riešenie pomocou rôznych skratiek a heuristík radšej ako pomocou „hrubej sily“. Jedným zo spôsobov „šikovného“ prehľadávania je genetický algoritmus [6]. Genetický algoritmus je stochastická optimalizačná metóda určená práve k preľadávaniu priestoru potenciálnych riešení problému. Ako názov naznačuje, inšpirácia genetického algoritmu pochádza z prírody, kde si Charles Darwin všimol vzťah medzi úspešnosťou prežitia organizmu a jeho schopnosťou sa prispôbiť svojmu okoliu. Darwin označil túto vlastnosť ako fitness organizmu. Darwinova teória evolúcie je založená na myšlienke prirodzeného výberu a hovorí, že potomok dvoch jedincov s dobrou fitness sa s veľkou pravdepodobnosťou taktiež dokáže dobre prispôsobiť svojmu okoliu. Na rovnakom princípe je založený aj genetický algoritmus, kde sú z populácie možných riešení, nachádzajúcich sa v prehľadávanom priestore, náhodným výberom zvolené dva jedince. Jedince s väčšou hodnotou fitness sú volené s väčšou pravdepodobnosťou. Krížením týchto jedincov vznikajú riešenia generácie potomkov. Analýzou genetických algoritmov bolo zistené, že samotné kríženie jedincov často nestačí na nájdenie dostatočne dobrého výsledku a preto bola do procesu reprodukcie zavedená aj operácie mutácie. [7]

3.1 Základné pojmy genetického algoritmu

Pred hlbším skúmaním vlastností a funkcionality gentetických algoritmov je nevyhnutné definovať s nimi súvisiace pojmy. Dôležitým pojmom v súvislosti s genetickými algoritmami je pojem jedinec, tiež označovaný ako chromozóm. Jedinec, alebo chromozóm, je abstrakciou organizmu živej populácie. Analogicky k svojmu živému vzoru, aj jedinec genetického algoritmu má svoj kód. V prírode je kódom organizmu postupnosť DNA sekvencií, tvoriaca jeho genotyp. Genotypom umelého jedinca je postupnosť čísel, typicky binárnej sústavy, ktorá kóduje potenciálne riešenie skúmaného problému. Formálne môžeme jedinca p s genotypom veľkosti k definovať ako prvok množiny binárnych reťazcov dĺžky k.

$$p \in \{0,1\}^k \tag{3.1}$$

Pod pojmom populácia sa v genetických algoritmoch, obdobne ako aj v prírode, rozumie súbor príbuzných jedincov. Populácia je formálne definovaná ako multimnožina jedincov (binárnych reťazcov dĺžky k) s kardinalitou n. Rôzne jedince v populácii môžu mať rovnaký genotyp.

$$P = \{p_1, p_2, \ldots, p_n\} \tag{3.2}$$

Riešenie optimalizačných úloh je často definované ako globálne minimum (úloha minimalizácie) alebo globálne maximum (úloha maximalizácie) v priestore potenciálnych riešení problému. Jednotne sa takéto riešenie označuje ako globálne optimum. Definujme pomocnú účelovú funkciu následovne:

$$f_u : \{0,1\}^k \to \mathbb{R} \tag{3.3}$$

Funckia f_u je úplné zobrazenie množiny možných genotypov jedinca na hodnoty potenciálnych riešení optimalizačného problému, ktoré kódujú. Tak ako v živých systémoch predstavuje fitness schopnosť organizmu prežiť, získať potravu, prípadne sa rozmnožiť, tak v umelých systémoch predstavuje fitness kvalitu riešenia problému, zakódovaného v chromozóme daného jedinca. Účelová funkcia je základom pre určenie fitness umelého jedinca. Často je nepraktické pracovať s intervalom všetkých reálnych čísel a preto sa zavádza funkcia fitness f_{fit}, ktorá normalizuje výsledky účelovej funkcie do intervalu $(0, 1)$.

$$f_{fit}(p) = \frac{f_u(p)}{\sum_{i=0}^{n} p_i} \tag{3.4}$$

Fitness jedinca je teda určená ako funkčná hodnota funkcie f_{fit} s parametrom genotypu $p \in P$ daného jedinca. Funkciu f_{fit} možno brať ako „skrášľujúcu obálku“ účelovej funkcie, a preto je vhodné ju „ušiť na mieru“ riešeného optimalizačného problému, či dokonca ju ani nepoužiť. V takom prípade môže byť fitness jedinca stanovená ako funkčná hodnota samotnej účelovej funkcie. [7]

Ďalšími dôležitými časťami genetického algoritmu sú spôsob náhodného výberu dvoch jedincov populácie za účelom kríženia a modifikácia genotypu ich potomkov. Algoritmy na popis týchto častí sú uvedené v nasledujúcej sekcii.

3.2 Genetické operátory

Pred samotnou definíciou vzorového genetického algoritmu je potrebné uviesť definície často využívaných genetických operátorov. Genetické operátory sú dôležitou súčasťou genetických algoritmov. Pracujú nad aktuálnou populáciou jedincov a ich účelom je tvorba novej populácie jedincov s lepšou schopnosťou riešiť daný problém. Prvým významným genetickým operátorom je operátor selekcie. Operátor selekcie je zodpovedný za výber jedincov z aktuálnej populácie, za účelom vytvorenia novej populácie. Vybrané jedince by mali byť čo najúspešnejšie pri riešení daného problému. Zároveň však počas evolúcie môže nastať situácia, kedy aktuálne najlepšie jedince konvergujú iba k nejakému lokálnemu optimu v priestore riešení, ktoré ale nepatrí medzi optimálne riešenia daného problému. Genetický operátor selekcie preto musí zaviesť určité opatrenia, aby k takému nežiadúcemu efektu nedochádzalo.

Medzi populárne operátory selekcie patrí ruletový výber, výber podľa poradia, deterministický výber a turnajový výber. Najzákladnejší z uvedených je deterministický výber, v ktorom je do novej populácie zaradených niekoľko (počet je vopred daný) jedincov s najvyššou fitness. Zvyšní jedinci v novej populácii sú dotvorení pomocou operátorov mutácie

a kríženia diskutovaných ďalej. Ruletový výber je technika, kde šanca, že jedinec bude náhodne vybraný do novej populácie rastie úmerne s hodnotou jeho fitness. Pravdepodobnosť výberu jedinca určuje pomer jeho fitness voči fitness celej populácie. Výber podľa poradia je veľmi podobný ruletovému výberu. Jedince v populácii sú zoradené podľa svojej fitness a k výberu jedinca dochádza na základe jeho poradia a nie hodnoty jeho fitness. Tieto techniky sú pre mnoho optimalizačných úloh príliš elitistické a hrozí, že celá populácia uviazne v lokálnom optime a globálne optimum nebude dosiahnuté. V práci je použitý turnajový výber jedincov. Turnajový výber je založený na myšlienke rytierskeho turnaja. Z populácie je náhodne vybraných niekoľko jedincov, ktorí „súperia“ svojou fitness hodnotou. Jedinec s najlepšou fitness hodnotou sa stáva víťazom a postupuje do novej populácie. Detaily implementácie ruletového výberu sú diskutované v sekcii 3.4.[12]

Ďalší významný genetický operátor predstavuje operátor mutácie. Operátor mutácie je aplikovaný na jedince v populácii za účelom zlepšenia ich fitness. Jedinec, alebo chromozóm, ktorý pozostáva z génov môže predstavovať takmer požadované riešenie stanoveného problému. Úlohou operátora mutácie je náhodne pozmeniť niekoľko génov chromozómu, ktoré by potenciálne mohli zmeniť takmer požadované riešenie problému na jedno zo skutočných riešení problému. Tiež je však možné, že zmena v génoch chromozómu aktuálne riešenie naopak zhorší. Operátor mutácie preto „mutuje“ náhodne iba malý počet génov.

Posledným diskutovaným genetickým operátorom je operátor kríženia. Operátor kríženia je aplikovaný na páry jedincov a v genetickom algoritme zastáva exploratórnu funkciu. Najpoužívanejším variantom operátora kríženia je jednobodové kríženie, kde je náhodne zvolený index delenia chromozómu. Každý z vybraného páru chromozómov je na danom indexe rozdelený na dva podchromozómy. Podchromozómy za indexom delenia sú medzi pôvodnými jedincami prehodené, čím vznikajú nové, potenciálne nepreskúmané jedince. Kríženie môže takýmto spôsobom zaniesť do populácie existujúcich potenciálnych riešení nové potenciálne riešenie z úplne inej časti priestoru riešení. [7]

3.3 Základný genetický algoritmus

V algoritme 1 je definovaný vzorový genetický algoritmus. Jeho vstupom je maximálny povolený počet generácií. Jedinec reprezentujúci najlepšie nájdené riešenie je jeho výstupom. V prípravnej fáze algoritmu sú nastavené premenné udávajúce aktuálnu generáciu, kritérium pre ukončenie výpočtu a počiatočnú populáciu jedincov na svoje počiatočné hodnoty. Počiatočná populácia je zložená z náhodne vybraných prvkov prehľadávaného priestoru potenciálnych riešení. Následne je cyklicky modifikovaná populácia jedincov pokiaľ výpočet nedovŕši maximálnu povolenú generáciu, alebo nie je splnené kritérium pre ukončenie výpočtu. V cykle je vytvorená prázdna pomocná populácia a je inkrementovaná hodnota aktuálnej generácie, čo reprezentuje odovzdanie genetickej informácie z aktuálnej populácie na potomstvo. Následne je každému jedincovi aktuálnej populácie stanovená jeho fitness, ktorej hodnota stúpa s kvalitou riešenia zadaného problému reprezentovaného jedincom. Ak je v aktuálnej populácii jedinec s požadovanou fitness hodnotou, cyklus končí. Inak v ďalšom kroku algoritmu prebieha náhodný výber dvoch jedincov (rodičov) z aktuálnej populácie, z ktorých vzniknú dva nové jedince. Tie sú pridané do pomocnej populácie. Proces výberu a tvorby novej populácie sa opakuje pokiaľ pomocná populácia nemá rovnaký počet jedincov ako aktuálna. Pomocná populácia sa stáva aktuálnou. Po ukončení cyklu výpočtu je za najlepšie nájdené riešenie prehlásený jedinec aktuálnej populácie s najväčšou hodnotou fitness, ktorý je zároveň výstupom algoritmu. Genetický operátor selekcie je súčasťou funkcie `SelectRandomIndividual`. Operátory kríženia a mutácie sú súčasťou funkcie `Reproduce`.

Algoritmus 1: GeneticAlgorithm [7]

vstup : t_{max}
výstup: p_{opt}

```
1  t ← 0
2  stop ← false
3  P ← InitializeRandomPopulation()
4  while t < t_max ∧ not stop do
5        t ← t + 1
6        Q ← ∅
7        foreach p ∈ P do
8              ComputeFitness(p)
9        end foreach
10       if MaxFitness(P) ≥ fit_max then
11             stop ← true
12       else
13             while | Q | < | P | do
14                   p_1 ← SelectRandomIndividual(P)
15                   p_2 ← SelectRandomIndividual(P)
16                   (q_1, q_2) ← Reproduce(p_1, p_2)
17                   Q ← Q ∪ { q_1, q_2 }
18             end while
19       end if
20       P ← Q
21 end while
22 p_opt ← IndividualWithHighestFitness(P)
23 return p_opt
```

3.4 Implementácia genetického algoritmu

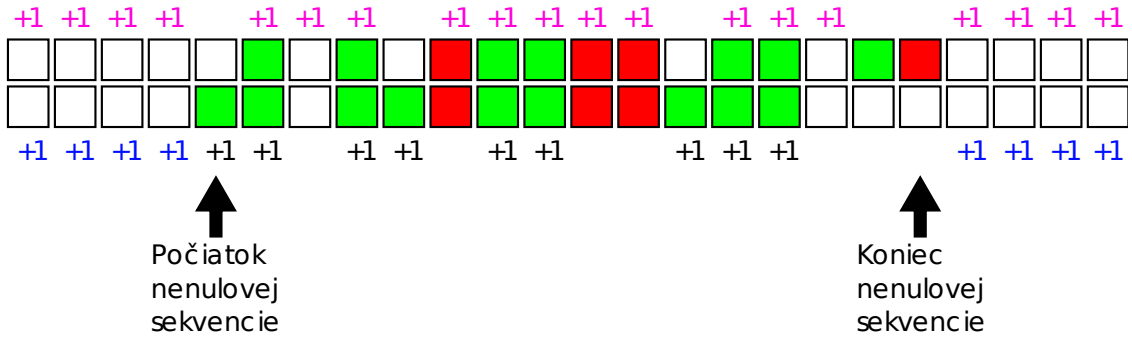
V tejto časti sú popísané hlavné súčasti genetického algoritmu použitého pri experimentoch s celulárnym automatom. Použitý genetický algoritmus je založený na algoritme 1. Dôkladne sú diskutované časti implementácie zahŕňajúce vyhodnotenie fitness chromozómu a genetický operátor selekcie. V experimentoch je použitá populácia s ôsmimi jedincami. Každý jedinec je reprezentovaný chromozómom, teda poľom celočíselných hodnôt, ktorý predstavuje zakódovanú prechodovú funkciu celulárneho automatu. Všetky gény chromozómov sú na začiatku evolúcie inicializované náhodnými platnými hodnotami.

Vyhodnotenie fitness chromozómu je prvým krokom evolučného cyklu a zároveň je výpočtovo najzložitejšou časťou algoritmu. Výpočet hodnoty fitness je v práci implementovaný nasledujúcim spôsobom. Každý jedinec, ktorého fitness ešte nie je známa, podstúpi ohodnotenie pomocou fitness funkcie. Maximálna fitness, ktorú jedinec môže dosiahnuť je daná vzťahom $2 * n * w$, kde n je počet testovaných vstupných hodnôt a w je šírka celulárneho automatu. Konštanta 2 poukazuje na fakt, že fitness hodnotí nielen poslednú (koncovú) konfiguráciu automatu, ale aj konfiguráciu predposlednú.

Vo fitness funkcii je pre každú zo sady vstupných hodnôt vyhodnotený vývoj celulárneho automatu. Vývoj celulárneho automatu je určený prechodovou funkciou zakódovanou v hodnotenom chromozóme z počiatočnej konfigurácie automatu pre danú vstupnú hod-

notu. Po stanovenom počte krokov vývoja automatu je získaná jeho koncová konfigurácia. Fitness funkcia hodnotí nielen správnosť, ale aj stabilitu výsledku. Výsledok je považovaný za stabilný, ak má každá bunka automatu rovnaký stav ako v predchádzajúcej konfigurácií. Za každú bunku automatu, ktorá je stabilná je chromozómu pridelený jeden „fitness bod“. Výsledok je správny, ak v koncovej konigurácii existuje postupnosť buniek v rovnakom nenulovom stave so správnou dĺžkou (očakávaná koncová konfigurácia je podrobne popísaná v sekcii 5.1). Zároveň musí platiť že zvyšné bunky automatu sú v nulovom stave. Za správnosť výsledku pridelí fitness funkcia chromozómu „fitness bod“ nasledujúcim spôsobom. Za každú bunku v nulovom stave od začiatku postupnosti až do prvej bunky s nenulovým stavom dostane chromozóm jeden „fitness bod“. S prvou bunkou v nenulovom stave s fitness predpokladá počiatok nenulovej sekvencie. Chromozóm dostane „fitness bod“ za každú bunku v stave s v nenulovej sekvencii. Nakoniec dostane chromozóm „fitness bod“ za každú bunku v nulovom stave po skončení nenulovej sekvencie.

Výpočet fitness chromozómu je zobrazený na obrázku 3.1, kde spodná konfigurácia predstavuje posledný krok výpočtu automatu (koncovú konfiguráciu), a vrchná predposledný krok. Dĺžka očakávanej nenulovej sekvencie je 16. Na obrázku 3.1 predstavujú fialové body „fitness body“ získané za stabilitu riešenia. Modré a čierne body chromozóm získa za správnosť riešenia. Modré body predstavujú správnosť výsledku v nulovej sekvencii. Čierne body predstavujú správnosť výsledku v nenulovej sekvencii.



Obr. 3.1: Vyhodnotenie fitness funkcie

Pri takejto špecifikácii je možné aby chromozóm za daný vstup dostal plnú fitness aj vtedy, keď nenulová sekvencia nikdy nezačne a koncová konfigurácia bude pozostávať len z buniek v nulovom stave. Aby takému nežiadúcemu javu funkcia zamedzila, je výsledok daný vzťahom $b - (x^2 - c)$, kde b je celkový počet „fitness bodov“ a c je počet buniek v nenulovej sekvencii. Tento vzťah zaisťuje, že maximálnu fitness získa chromozóm práve vtedy, keď sa dĺžka nenulovej sekvencie bude rovnať druhej mocnine vstupu. Pre daný vstup je teda fitness chromozómu 36. Rovnaký výpočet je potrebné vykonať pre každý vstup samostatne a výsledné hodnoty fitness sčítať. Takto získaná fitness predstavuje celkovú fitness hodnoteného chromozómu. V priebehu evolúcie sa uchováva najlepší nájdený jedinec za účelom predčasného ukončenia algoritmu.

Ďalším krokom v evolučnom cykle je tvorba novej populácie, v ktorej sú aplikované genetické operátory selekcie, mutácie a kríženia. V rámci selekcie je do novej populácie zaradený aktuálne najlepší jedinec a jeho mutant. Zvyšných šesť jedincov je vybraných pomocou turnajového výberu nasledujúcim spôsobom. Z aktuálnej generácie sú náhodne

vybrané dva páry po dvoch jedincoch. Jedinci s vyššou hodnotou fitness z oboch párov sa stávajú „rodičmi“. „Rodičia“ podstúpia jednobodové kríženie a mutáciu s vopred stanovenou pravdepodobnosťou. Z takto modifikovaných „rodičov“ vznikajú dvaja jedinci do nasledujúcej generácie. Pravdepodobnosť kríženia je vo vykonaných experimentoch nulová, a naopak pravdepodobnosť mutácie je 100 %. Pri mutácií je náhodne zvolených niekoľko génov chromozómu jedinca, ktoré sú mutované, teda nahradené nejakou náhodnou platnou hodnotou. Algoritmus končí pri dosiahnutí maximálneho počtu generácií, alebo pri nájdení jedinca s maximálnou fitness.

Kapitola 4

Reprezentácie prechodovej funkcie

V kapitole 2 bol predstavený pojem celulárny automat ako model výpočtu. Matricou pre výpočet v celulárnych automatoch je mriežka buniek. Samotný výpočet je však realizovaný cyklickou aplikáciou prechodovej funkcie automatu. V praktických aplikáciach výpočtových systémov je žiadúce, aby bol výpočet vykonávaný čo najrýchlejšie a čo najefektívnejšie. Tento dôvod je zároveň podstatou snahy o optimalizáciu prechodovej funkcie, ktorá je z hľadiska efektivity výpočtu najdôležitejším prvkom celulárneho automatu. Optimalizácia prechodovej funkcie môže prebiehať za pomoci rôznych techník, čo typicky vedie k jej rozličným reprezentáciam. Techniky diskutované v tejto kapitole sú založené na zovšeobecňovaní pravidiel prechodovej funkcie, prípadne redukcii ich počtu.

Z hľadiska evolučných algoritmov je veľmi žiadúce aby, bol chromozóm homogénny. Operácie, ako mutácia a kríženie sú v evolučných algoritmoch veľmi časté a je žiadúce, aby boli vykonané rýchlo a efektívne. Na homogénnom poli hodnôt je operácia mutácie triviálna (jednoduchý náhodný výber hodnoty z rozsahu platných hodnôt), čo značne prispieva k efektivite algoritmu. Homogenita chromozómu tiež prispieva k jednoduchosti operácie kríženia, ktorú je možné implementovať ako jednoduchú zámenu častí dvoch chromozómov, rozdelených v rovnakých bodoch. Homogénne pole tiež zjednodušuje implementáciu inicializácie chromozómu.

4.1 Tabuľková reprezentácia

Tabuľku je možné považovať za základnú reprezentáciu prechodovej funkcie. V tabuľkovej reprezentácii je na jednotlivých riadkoch popísaná každá možná konfigurácia okolia bunky spolu so stavom, do ktorého sa po aplikácii prechodovej funkcie bunka dostane. Za príklad tabuľkovej reprezentácie je možné považovať obrázok 2.1 prechodovej funkcie 30 z kapitoly 2.1. Na obrázku vyjadruje prvý riadok konfiguráciu okolia bunky. Druhý riadok špecifikuje stav po aplikácii prechodovej funkcie. Tabuľka pre prechodovú funkciu 30 by vyzerala následovne.

Stĺpce tabuľky reprezentujú jednotlivé prvky okolia bunky. Spodný index prvku udáva jeho pozíciu v okolí bunky, kde $i-1$ je ľavá susediaca bunka skúmanej bunky, i je skúmaná bunka a $i+1$ je pravá susediaca bunka skúmanej bunky. Vrchný index prvku predstavuje diskrétne časové kroky, v ktorých daný prvok nadobúda uvedené hodnoty. Tabuľka teda špecifikuje vplyv hodnôt okolia bunky v čase t na jej stav v nasledujúcom časovom okamihu $t+1$.

c_{i-1}^t	c_i^t	c_{i+1}^t	c_i^{t+1}
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

Tabuľka 4.1: Tabuľková reprezentácia prechodovej funkcie 30

Hlavnou výhodou tabuľkovej reprezentácie je jej výpočtová nenáročnosť. Použitím systematického usporiadania hodnôt okolia je možné tabuľku implementovať ako jednorozmerné pole. V elementárnom celulárnom automate môžeme hodnoty stavu okolia bunky považovať za číslo v binárnej sústave. Tabuľku usporiadame vzostupne podľa čísla odvodeného z okolia bunky. Po takej úprave strácajú stĺpce okolia bunky v tabuľke svoj účel a tabuľka môže byť redukovaná na jednorozmerné pole. Do poľa sú na pozície určené číslom okolia priradené zodpovedajúce hodnoty zo stĺpca c_i^{t+1}. Z hľadiska programátora je takáto redukcia veľmi žiadúca, pretože na určenie nového stavu bunky stačí jediný prístup do pamäte počítača. Predmetom evolúcie sa po redukcii stávajú práve jednotlivé hodnoty poľa.

číslo okolia	c_{i-1}^t	c_i^t	c_{i+1}^t	c_i^{t+1}
$111_b = 7_d$	1	1	1	0
$110_b = 6_d$	1	1	0	0
$101_b = 5_d$	1	0	1	0
$100_b = 4_d$	1	0	0	1
$011_b = 3_d$	0	1	1	1
$010_b = 2_d$	0	1	0	1
$001_b = 1_d$	0	0	1	1
$000_b = 0_d$	0	0	0	0

(a) Tabuľka prechodovej funkcie 30 rozšírená o číslo okolia

číslo okolia	c_{i-1}^t	c_i^t	c_{i+1}^t	c_i^{t+1}
$000_b = 0_d$	0	0	0	0
$001_b = 1_d$	0	0	1	1
$010_b = 2_d$	0	1	0	1
$011_b = 3_d$	0	1	1	1
$100_b = 4_d$	1	0	0	1
$101_b = 5_d$	1	0	1	0
$110_b = 6_d$	1	1	0	0
$111_b = 7_d$	1	1	1	0

(b) Tabuľka prechodovej funkcie 30 usporiadaná vzostupne podľa čísla okolia

0	1	1	1	1	0	0	0

(c) Tabuľka prechodovej funkcie 30 redukovaná na jednorozmerné pole

Obr. 4.1: Transformácia tabuľkovej reprezentácie na jednorozmerné pole

Nevýhodou tabuľkovej reprezentácie je jej extrémna spotreba pamäťových zdrojov. Po-

čet prvkov tabuľky rastie exponenciálne s polomerom okolia bunky a s rastúcim počtom stavov. Počet riadkov tabuľky, a teda aj počet prvkov jej redukovanej podoby, je daný vzťahom $p = a^o$, kde p značí počet riadkov, a počet možných stavov bunky automatu a o veľkosť bunečného okolia. Pri práci s komplexnejšími niekoľko rozmernými automatmi, ktorých bunky môžu nadobúdať viac ako dva stavy pri okolí s vačším polomerom sa táto reprezentácia stáva prakticky nepoužiteľnou. Už pri jednorozmernom automate s polomerom okolia 3 a bunkami s piatimi možnými stavmi potrebuje tabuľková reprezentácia až $5^{2*3+1} = 78125$ riadkov.

4.2 Nastavenia tabuľkovej reprezentácie

Pri tabuľkovej reprezentácii prechodovej funkcie je pole chromozómu prirodzene homogénne a jeho prvky nadobúdajú hodnoty medzi nulou a maximálnym počtom stavov. Pre maximálny počet stavov 8 a okolie bunky automatu s polomerom 1 má chromozóm tabuľkovej reprezentácie veľkosť $(1 + 1 + 1)^8 = 512$.

Nový stav bunky automatu je určený na základe okolia bunky, ktoré je prepočítané na index do poľa chromozómu. Pre okolie bunky automatu s polomerom 1 a maximálnou hodnotou stavu 8 je index vypočítaný pomocou vzťahu $s_{i-1} * 8^2 + s_i * 8 + s_i$, kde s_i je skúmaná bunka automatu. Hodnota na tomto indexe je použitá ako stav bunky v nasledujúcom kroku. Ak každý gén chromozómu môže nadobudnúť práve 8 rôznych hodnôt, potom prehľadávaný stavový priestor dosahuje veľkosť $8^{512} = 2^{1536}$.

4.3 Conditionally matching rules

Conditionally matching rules (ďalej CMR) [4] je ďalšou skúmanou technikou reprezentácie prechodovej funkcie celulárneho automatu. Cieľom tejto techniky je odstrániť extrémnu priestorovú náročnosť tabuľkovej reprezentácie vytvorením sady podmienkových pravidiel. Podmienkové pravidlo je tvarom podobné riadku tabuľky z predchádzajúcej reprezentácie. Pravidlo obsahuje $o+1$ stĺpcov, kde o je veľkosť okolia bunky. Pre jednorozmerný celulárny automat s päť-stavovými bunkami a polomerom okolia 3, by sada podmienkových pravidiel mohla vyzerať ako na obrázku 4.2.

c_{i-3}^t	c_{i-2}^t	c_{i-1}^t	c_i^t	c_{i+1}^t	c_{i+2}^t	c_{i+3}^t	c_i^{t+1}
$\leq$ 3	$=$ 2	$\neq$ 5	$=$ 1	$\geq$ 3	$\geq$ 2	$=$ 4	2
$=$ 1	$\star$ 1	$=$ 3	$\neq$ 4	$\leq$ 4	$\leq$ 3	$=$ 1	1
$\neq$ 2	$\geq$ 4	$\neq$ 4	$=$ 5	$=$ 3	$\star$ 2	$\geq$ 5	4

$\vdots$

Obr. 4.2: Podmienkové pravidlá v reprezentácii CMR

Pri aplikovaní prechodovej funkcie je podmienkové pravidlo vybrané práve vtedy, keď sú splnené podmienky v každom zo stĺpcov okolia bunky. Vo svojom článku [4], Bidlo označil relačné operátory $\leq$, $=$, $\geq$ a $\neq$ ako vhodné pre využitie v kombinácii s evolučnými technikami. Prvé pravidlo z obrázku 4.2 by bolo vybrané vtedy, ak by bol stav bunky $c^t_{i-3} \leq 3$, a zároveň by bol stav bunky $c^t_{i-2} = 2$, a zároveň by bol stav bunky $c^t_{i-1} \neq 5$ a tak ďalej pre celé bunečné okolie. V prípade, že konfigurácia okolia spĺňa podmienky pravidla, je pravidlo vybrané a v nalsedujúcom kroku bunka nadobúda stav uvedený v poslednom stĺpci podmienkového pravidla. Pri aplikácii prvého pravidla z obrázku 4.2 sa v nasledujúcom kroku bunka dostáva do stavu 2. Predmetom evolúcie pri použití CMR sa stávajú všetky časti podmienkového pravidla, teda relačný operátor a referenčný stav každej bunky okolia a taktiež stav bunky v nasledujúcom kroku.

Sada podmienkových pravidiel reprezentácie CMR nemusí byť veľká. Vďaka tomu, že jedno podmienkové pravidlo môže, v závislosti na komplexnosti automatu, pokryť stovky alebo aj tisícky konfigurácií okolia bunky, na zakódovanie prechodovej funkcie často postačuje aj niekoľko málo desiatok prechodových pravidiel. Môžu nastať okrajové prípady, keď nejakú konfiguráciu okolia bunky nepostihuje žiadne podmienkové pravidlo zo sady CMR. Aby bola prechodová funkcia definovaná totálne a deterministicky, je potrebné pre takéto prípady zaviesť špeciálne implicitné pravidlo. V práci je použité implicitné pravidlo uvedené Bidlom v jeho článku [4], ktoré ponechá bunke v nasledujúcom kroku jej aktuálny stav, ak žiadne zo sady pravidiel nepokrýva aktuálnu konfiguráciu jej okolia. Inou možnosťou riešenia tohto problému je vytvorenie práve a podmienkových pravidiel, kde a predstavuje práve počet stavov bunky celulárneho automatu. Podmienky týchto pravidiel by pokrývali všetky konfigurácie okolia, ktoré vedú k rovnakému nasledujúcemu stavu bunky. Takáto realizácia však prináša potrebu komplexnejších podmienok, čo je z hľadiska evolučných techník veľmi nepraktické.

Nevýhodou tejto reprezentácie v porovnaní s tabuľkovou reprezentáciou je zvýšená výpočtová náročnosť. Pri aplikácii prechodovej funkcie je sada podmienkových pravidiel prehľadávaná systematicky od prvého k poslednému, pokiaľ niektoré pravidlo nepokryje konfiguráciu okolia bunky, alebo pokiaľ nie sú preskúmané všetky podmienkové pravidlá. Časová aj priestorová zložitosť výpočtu pri použití tejto reprezentácie je lineárna vzhľadom k veľkosti okolia bunky, a lineárna vzhľadom k počtu podmienkových pravidiel. [4]

4.4 Nastavenia reprezentácie CMR

Pre reprezentáciu CMR nie je chromozóm všeobecne homogénny. Uvažujme chromozóm tvaru:

$$(\ldots, p_{i-n}, s^t_{i-n}, p_{i-n-1}, s^t_{i-n-1}, \ldots, p_i, s^t_i, \ldots, p_{i+n-1}, s^t_{i+n-1}, p_{i+n}, s^t_{i+n}, s^{t+1}_i, \ldots) \quad (4.1)$$

kde s^t_i je hodnota bunky automatu na pozícii i v čase t, p_i je podmienka, ktorá sa k danej bunke vzťahuje, n je polomer okolia bunky a s^{t+1}_i je hodnota danej bunky v nasledujúcom kroku. Daný chromozóm bude nehomogénny, ak sa veľkosť množín stavov bunky automatu a podmienok líši.

Uvedený chromozóm je možné transformovať na homogénnu formu. Nech veľkosť množiny stavov bunky automatu je $|S|$ a veľkosť množiny podmienok je $|P|$. Homogénny chromozóm pre okolie bunky n bude potom obsahovať $n + 1$ bodov, ktoré môžu nadobúdať

práve $|S| * |P|$ hodnôt. Uvažujme nasledujúce zobrazenia:

$$f_1 : S \times P \to SP, \qquad f_1(s,p) = s * p \tag{4.2}$$

$$f_2 : S \to SP, \qquad f_2(s) = s \tag{4.3}$$

Aplikáciou daných zobrazení na vyššie uvedenú formu chromozómu získame homogénny chromozóm tvaru:

$$(\ldots, sp_{i-n}^t, sp_{i-n-1}^t, \ldots, sp_i^t, \ldots, sp_{i+n-1}^t, sp_{i+n}^t, sp_i^{t+1}, \ldots) \tag{4.4}$$

Celočíselným delením novej hodnoty prvku chromozómu maximálnou hodnotou z množiny stavov bunky automatu je možné získať pôvodnú podmienku pre daný prvok chromozómu. Zvyšok po delení predstavuje pôvodný stav bunky. Keďže počet prvkov množiny SP je konečný, je možné výsledky operácií celočíselného delenia a zvyšku po delení vypočítať vopred.

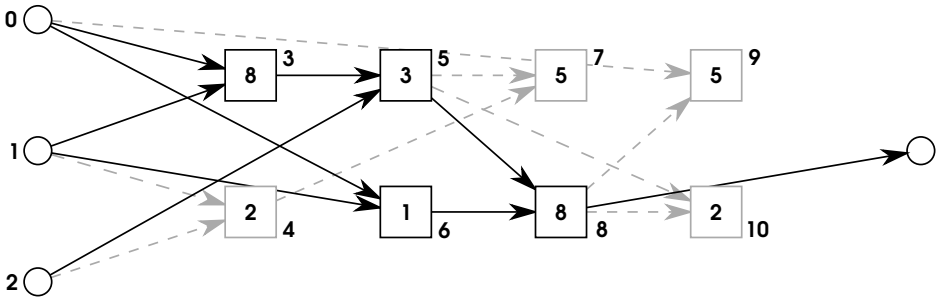
Reprezentácia CMR pracuje so sadou podmienok, ktorá obsahuje relačné operácie rovný nule ($s = 0$), nerovný nule ($s \neq 0$), menší rovný ($s \leq s_r$) a väčší rovný ($s \geq s_r$), kde s je stav bunky v jej okolí a s_r je referenčný stav bunky podmienkového pravidla zakódovaný v chromozóme. Hodnoty, ktoré môžu gény chromozómu nadobudnúť sú pre sadu štyroch podmienok a osem stavov bunky v rozsahu 0 až $4 * 8 = 32$. Hodnoty $0 \leq s_r < 8$ odpovedajú podmienke $s = 0$, hodnoty $8 \leq s_r < 16$ podmienke $s \neq 0$, hodnoty $16 \leq s_r < 24$ podmienke $s \leq (s_r \mod 8)$ a hodnoty $24 \leq s_r < 32$ podmienke $s \geq (s_r \mod 8)$. Ako príklad je uvedený chromozóm s pravidlom $(\ldots, 11, 30, 3, 19, \ldots)$. Aby bolo možné aplikovať uvedené pravidlo musia byť splnené nasledujúce podmienky: ľavá bunka okolia sa nesmie rovnať nule ($8 \leq 11 < 15 \Rightarrow s \neq 0$), skúmaná bunka musí byť v stave väčšom alebo rovnom referenčnému stavu 6 ($24 \leq 30 < 32 \Rightarrow s \geq 6$) a pravá bunka okolia sa musí rovnať nule ($0 \leq 3 < 8 \Rightarrow s = 0$). Po splnení týchto podmienok nadobudne skúmaná bunka v ďalšom kroku hodnotu 3 ($19 \mod 8$).

Veľkosť chromozómu v reprezentácii CMR je pre okolie s polomerom $r = 1$ daná vzťahom $((2 * r + 1) + 1) * m = 4 * m$, kde m je počet podmienkových pravidiel v chromozóme. Prehľadávaný priestor pri použití reprezentácie CMR so sadou štyroch podmienok, ôsmych stavov bunky a okolia bunky s polomerom 1 dosahuje veľkosť $(4*8)^{4*m} = 2^{20*m}$, kde m udáva počet podmienkových pravidiel v chromozóme. Aby s takýmito parametrami reprezentácia CMR prekonala veľkosťou prehľadávaného stavového priestoru tabuľkovú reprezentáciu, musel by jej chromozóm obsahovať aspoň 77 podmienkových pravidiel.

4.5 Kartézske genetické programovanie

Ďalšou používanou reprezentáciou prechodovej funkcie je kartézske genetické programovanie (CGP) [9]. Kartézske genetické programovanie je obdoba genetického programovania [1][7], kde je program zakódovaný formou acyklického grafu. Hlavný rozdiel oproti genetickému programovaniu spočíva v topológii grafu. Pri kartézskom genetickom programovaní sú funkčné bloky usporiadané do dvoj-rozmernej mriežky. Počet funkčných blokov v CGP je definovaný súčinom počtov riadkov a stĺpcov mriežky. Vstupy jednotlivých funkčných blokov môžu byť pripojené na hlavné vstupy programu, prípadne na výstupy iných funkčných blokov tak, aby bola zachovaná acyklickosť grafu. Ďalsím dôležitým parametrom reprezentácie CGP je sada funkcií, ktorých kombináciou a kompozíciou by mal program vykonať určený výpočet.

Prepojenie funkčných blokov je obmedzené parametrom *l-back*, ktorý udáva počet predchádzajúcich stĺpcov, na ktoré sa môže funkčný blok pripojiť. Funkčné bloky v rovnakom stĺpci nesmú byť medzi sebou prepojené. Minimálne prepojenie nastáva, keď *l-back* parameter nadobúda hodnotu jeden. Vtedy môžu byť funkčné bloky pripojené len na bloky v predchádzajúcom stĺpci, alebo na hlavné vstupy programu. Maximálne prepojenie v CGP je dosiahnuté, ak sú všetky funkčné bloky v jednom riadku a hodnota parametru *l-back* zodpovedá počtu stĺpcov. V takom prípade sa každý funkčný blok môže pripojiť na ľubovoľný predchádzajúci blok alebo na hlavný vstup programu. Súčasťou chromozómu CGP sú aj výstupné bloky, ktoré sú prepojené rovnakým spôsobom ako bloky funkčné. Výstupné bloky už nie sú súčasťou CGP mriežky, ale svojím zapojením špecifikujú, ktoré funkčné bloky budú vo výpočte použité, a ktoré nie. Príklad topológie CGP je zobrazený na obrázku 4.3.



Obr. 4.3: Topológia funkčných blokov v CGP

Na obrázku 4.3 sú hlavné vstupy programu zobrazené ako kruhy v ľavej časti obrázku. Výstupný blok CGP predstavuje kruh v pravej časti obrázku. Štvorce predstavujú prepojené funkčné bloky. Nie všetky funkčné bloky v CGP musia, prípadne môžu, byť súčasťou progamu. Zároveň maximálny počet funkčných blokov programu je vymedzený topológiou programu, a počet funkcií v programe je tým pádom konečný. Funkčné bloky sú v programe použité, ak sú pripojené na niektorý z výstupných blokov, alebo na iné funkčné bloky pripojené na výstupný blok. Na obrázku 4.3 majú použité (aktívne) funkčné bloky čierny rám, a nepoužité (neaktívne) funkčné bloky majú sivý rám.

Chromozóm v reprezentácii CGP je zložený z n-tíc predstavujúcich funkčné bloky a je zakončený bunkami reprezentujúcimi výstupné bloky programu. Jednotlivé n-tice obsahujú jeden funkčný gén a počet génov spojenia odpovedá maximálnej arite použitých funkcií. Ak sú všetky funkcie binárne, funkčné bloky sú reprezentované trojicami, kde prvá hodnota značí identifikátor funkcie (funkčný gén), ktorú funkčný blok vykonáva a zvyšné dve predstavujú parametre funkcie funkčného bloku. Parametre funkcie určujú hlavný vstup programu, prípadne výstup niektorého z predchádzajúcich funkčných blokov, s ktorého hodnotami daný funkčný blok počíta, kvôli čomu sú často označované ako gény spojenia. Príklad chromozómu v CGP, ktorý odpovedá topológii z obrázku 4.3 je znázornený na obrázku 4.4. Podčiarknuté hodnoty predstavujú funkčné gény jednotlivých funkčných členov. Posledný prvok chromozómu, osamotený gén spojenia, predstavuje výstup programu.

$$(\underline{8},0,1)(\underline{2},1,2)(\underline{3},3,2)(\underline{1},0,1)(\underline{5},5,4)(\underline{8},5,6)(\underline{5},0,8)(\underline{2},5,8)(8)$$

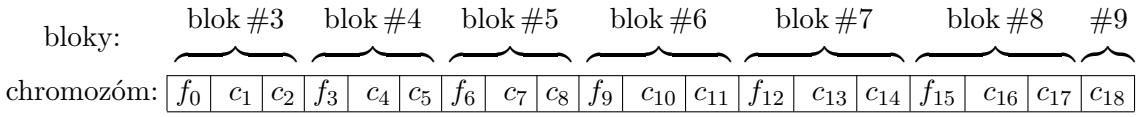
Obr. 4.4: Chromozóm v CGP

V katrézskom genetickom programovaní nie je použitý exploratórny genetický operátor kríženia. Operátor mutácie zastáva túto funkciu s využitím neutrálnych mutácií. Neutrálne mutácie sú také, ktoré nemajú vplyv na fenotyp daného chromozómu. Pri náhodnej mutácii génov v chromozóme, môže nastať situácia, keď bude mutovaný gén niektorého z neaktívnych funkčných blokov. Ak takáto operácia nastane niekoľko krát po sebe, môže v neaktívnej časti chromozómu vzniknúť podstrom, ktorého vlastnosti neboli dosiaľ odhalené. Pri mutácii, ktorá neaktívny podstrom začlení medzi aktívne funkčné bloky pôsobí samotný operátor mutácie exploratórnu činnosť. Predmetom mutácie v kartézskom genetickom programovaní sú funkcie jednotlivých funkčných blokov a prepojenia medzi nimi. [9] [13]

4.6 Nastavenia reprezentácie CGP

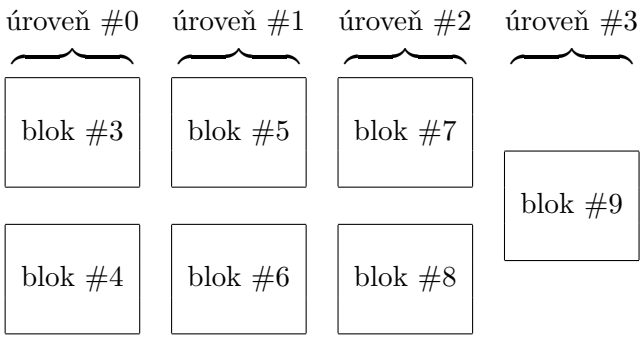
Na rozdiel od predchádzjúcich reprezentácií je chromozóm reprezentácie CGP veľmi obtiažne transformovať na homogénny. Implementovaná reprezentácia CGP používa binárne funkcie, čo spolu s identifikárom funkcie predstavuje trojicu pre daný funkčný blok. Výstupná hodnota v prípade celulárneho automatu predstavuje práve jeden gén chromozómu. Problém pri transformovaní chromozómu CGP na homogénny nastáva pri použití parametru *l-back*, kvôli ktorému môžu mať gény prepojenia rôznych funkčných blokov rôzny počet platných hodnôt.

Pre potreby mutácie a inicializácie génov chromozómu je v implementácii CGP chromozóm rozdelený podľa niekoľkých faktorov. Základné rozdelenie chromozómu je rozdelenie jeho génov medzi funkčné a výstupné bloky. Každej trojici génov chromozómu je podľa ich indexu pridelený blok. Zvyšná bunka na konci chromozómu predstavuje výstup programu a má pridelený samostatný výstupný blok. Príklad rozdelenia buniek chromozómu do blokov je uvedený na obrázku 4.5.



Obr. 4.5: Rozdelenie chromozómu do funkčných a výstupných blokov v CGP

V chromozóme na obrázku 4.5 sú gény rozdelené na dva druhy. Gény s popisom f označujú identifikátor funkcie a gény s popisom c označujú spojenia medzi blokmi. Indexy popisov génov predstavujú pozíciu génu v chromozóme. Funkčné bloky sú indexované od čísla 3, vďaka čomu môžu byť medzi prepojenia blokov jednoducho zaradené aj hlavné vstupy programu (troj-okolie skúmanej bunky) ako bloky na indexoch 0, 1 a 2. Výstupný blok obsahuje posledný gén chromozómu, udávajúci spojenie s funkčnými blokmi programu, ktoré budú použité pri výpočte nového stavu bunky automatu. Uvažujme ďalej CGP s dvoma riadkami a tromi stĺpcami. Chromozóm rozdelený na funkčné a výstupné bloky môžeme ďalej rozdeliť do úrovní podľa stĺpca, v ktorom sa blok nachádza. Toto delenie je zobrazené na obrázku 4.6.



Obr. 4.6: Rozdelenie chromozómu v CGP do úrovní podľa stĺpca bloku

Po prepočítaní indexov jednotlivých génov chromozómu na bloky a úrovne je pomerne jednoduché určiť hraničné hodnoty, ktoré môžu jednotlivé gény nadobúdať pri mutácii a inicializácii. V závislosti na hodnote *l-back* parametru je minimálna hodnota, ktorú môže gén nadobudnúť daná nasledujúcim vzťahom.

$$f_{min}(x) = \begin{cases} (l(x) - l\text{-}back) * R & \text{pre} \quad l(x) \geq l\text{-}back \\ 0 & \text{inak} \end{cases} \tag{4.5}$$

Maximálna hodnota génu je nezávislá na hodnote parametru *l-back* a je daná vzťahom 4.6.

$$f_{max}(x) = l(x) * R + O \tag{4.6}$$

Vo vyššie uvedených vzťahoch premenná x predstavuje index génu chromozómu. Funkcie $f_{min}(x)$ a $f_{max}(x)$ predstavujú minimálny a maximálny funkčný blok, na ktorý sa môže gén chromozómu s indexom x pripojiť. Funkcia $l(x)$ udáva úroveň bloku, ktorého súčasťou je gén s indexom x. Konštanty O a R predstavujú veľkosť okolia bunky automatu a počet riadkov kartézkeho genetického programovania. Operácie inicializácie a mutácie rozlišujú funkčné gény a gény spojenia. Funkčné gény nadobúdajú hodnoty z množiny používaných funkcií bez ohľadu na index daného génu v chromozóme. Zoznam použitých funkcií je uvedený v tabuľke 4.2 Pri inicializácií a mutácii génov spojenia sú použité hodnoty z rozsahu $f_{min}(x)$ až $f_{max}(x)$. Hodnoty na jednotlivých génoch prepojenia teda špecifikujú funkčné bloky, prípadne hlavné vstupy programu.

Výpočet nového stavu bunky spočíva v rekurzívnom prechádzaní funkčných blokov pripojených k výstupnému bloku. Ako prvý je vyhodnotený výstupný blok. V prípade, že je pripojený priamo na hlavný vstup programu, výpočet končí a skúmaná bunka automatu nadobúda zodpovedajúcu hodnotu bunky vo svojom okolí. V prípade, že je výstupný blok pripojený na jeden z funkčných blokov, nový stav bunky predstavuje výsledok výpočtu daného funkčného bloku. Funkčný blok obsahuje dva gény spojenia, ktoré môžu byť pripojené na hlavný vstup programu alebo na iné funkčné bloky, podľa čoho je funkcia funkčného bloku buď priamo vyhodnotená, alebo je vykonané rekurzívne vyhodnotenie pripojených funkčných blokov.

Veľkosť chromozómu je v katrézskom genetickom programovaní špecifikovaná parametrami udávajúcimi počet riadkov, stĺpcov, výstupných blokov programu a maximálnou aritou použitých funkcií. V implementácii sú počet výstupných blokov programu a arita funkcií pevne dané, a veľkosť chromozómu ovplyvňujú iba parametre počtu riadkov a stĺpcov. Implicitne sú parametre počtu riadkov a stĺpcov nastavené na hodnoty 1 a 100, čo spolu

Názov funkcie	Definícia funkcie
nulová funkcia	$\text{zero}(a,b) = 0$
funkcia identity	$\text{id}_a(a,b) = a$
inverzná funkcia	$\text{inv}_a(a,b) = 7 - a$
funkcia minima	$\min(a,b) = \begin{cases} a & (a < b) \\ b & \text{inak} \end{cases}$
funkcia maxima	$\max(a,b) = \begin{cases} b & (a < b) \\ a & \text{inak} \end{cases}$
funkcia súčtu	$\text{add}(a,b) = (a + b) \mod 8$
funkcia rozdielu	$\text{abssub}(a,b) = \begin{cases} a - b & (a > b) \\ b - a & \text{inak} \end{cases}$
funkcia priemeru	$\text{avg}(a,b) = (a + b) \div 2$
hraničná funkcia	$\text{treshold}(a,b) = \begin{cases} b & (a > 3) \\ a & \text{inak} \end{cases}$

Tabuľka 4.2: Zoznam použitých funkcií v reprezentácii CGP

s hodnotou *l-back* parametru rovnou počtu stĺpcov umožňuje maximálne prepojenie v rámci funkčných blokov. Veľkosť chromozómu je daná vzťahom $R * C * A + V$, kde R a C udávajú počet riadkov a stĺpcov programu a ich súčin udáva počet funkčných blokov programu, V je počet výstupných blokov a premenná A predstavuje šírku funkčného bloku (počet génov spojenia, daný aritou funkcie, a identifikačné číslo funkcie).

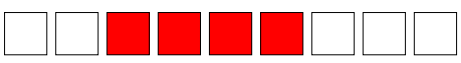
Kapitola 5

Návrh experimentov

Jednou z typických úloh („benchmarkových“ problémov) pre celulárne automaty je problém výpočtu druhej mocniny. V práci je riešenie tohto problému hľadané automaticky pomocou pokročilých techník reprezentácie prechodovej funkcie celulárneho automatu v kombinácii s evolučnými technikami. Platformou pre riešenie daného problému je osemstavový jednorozmerný celulárny automat s cyklickými okrajovými podmienkami a okolím bunky s polomerom 1. V rámci evolúcie nie je možné vyhodnotiť fitness chromozómu pre všetky hodnoty z nekonečnej množiny $\mathbb{N}$. Z toho dôvodu je množina vstupných hodnôt, s ktorými genetický algoritmus pracuje obmedzená na niekoľko málo prvkov z množiny $\mathbb{N}$. V experimentoch sú použité tri rôzne vstupné množiny obsahujúce hodnoty 2 až 5, 2 až 6 a 2 až 7. Každé evolučne odhalené riešenie teda predstavuje prechodovú funkciu celulárneho automatu, ktorá dokáže správne vypočítať druhú mocninu každej hodnoty z množiny vstupných hodnôt genetického algoritmu. Aby bolo možné prehlásiť nájdené riešenie za všeobecné, musí byť toto riešenie overené na množine hodnôt obsahujúcej čísla, s ktorými genetický algoritmus nepočítal. Na určenie všeobecnosti riešenia je v práci použitá množina s hodnotami 2 až 20. Pri výbere množiny vstupných hodnôt má veľký význam najväčšia zvolená hodnota, podľa ktorej je určená šírka celulárneho automatu.

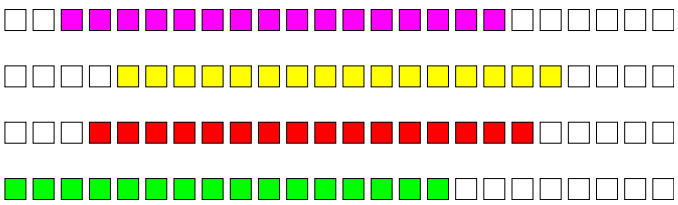
5.1 Interpretácia problému druhej mocniny

Problém druhej mocniny je v automate interpretovaný ako vývoj automatu v čase z počiatočnej konfigurácie do niektorej z platných koncových konfigurácií. Maximálny počet krokov, ktorý môže automat pri vývoji vykonať je závislý na maximálnej vstupnej hodnote n a je daný vzťahom $2 * n^2$. Počiatočná konfigurácia automatu má jednotný tvar a kóduje vstupnú celočíselnú hodnotu, ktorej mocninu chceme odhaliť. Pre celočíselnú hodnotu x obsahuje počiatočná konfigurácia automatu postupnosť buniek v stave 1 o dĺžke x, doplnenú bunkami v nulovom stave. Na obrázku 5.1 je zobrazená počiatočná konfigurácia automatu pre vstupnú hodnotu 4. Biele bunky predstavujú bunky v nulovom stave, bunky v stave 1 sú znázornené červenou.



Obr. 5.1: Počiatočná konfigurácia celulárneho automatu so vstupnou hodnotou 4

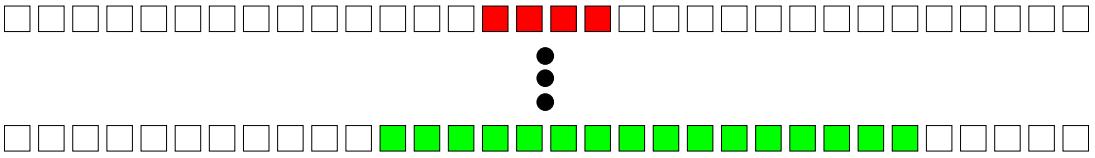
Množina platných koncových konfigurácií automatu je iná pre každú vstupnú hodnotu. Koncová konfigurácia patrí do množiny platných koncových konfigurácií pre nejakú počiatočnú konfiguráciu kódujúcu vstupnú hodnotu x, ak sa v nej vyskytuje postupnosť buniek v rovnakom nenulovom stave dlhá práve x^2 buniek. Zvyšné bunky koncovej konfigurácie automatu musia byť v nulovom stave. Všeobecne je pri takejto interpretácií počet koncových konfigurácii automatu nezávislý na vstupnej hodnote a je daný ako súčin šírky automatu a počtu stavov buniek automatu. V kombinácii s genetickým algoritmom v sekcii 3.4 je však tento počet obmedzený spôsobom, akým fitness funkcia vyhodnocuje správnosť výsledku. Fitness funkcia nepočíta s riešeniami, ktoré by „presahovali“ cez okraj automatu. V takom prípade je počet koncových konfigurácií závislý na vstupnej hodnote a je daný vzťahom $(w - x^2) * s$, kde w je šírka automatu, x je vstupná hodnota a s predstavuje počet stavov, ktoré môžu bunky v automate nadobúdať. Na obrázku 5.2 sú znázornené štyri vybrané koncové konfigurácie pre počiatočnú konfiguráciu celulárneho automatu kódujúcu vstupnú hodnotu 4.



Obr. 5.2: Vybrané platné koncové konfigurácie celulárneho automatu pre vstupnú hodnotu 4

5.2 Prípadová štúdia: počiatočná konfigurácia v strede

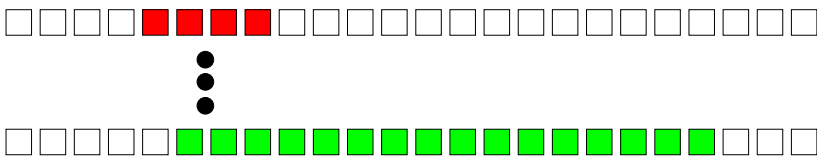
V predchádzajúcej sekcii bola popísaná interpretácia počiatočnej konfigurácie celulárneho automatu pre problém výpočtu druhej mocniny. Počiatočná konfigurácia bola definovaná ako postupnosť buniek automatu v stave 1 s dĺžkou zodpovedajúcou vstupnej hodnote, doplnená bunkami v nulovom stave. V tejto sekcii je popísaný konkrétny variant počiatočnej konfigurácie, kde sa postupnosť buniek v stave 1 nachádza v strede celulárneho automatu a tvorí tak symetrickú počiatočnú konfiguráciu. Táto konfigurácia bola zvolená za prípadovú štúdiu za účelom určenia úspešnosti evolúcie v prípade, kde je celulárny automat dostatočne široký a umožňuje tak genetickému algoritmu skúmať riešenia vyvíjajúce sa oboma smermi zo stredovej počiatočnej konfigurácie. Z toho dôvodu je šírka automatu použitého v experimentoch tejto prípadovej štúdie daná vzťahom $2 * n^2$, kde n je maximálna vstupná hodnota. Zvolená šírka umožnuje automatu vytvoriť koncovú konfiguráciu s dĺžkou nenulovej postupnosti n^2 na oboch stranách, prípadne v strede celulárneho automatu. Počiatočná konfigurácia v strede automatu a jedna z možných koncových konfigurácií pre maximálnu vstupnú hodnotu 4 sú zobrazené na obrázku 5.3.



Obr. 5.3: Počiatočná konfigurácia a jedna z platných koncových konfigurácií pre prípadovú štúdiu s počiatočnou konfiguráciou v strede automatu

5.3 Prípadová štúdia: počiatočná konfigurácia vľavo

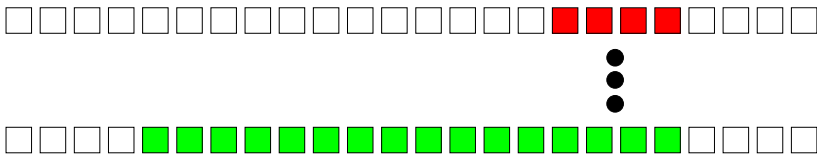
Ďalšou skúmanou prípadovou štúdiou je počiatočná konfigurácia celulárneho automatu s postupnosťou buniek v stave 1 vľavo. Táto prípadová štúdia bola v niekoľkých smeroch zvolená s opačným zámerom ako predchádzajúca prípadová štúdia. Umiestnenie postupnosti buniek v stave 1 na kraj automatu predstavuje pre evolúciu pomerne významné obmedzenie. Riešenia, ktoré počítajú s vývojom automatu na ľavú stranu sú pre túto prípadovú štúdiu nevyhovujúce, kvôli spôsobu akým fitness funkcia určuje fitness chromozómu. Zároveň však takéto obmedzenie umožňuje zmenšiť šírku automatu vďaka čomu je vývoj automatu výpočtovo menej náročný a genetický algoritmus tak za rovnaký čas stihne preskúmať viac generácií riešení. V tejto prípadovej štúdii je šírka celulárneho automatu daná vzťahom $2 * n + n^2$, kde n je maximálna vstupná hodnota. Maximálna dĺžka nenulovej sekvencie v koncovej konfigurácii je daná vzťahom n^2. Zvyšných $2 * n$ buniek automatu slúži ako obojstranná „výplň“ nenulovej postupnosti, bez ktorej by nájdené riešenia kolidovali sami so sebou kvôli cyklickým okrajovým podmienkam automatu. Počiatočná konfigurácia v ľavej časti automatu a jedna z možných koncových konfigurácií pre maximálnu vstupnú hodnotu 4 sú zobrazené na obrázku 5.4.



Obr. 5.4: Počiatočná konfigurácia a jedna z platných koncových konfigurácií pre prípadovú štúdiu s počiatočnou konfiguráciou vľavo

5.4 Prípadová štúdia: počiatočná konfigurácia vpravo

Poslednou skúmanou prípadovou štúdiou je počiatočná konfigurácia celulárneho automatu s postupnosťou buniek v stave 1 vpravo. Šírka automatu tejto štúdie je identická s predchádzajúcou prípadovou štúdiou. V experimentoch je uvedená práve za účelom určenia vplyvu spôsobu vyhodnotenia fitness funkcie (vyhodnocovanie nenulovej postupnosti z ľava do prava) na úspešnosť genetického algoritmu. Počiatočná konfigurácia v pravej časti automatu a jedna z možných koncových konfigurácií pre maximálnu vstupnú hodnotu 4 sú zobrazené na obrázku 5.5.



Obr. 5.5: Počiatočná konfigurácia a jedna z platných koncových konfigurácií pre prípadovú štúdiu s počiatočnou konfiguráciou vpravo

Kapitola 6

Experimenty

Táto kapitola je zameraná na popis experimentov s jednotlivými reprezentačnými technikami prechodovej funkcie celulárneho automatu. Kapitola je zároveň ústrednou časťou práce. Kapitola obsahuje popis parametrov a prípravných krokov pre každý z vykonaných experimentov, rovnako ako aj štatistické zhrnutie výsledkov a prehľad zaujímavých riešení. Ak nie je uvedené inak, v experimentoch sú implicitne použité nasledujúce hodnoty parametrov. Pravdepodobnosť mutácie je sto percent. Pravdepodobnosť kríženia je nula percent. Počet mutovaných buniek chromozómu je dva. V rámci turnajového výberu sú použité štyri jedince. Veľkosť populácie je osem jedincov a maximálny počet generácií v genetickom algoritme je desať miliónov. Je použitý osemstavový jednorozmerný celulárny automat, s veľkosťou polomeru okolia bunky jeden. Experimenty boli vykonané serveri Salomon v rámci projektu `IT4I`. Pre každé nastavenie experimentu bolo vykonaných 96 nezávislých spustení. Vykonané experimenty mali časové obmedzenie jednu hodinu.

Výsledky experimentov pre vybrané prípadové štúdie sú zhrnuté v samostatných tabuľkách. Štruktúra tabuľky je nasledujúca. Prvý stĺpec tabuľky popisuje rôzne nastavenia hodnoty skúmaného parametru. Druhý a tretí stĺpec tabuľky obsahujú experimentálne získané výsledky pre dve rôzne množiny vstupných hodnôt. Výsledok úspešnosti evolúcie udáva počet evolučne nájdených riešení. Úspešnosť evolúcie je vyjadrená počtom nájdených riešení zo všetkých 96 spustení, čo približne zodpovedá percentuálnej hodnote. Hodnota v zátvorke udáva, koľko z nájdených riešení je všeobecných. Ďalej sú v druhom a treťom stĺpci tabuľky uvedené štatistické údaje o priemernej generácii, v ktorej bolo riešenie nájdené a štandardnej odchýlke súboru nájdených riešení.

6.1 Experimenty s tabuľkovou reprezentáciou

Nad tabuľkovou reprezentáciou bolo vykonaných 36 sád experimentov za účelom odhalenia vplyvu počtu mutovaných génov chromozómu na úspešnosť evolúcie. Experimenty sa líšili v parametroch počiatočnej konfigurácie automatu, maximálnej vstupnej hodnoty a počtu mutovaných buniek v rámci operácie mutácie. V experimentoch boli uvažované tri uvedené prípadové štúdie počiatočnej konfigurácie celulárneho automatu tak, ako to znázorňujú obrázky 5.3, 5.4 a 5.5. Použité boli dve sady vstupných hodnôt, konkrétne vstupné hodnoty 2 až 5 a hodnoty 2 až 6. Pre počet mutovaných génov bolo v experimentoch použitých až šesť variant, konkrénte 2, 4, 6, 8, 10, respektíve 12 mutovaných génov. Nad každou kombináciou variant uvedených parametrov bolo vykonaných 96 nezávislých behov programu. Tabuľky 6.1, 6.2 a 6.3 obsahujú súhrn výsledkov experimentov s tabuľkovou reprezentáciou.

mutovaných génov	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
2	41 (0)	2492516	2335955	32 (0)	1245682	1006163
4	48 (0)	1982556	2075361	32 (0)	1129177	943649
6	60 (0)	1775669	1953334	49 (0)	868830	864796
8	63 (0)	1774777	1962763	33 (0)	989026	796823
10	67 (0)	1461817	1847330	43 (0)	842753	874318
12	68 (0)	1636711	1897578	39 (0)	889966	852499

Tabuľka 6.1: Zhrnutie výsledkov experimentov s tabuľkovou reprezentáciou pre prípadovú štúdiu s počiatočnou konfiguráciou v strede s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

mutovaných génov	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
2	30 (0)	2376041	2556210	12 (0)	2519892	1472830
4	41 (0)	2440191	2117771	27 (0)	2295902	1472830
6	48 (0)	3172603	2914164	43 (0)	1630507	1289053
8	61 (0)	2560853	2218355	36 (0)	1446799	1120732
10	65 (0)	2853650	2847595	42 (0)	1586912	1447934
12	65 (0)	2308791	2596221	33 (0)	1608272	1320027

Tabuľka 6.2: Zhrnutie výsledkov experimentov s tabuľkovou reprezentáciou pre prípadovú štúdiu s počiatočnou konfiguráciaou vľavo s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

mutovaných génov	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
2	22 (0)	2724526	2772104	29 (0)	1764192	1264933
4	44 (0)	2747732	2554881	42 (0)	1868895	1337320
6	46 (0)	3105901	2942461	51 (0)	1608854	1366346
8	57 (0)	2935866	2623011	35 (0)	1701109	1470842
10	68 (0)	2806698	2623011	52 (0)	1540896	1252334
12	59 (0)	2426152	2441631	51 (0)	1457871	1414103

Tabuľka 6.3: Zhrnutie výsledkov experimentov s tabuľkovou reprezentáciou pre prípadovú štúdiu s počiatočnou konfiguráciou vpravo s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

Z výsledkov je vidieť, že úspešnosť evolúcie rastie s počtom mutovaných buniek chromozómu, a ustáli sa pri mutovaní desiatich až dvanástich génov. Úspešnosť evolúcie je podľa očakávania menšia pri použití väčšej sady vstupných hodnôt. Je však zaujímavé, že pri použití väčšej sady vstupných hodnôt sú riešenia objavené skôr. Tiež je zaujímavé, že pri použití sady vstupných hodnôt 2 až 5 dosahuje evolúcia najväčšiu úspešnosť pri inicializácii automatu v strede, no pri vstupných hodnotách 2 až 6 je najúspešnejšia pri inicializácii vľavo, respektíve vpravo.

Relatívne vysoká úspešnosť evolúcie pri použití desiatich a dvanástich mutovaných buniek podnecuje k experimentom, ktoré by mohli odhaliť aspoň jedno všeobecné riešenie. V nasledujúcej sade experimentov je skúmaný vplyv väčšej sady vstupných hodnôt na úspešnosť evolúcie, hlavne za účelom odhalenia všeobecného riešenia. Množina vstupných hodnôt obsahuje čísla 2 až 7. Ďalšie skúmané parametre zahŕňajú počet mutovaných buniek chromozómu a počiatočnú konfiguráciu automatu. Výsledky sady experimentov so vstupnými hodnotami 2 až 7 sú zhrnuté v tabuľke 6.4.

počiatočná konfigurácia	vľavo	v strede	vpravo	vľavo	v strede	vpravo
úspešnosť evolúcie	15 (0)	22 (0)	20 (0)	21 (0)	13 (0)	21 (0)
priemerná generácia	1139060	610537	1024178	976409	534346	816414
štandardná odchýlka	619421	432213	600025	549395	363601	522311

Tabuľka 6.4: Zhrnutie výsledkov experimentov s tabuľkovou reprezentáciou; množina vstupných hodnôt 2 až 7 s počtom mutovaných buniek 10 (vľavo) a 12 (vpravo)

6.2 Experimenty s reprezentáciou CMR

Experimenty s reprezentáciou podmienkových pravidiel boli založené na výsledkoch dosiahnutých Bidlom v jeho článku [4]. Testované parametre výpočtu zahŕňali počet podmienkových pravidiel, veľkosť vstupnej množiny a počiatočnú konfiguráciu automatu. Pre parameter počtu podmienkových pravidiel boli vybrané hodnoty 20 a 30, ktoré Bidlo uviedol ako najúspešnejšie z hľadiska evolučného výpočtu. Skúmané veľkosti vstupnej množiny boli, podobne ako pri tabuľkovej reprezentácii, dve, konkrétne množiny vstupných hodnôt od 2 do 5 a od 2 do 6. Počiatočné konfigurácie automatu boli taktiež rovnaké ako pri tabuľkovej reprezentácii. Tabuľky 6.5, 6.6 a 6.7 obsahujú súhrn výsledkov testov nad reprezentáciou CMR. Ich štruktúra je podobná tabuľkám 6.1 až 6.3 pre experimenty s tabuľkovou reprezentáciou.

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
20	62 (2)	214470	252331	31 (1)	260432	255950
30	55 (1)	104270	113652	27 (0)	220846	172219

Tabuľka 6.5: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou v strede s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

Vysoká úspešnosť evolúcie a relatívne malý prehľadávaný priestor riešení zvyšujú pravdepodobnosť odhalenia všeobecného riešenia. Dopad veľkosti prehľadávaného priestoru je vidieť pri porovnaní úspešností evolučných behov s dvadsiatimi a tridsiatimi podmienkovými pravidlami. V oboch prípadoch je úspešnosť evolúcie veľmi podobná, no zmenšenie prehľadávaného priestoru použitím menšieho počtu podmienkových pravidiel má výrazný dopad na pravdepodobnosť odhalenia všeobecného riešenia. Na základe tejto tendencie boli vypracované experimenty s osemnástimi, šesnástimi a štrnástimi prechodovými pravidlami, ktorých výsledky sú uvedené v tabuľkách 6.8, 6.9 a 6.10.

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
20	63 (2)	154918	187179	35 (4)	221614	193316
30	56 (0)	156262	208528	37 (0)	298975	277542

Tabuľka 6.6: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vľavo s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
20	64 (1)	133428	177945	39 (1)	240556	192038
30	60 (0)	125686	164660	37 (0)	193260	202739

Tabuľka 6.7: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vpravo s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
18	45 (1)	110223	96654	15 (2)	123844	98082
16	40 (1)	140182	169082	18 (0)	78776	51702
14	46 (2)	159330	150170	19 (3)	137609	102969

Tabuľka 6.8: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou v strede s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
18	38 (0)	189337	238979	10 (0)	190718	90673
16	47 (4)	171642	200256	16 (0)	185750	113556
14	36 (0)	306403	361664	14 (3)	138904	156358

Tabuľka 6.9: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vľavo s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

Experimenty s menej ako dvadsiatimi podmienkovými pravidlami potvrdzujú predpoklad o vplyve veľkosti prehľadávaného priestoru na úspešnosť evolučného algoritmu. Pri použití šestnástich a štrnástich podmienkových pravidiel stúpla úspešnosť evolúcie oproti úspešnosti s dvadsiatimi podmienkovými pravidlami v priemere asi o šesť percent. Je zaujímavé, že napriek vyššej úspešnosti sa evolúcii nepodarilo odhaliť žiadne všeobecné riešenie pri použití šestnástich podmienkových pravidiel so sadou vstupných hodnôt 2 až 7. Je tiež zaujímavé, že pri použití štrnástich podmienkových pravidiel s rovnakou sadou vstupných hodnôt bolo zo všetkých odhalených riešení až 15, respektíve 20 percent všeobecných. Nie-

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
18	39 (0)	224867	200271	11 (0)	189541	201936
16	37 (0)	216479	188637	9 (0)	170592	98904
14	43 (0)	315748	297063	5 (0)	159564	54067

Tabuľka 6.10: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vpravo s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

ktoré z nájdených všeobecných riešení využívalo len sedem stavov bunky automatu. Na základe týchto výsledkov boli navrhnuté experimenty s počtom podmienkových pravidiel dvanásť, desať a osem, ktorých účelom je zistiť, či evolúcia dokáže nájsť všeobecné riešenia problému druhej mocniny v celulárnych automatoch s menej ako siedmimi rôznymi stavmi bunky. Množiny vstupných hodnôt pre navrhnuté experimenty obsahujú čísla 2 až 6 a 2 až 7 s počiatočnou konfiguráciou automatu vľavo, v strede a vpravo. Tabuľky 6.11 až 6.13 obsahujú výsledky vykonaných experimentov.

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
12	49 (0)	124569	114466	16 (4)	54559	39542
10	30 (1)	250169	250899	17 (6)	143526	88475
8	28 (1)	224187	224887	8 (4)	160280	77464

Tabuľka 6.11: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou v strede s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
12	43 (1)	276069	239355	13 (4)	192840	168153
10	31 (3)	336648	286817	13 (2)	214913	153836
8	27 (6)	470023	454979	9 (6)	152460	190098

Tabuľka 6.12: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vľavo s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

Úspešnosť experimentov s reprezentáciou CMR s počtom podmienkových pravidiel dvanásť je podobná úspešnosti predchádzajúcich experimentov. Pri desiatich, respektíve ôsmich podmienkových pravidlách úspešnosť evolúcie začína klesať a dostáva sa až pod 30 % pri použití vstupných hodnôt 2 až 6. Pomer nájdených voči nájdeným všeobecným riešeniam však drasticky stúpa, najmä pri vstupných hodnotách 2 až 7. Experimenty s ôsmimi podmienkovými pravidlami tiež odhalili všeobecné riešenia problému druhej mocniny v celulárnom automate s použitím iba piatich rôznych stavov bunky automatu. Tieto riešenia sú uvedené v prílohe B v tabuľke B.5. Náhľad na vývoj automatu k uvedeným riešeniam je uvedený na

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
12	31 (0)	333235	195575	9 (0)	276020	190607
10	29 (0)	420922	353216	1 (0)	336648	286817
8	14 (0)	432178	304525	0 (0)	-	-

Tabuľka 6.13: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vpravo s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

obrázku A.6 v prílohe A. Odhalenie všeobecného riešenia s použitím piatich stavov bunky automatu inšpirovalo poslednú sadu epxerimentov nad reprezentáciou CMR. Sada obsahuje experimenty s množinou vstupných hodnôt 2 až 6 a 2 až 7, počiatočnou konfiguráciou vľavo, vpravo a v strede automatu, s piatimi stavmi buniek automatu a s počtom podmienkových pravidiel šesť a päť. Získané výsledky sú zhrnuté v tabuľkách 6.14 a 6.15.

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
6	4 (4)	421050	316470	2 (2)	197223	122882
5	2 (2)	20459	4078	0 (0)	-	-

Tabuľka 6.14: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou v strede s počtom stavov buniek automatu 5 a s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

počet pravidiel	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
6	4 (4)	217949	225584	3 (3)	65700	13053
5	3 (3)	1330365	906807	0 (0)	-	-

Tabuľka 6.15: Zhrnutie výsledkov experimentov s reprezentáciou CMR pre prípadovú štúdiu s počiatočnou konfiguráciou vľavo s počtom stavov buniek automatu 5 a s množinou vstupných hodnôt 2 až 6 (vľavo) a 2 až 7 (vpravo)

Výsledky experimentov 6.14 a 6.15 vykazujú významný pokles úspešnosti evolučného algoritmu oproti predchádzajúcim experimentom. V prípadovej štúdii s počiatočnou konfiguráciou vpravo neboli nájdené žiadne riešenia. Tento trend nie je nečakaný, keďže každý stav automatu je zväčša výsledkom práve jedného, prípadne maximálne dvoch, podmienkových pravidiel, čo predstavuje pomerne veľké obmedzenie pri prehľadávaní tak veľkého priestoru riešení (pre automat s piatimi stavmi, trojokolím bunky, štyrmi podmienkami a CMR s piatimi pravidlami je veľkosť prehľadávaného priestoru $(5*(3+1))^{5*4} = 20^{20} \approx 2^{87}$). Na druhú stranu toto obmedzenie prispieva k pomeru nájdených a nájdených všeobecných riešení. Z výsledkov experimentov vyplýva, že každé nájdené riešenie bolo zároveň aj všeobecné. Zo zvyšných štatistických údajov experimentov nie je možné vyvodiť spoľahlivé závery, pretože v jednotlivých kategóriach sa nevyskytuje ani päť vzorkov riešení.

6.3 Experimenty s reprezentáciou CGP

Nad reprezentáciou CGP bolo, podobne ako pri tabuľkovej reprezentácii, vykonaných 36 sád experimentov za účelom odhalenia vplyvu počtu mutovaných génov chromozómu na úspešnosť evolúcie. V experimentoch bol uvažovaný počet mutovaných génov 2, 4, 6, 8, 10 a 12 pri vstupných hodnotách z rozsahu 2 až 5 a 2 až 6. Experimenty boli vykonané pre každú uvedenú prípadovú štúdiu počiatočnej konfigurácie celulárneho automatu. Tabuľky 6.16 a 6.17 obsahujú súhrn výsledkov experimentov.

mutovaných génov	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
4	2 (0)	166042	76829	0 (0)	-	-
8	2 (0)	180791	78050	0 (0)	-	-
12	3 (0)	56617	18505	0 (0)	-	-

Tabuľka 6.16: Zhrnutie výsledkov experimentov s reprezentáciou CGP pre prípadovú štúdiu s počiatočnou konfiguráciou v strede s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

mutovaných génov	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka	úspešnosť evolúcie	priemerná generácia	štandardná odchýlka
4	3 (0)	209030	93083	0 (0)	-	-
8	2 (0)	130888	103830	0 (0)	-	-
10	4 (0)	87649	31721	0 (0)	-	-

Tabuľka 6.17: Zhrnutie výsledkov experimentov s reprezentáciou CGP pre prípadovú štúdiu s počiatočnou konfiguráciou vľavo s množinou vstupných hodnôt 2 až 5 (vľavo) a 2 až 6 (vpravo)

Pre hodnoty mutovaných buniek, ktoré nie sú v tabuľkách 6.16 a 6.17 uvedené sa evolúcii nepodarilo nájsť riešenie. Tabuľka pre prípadovú štúdiu s počiatočnou konfiguráciou automatu vpravo nie je uvedená z rovnakého dôvodu.

6.4 Diskusia k vykonaným experimentom

Na základe dosiahnutých výsledkov sa zdá, že tabuľková reprezentácia prechodovej funkcie celulárneho automatu nie je vhodná pre potreby evolučného algoritmu. Výpočtová nenáročnosť tabuľkovej reprezentácie umožňuje evolúcii vyhodnotiť až desať miliónov generácii. Napriek tomu neboli pomocou tabuľkovej reprezentácie nájdené žiadne všeobecné riešenia problému druhej mocniny, čo je zrejme dôsledkom značnej veľkosti prehľadávaného priestoru riešení. Nájdené riešenia vykazujú chaotické, až náhodné správanie pred dosiahnutím ustáleného stavu, čo je zrejme hlavným dôvodom, prečo pri testovaní inej sady vstupných hodnôt riešenia neposkytujú očakávané výsledky. Príklad riešenia nájdeného pomocou tabuľkovej reprezentácie je znázornený na obrázku A.1. Riešením by mohla byť redukcia prehľadávaného priestoru zmenou počtu stavov celulárneho automatu. Pri použití päťstavového automatu je veľkosť prehľadávaného priestoru riešení $5^{5*5*5} \approx 2^{2.32^{125}} \approx 2^{290}$. Počet

všeobecných riešení problému druhej mocniny je však v päťstavovom celulárnom automate výrazne nižší. Pri porovnaní úspešnosti evolúcie na základe skúmaných prípadových štúdií sa pri použití vstupnej množiny s hodnotami 2 až 5 najviac osvedčilo umiestnenie nenulovej postupnosti počiatočnej konfigurácie do stredu automatu. Pri tomto nastavení bol automatický návrh riešení úspešnejší až o 10 % v porovnaní s ostatnými prípadovými štúdiami. Počiatočné konfigurácie s nenulovou postupnosťou na krajoch automatu boli úspešnejšie pri použití vstupnej množiny s hodnotami 2 až 6.

Pri použití CMR, evolúcia v priemere nachádza riešenia okolo sto až dvesto tisícej generácie, čo predstavuje značný pokrok oproti tabuľkovej reprezentácii. V porovnaní s tabuľkovou reprezentáciou je reprezentácia CMR výpočtovo náročnejšia. Pri reprezentácii podmienkových pravidiel prepočíta evolúcia asi milión generácií. Zo získaných výsledkov je možné vyvodiť záver, že reprezentácia prechodovej funkcie celulárneho automatu zakódovaná v podobe CMR je vhodná pre použitie v kombinácii s evolučnými algoritmami. Vybrané všeobecné riešenia problému druhej mocniny sú znázornené na obrázkoch A.2 až A.7. Na obrázkoch A.6 a A.7 je znázornený vývoj celulárnych automatov, ktoré riešia problém druhej mocniny s využitím len piatich rôznych stavov. Ďalšie zaujímavé riešenie je na obrázku A.4 vľavo. Jeho vývoj do koncovej konfigurácie pre vstupnú hodnotu 5 trvá 33 krokov. Analyticky navrhnutý celulárny automat Wolframa potrebuje na výpočet druhej mocniny pre vstupnú hodnotu 5 až 78 krokov [15]. Na základe výsledkov experimentov je možné odhadnúť optimálny počet podmienkových pravidiel použitých pri výpočtoch s viacstavovým celulárnym automatom na asi dvojnásobok počtu stavov v celulárnom automate. Hlavná výhoda reprezentácie CMR oproti tabuľkovej reprezentácii spočíva v masívnej redukcii prehľadávaného priestoru riešení. Pri šestnástich podmienkových pravidlách, štyroch relačných operátoroch, troj-okolí skúmanej bunky a ôsmich stavoch automatu je redukcia z priestoru prehľadávaného tabuľkovou reprezentáciou ($8^{8*8*8} = 8^{512} = 2^{1536}$) na priestor prehľadávaný reprezentáciou CMR ($((8 * 4)^{16*(3+1)} = 32^{64} = 2^{320}$) až $\frac{2^{1536}}{2^{320}} = 2^{1536-320} = 2^{1216}$ násobná. Nevýhodou stále ostáva náročnejší výpočet, ktorý však je možné paralelizovať a významne tak skrátiť dobu výpočtu. Z hľadiska prípadových štúdií boli počiatočné konfigurácie s nenulovou postupnosťou v ľavej a strednej časti automatu asi rovnako úspešné. Tretia prípadová štúdia s nenulovou postupnosťou v počiatočnej konfigurácii vpravo bola oproti ostatným dvom štúdiám menej úspešná ako z hľadiska nájdených riešení, tak aj z hľadiska nájdených všeobecných riešení.

Kartézske genetické programovanie sa javí ako veľmi nevhodná technika reprezentácie prechodovej funkcie viacstavového jednorozmerného celulárneho automatu. Počet odhalených riešení počas jednotlivých experimentov nepresahuje ani 5 %, a veľmi často nie je evolučne odhalené žiadne riešenie. Bežne je CGP používané v kombinácii s evolučnou stratégiou $1+\lambda$ [13]. Pri použití tejto evolučnej techniky by reprezentácia CGP mohla dosahovať lepšie výsledky. Iným dôvodom neúspechu tejto reprezentácie môže byť nevhodná použitá množina funkcií. Pri experimentoch s CGP bolo v genetickom algoritme vyhodnotených asi štyristo tisíc generácií. Zmena parametrov počtu riadkov a stĺpcov mriežky CGP, prípadne *l-back* parametru, by mohla výpočet ďalej zjednodušiť a umožniť tak evolúcii vyhodnotenie väčšieho počtu generácií.

Kapitola 7

Záver

V kapitole 2 bol predstavený celulárny automat ako výpočtová platforma pre simuláciu rôznych fyzikálnych a biologických javov. Detailne bol preskúmaný elementárny celulárny automat, jeho vývoj a triedy správania. Výpočtová sila celulárnych automatov bola ukázaná na implementácii Turingovho stroja v automate Game of Life. Využitie celulárnych automatov v praxi bolo prezentované na modeloch korisť-predátor, excitačné médium a iných. Kapitola 3 predstavila problematiku genetických algoritmov a ich účinnosť pri riešení optimalizačných úloh. Boli definované dôležité pojmy z tejto oblasti, a znázornené vybrané algoritmy pre prácu s evolučnými technikami. V kapitole 4 boli špecifikované vybrané techniky reprezentácie prechodovej funkcie celulárneho automatu, konkrétne základná tabuľková reprezentácia, podmienkové pravidlá a kartézske genetické programovanie. Návrh experimentov a popis interpretácie problému druhej mocniny bol uvedený v kapitole 5. Hlavnú časť práce predstavovala kapitole 6, v ktorej boli zhrnuté výsledky experimentov vykonaných nad tabuľkovou reprezentáciou, reprezetnáciou CMR a reprezentáciou CGP za účelom porovnania daných techník z hľadiska vhodnosti pre automatický návrh programov pre celulárne automaty za použitia genetických algoritmov a odhalenia všeobecných riešení problému duhej mocniny.

V práci boli experimentálne odhalené niektoré zaujímavé vlastnosti reprezentačnej techniky CMR ako napríklad vplyv počtu podmienkových pravidiel na úspešnosť evolúcie. Technika sa ukázala ako veľmi vhodná pre automatický návrh s využitím evolučných techník. Hlavnou výhodou reprezentácie je jej schopnosť organizácie a zhlukovania stavov pomocou relačných operátorov, čo významne redukuje prehľadávaný priestor riešení. Tabuľková reprezentácia sa ukázala ako nevhodná pre použitie v kombinácii s evolučnými technikami, z dôvodu extrémnej veľkosti priestoru riešení. Štruktúry, ktoré sa pomocou tabuľkovej reprezentácie v automatoch vyvíjajú majú chaotický charakter, kvôli čomu je veľmi obtiažne odhaliť obecné riešenie daného problému. Na základe výsledkov experimentov s reprezentáciou CGP sa javí táto reprezentácia ešte menej vhodná ako tabuľková reprezentácia, čo môže byť spôsobené použitým genetickým algoritmom alebo použitou množinou funkcií. Práca by mohla byť rozšírená o porovnanie reprezentácií CMR a CGP pri použití evolučnej stratégie $1+\lambda$, prípadne o dôkladný prieskum vhodných funkcií pri riešení problému druhej mocniny vo viacstavovom celulárnom automate.

Literatúra

[1] Andre, D.; Bennett, F. H., III; Koza, J. R.: Discovery by Genetic Programming of a Cellular Automata Rule That is Better Than Any Known Rule for the Majority Classification Problem. In *Proceedings of the 1st Annual Conference on Genetic Programming*, Cambridge, MA, USA: MIT Press, 1996, ISBN 0-262-61127-9, s. 3–11.

[2] Bandini, S.; Mauri, G.; Serra, R.: Cellular automata: From a theoretical parallel computational model to its application to complex systems. *Parallel Computing*, ročník 27, č. 5, 2001: s. 539 – 553, ISSN 0167-8191, doi:http://dx.doi.org/10.1016/S0167-8191(00)00076-4, cellular automata: From modeling to applications.

[3] Bidlo, M.; Vašíček, Z.: Evolution of Cellular Automata Using Instruction-Based Approach. In *2012 IEEE World Congress on Computational Intelligence*, Institute of Electrical and Electronics Engineers, 2012, ISBN 978-1-4673-1508-1, s. 1060–1067.

[4] Bidlo, M.; Vašíček, Z.: Evolution of Cellular Automata with Conditionally Matching Rules. In *2013 IEEE Congress on Evolutionary Computation (CEC 2013)*, IEEE Computer Society, 2013, ISBN 978-1-4799-0452-5, s. 1178–1185.

[5] Ermentrout, G.; Edelstein-Keshet, L.: Cellular Automata Approaches to Biological Modeling. *Journal of Theoretical Biology*, ročník 160, č. 1, 1993: s. 97 – 133, ISSN 0022-5193, doi:http://dx.doi.org/10.1006/jtbi.1993.1007.

[6] Holland, J. H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992, ISBN 0262082136.

[7] Kvasnička, V.; Pospíchal, J.; Tiňo, P.: *Evolučné algoritmy*. Bratislava: Vydavateľstvo STU v Bratislave, 2000, ISBN 80-227-1377-5, 223 s.

[8] Lim, H.: Lattice-gas automaton simulations of simple fluid dynamical problems. *Mathematical and Computer Modelling*, ročník 14, 1990: s. 720 – 727, ISSN 0895-7177, doi:http://dx.doi.org/10.1016/0895-7177(90)90276-S.

[9] Miller, J. F.: *Cartesian Genetic Programming*. Berlin: Springer-Verlag Berlin Heidelberg, 2011, ISBN 978-3-642-17309-7, 346 s.

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

[11] Rendell, P.: A Universal Turing Machine in Conway's Game of Life. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, July 2011, s. 764–772, doi:10.1109/HPCSim.2011.5999906.

[12] Sekanina, L.: Biologií inspirované počítače. Evoluční design. [online], 2015.

[13] Sekanina, L.; Vašíček, Z.; Růžička, R.; aj.: *Evoluční hardware: Od automatického generování patentovatelných invencí k sebemodifikujícím se strojům*. Edice Gerstner, Academia, 2009, ISBN 978-80-200-1729-1, 328 s.

[14] Wolfram, S.: Statistical mechanics of cellular automata. *Rev. Mod. Phys.*, ročník 55, Júl 1983: s. 601–644, doi:10.1103/RevModPhys.55.601.

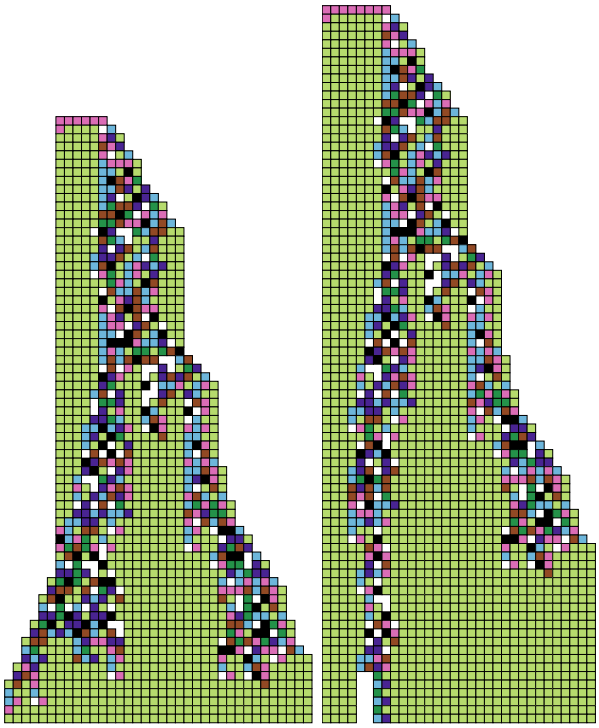
[15] Wolfram, S.: *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002, ISBN 1579550088, 1197 s.

Prílohy

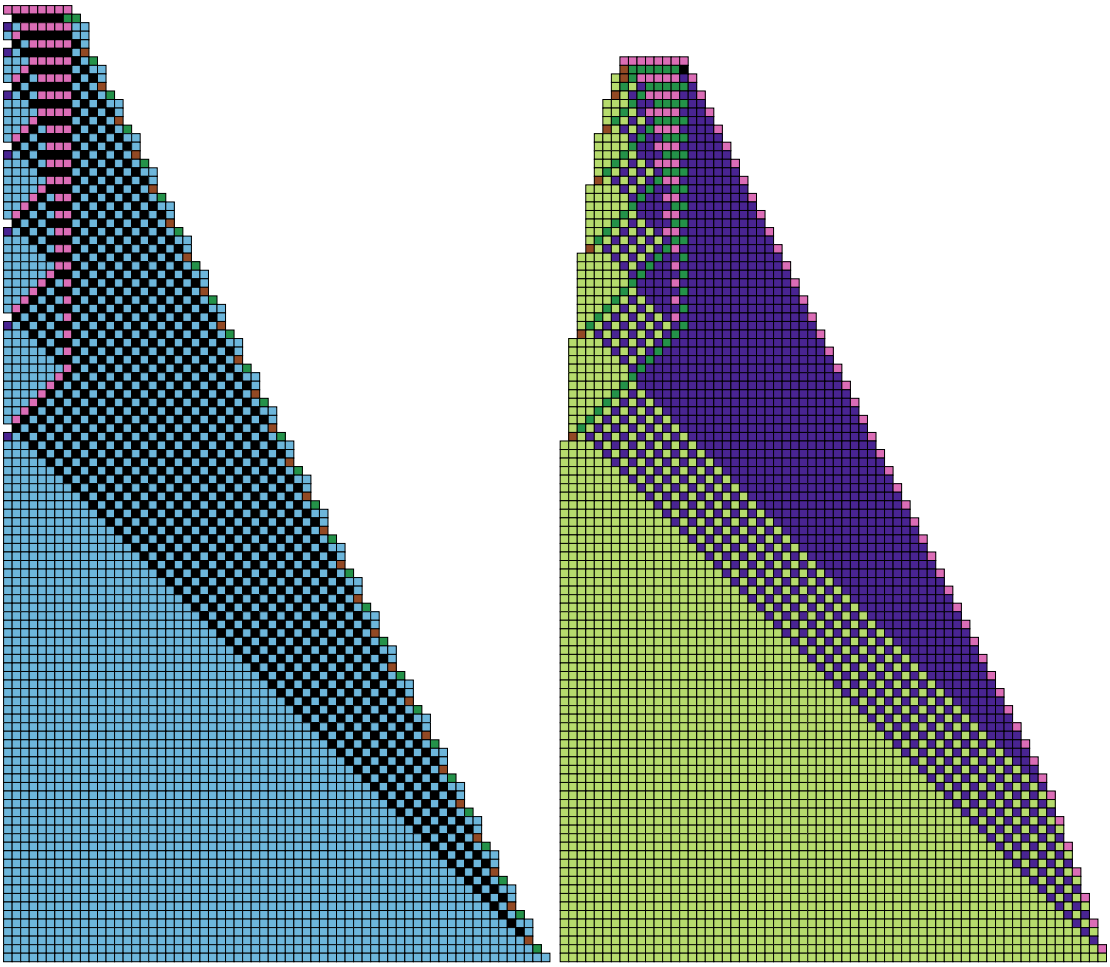
Zoznam príloh

Príloha A

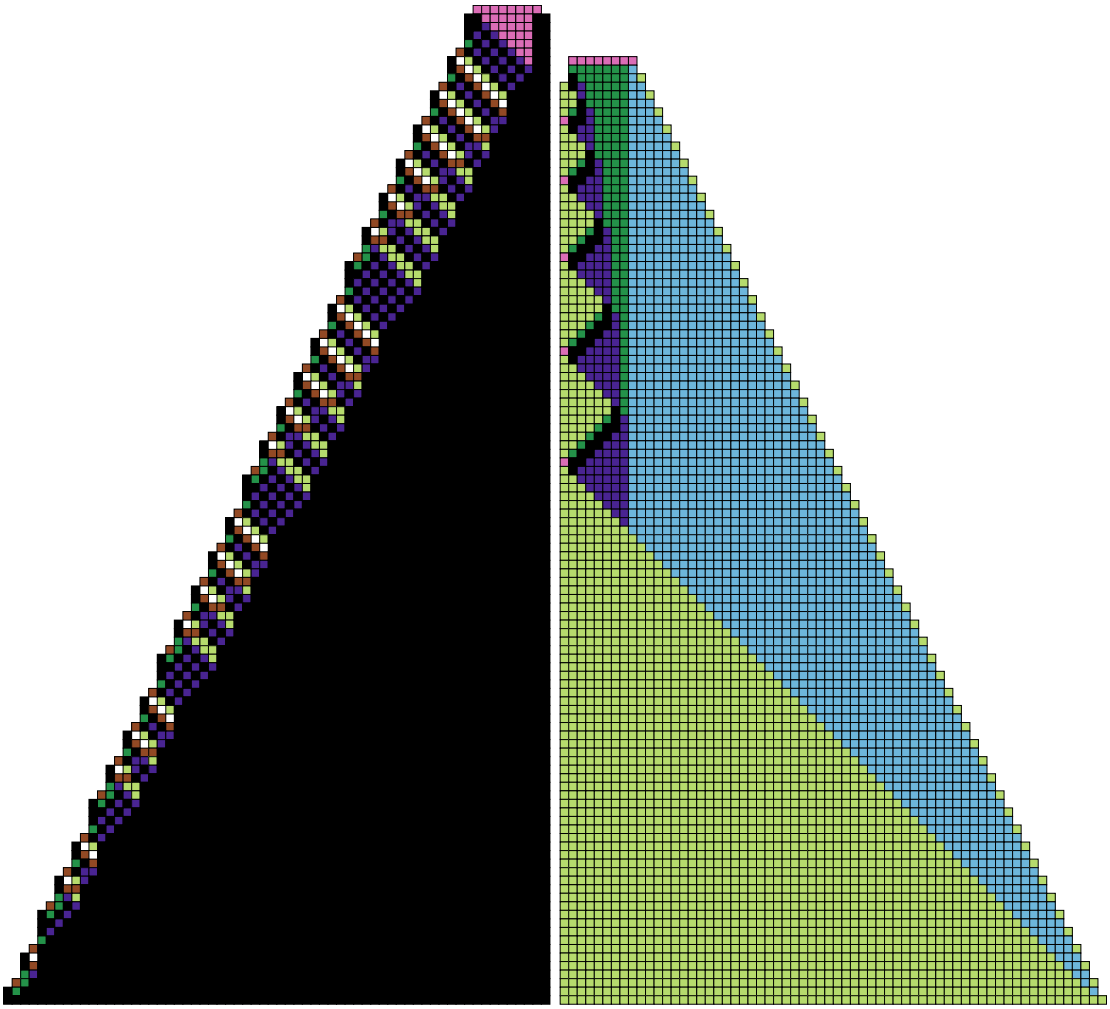
Prehľad zaujímavých riešení



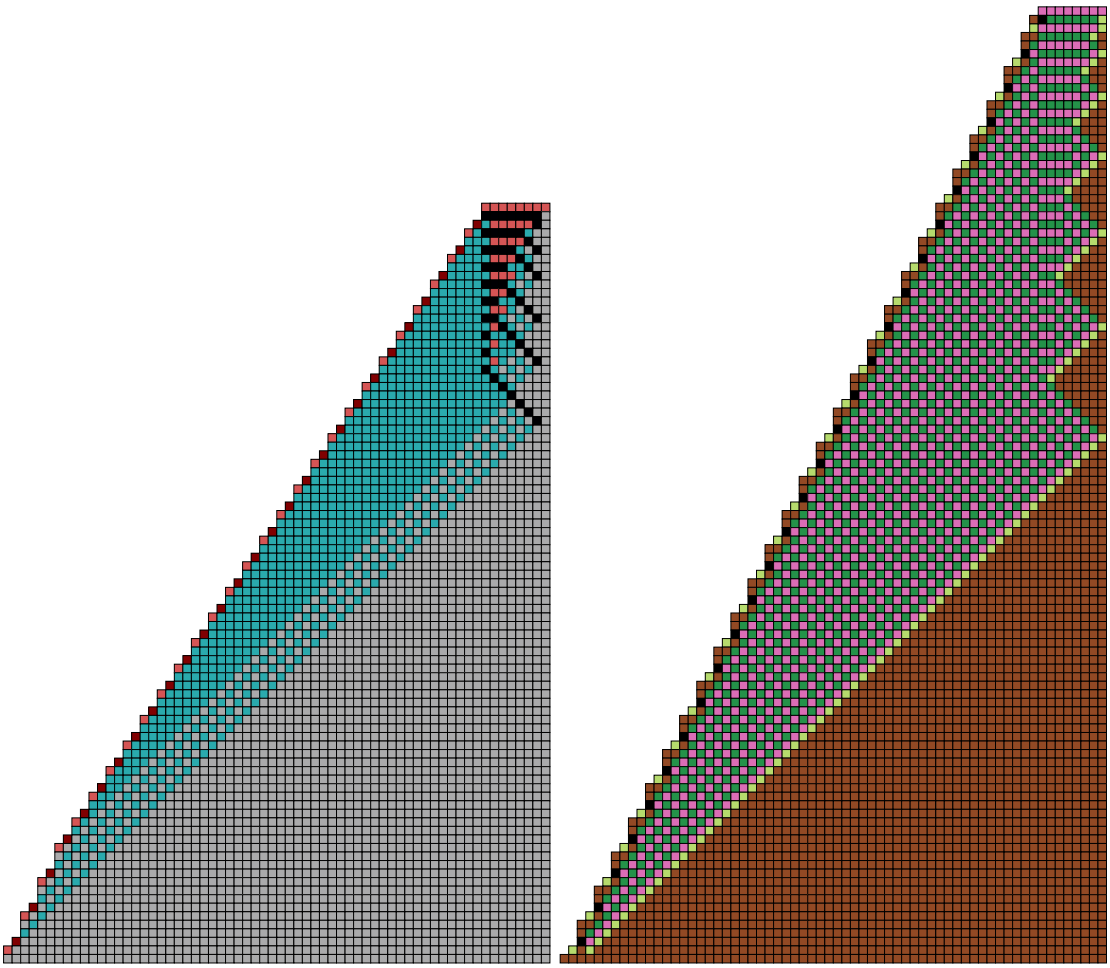
Obr. A.1: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Vývoj oboch automatov je založený na prechodovej funkcii zakódovanej pomocou tabuľkovej reprezentácie. Riešenie bolo nájdené pri mutovaní desiatich génov chromozómu so sadou vstupných hodnôt od 2 do 6 a počiatočnou konfiguráciou vľavo. Vývoj automatu vľavo nastáva pri vstupnej hodnote 6. Vývoj automatu vpravo nastáva pri vstupnej hodnote 8. Riešenie nie je všeobecné.



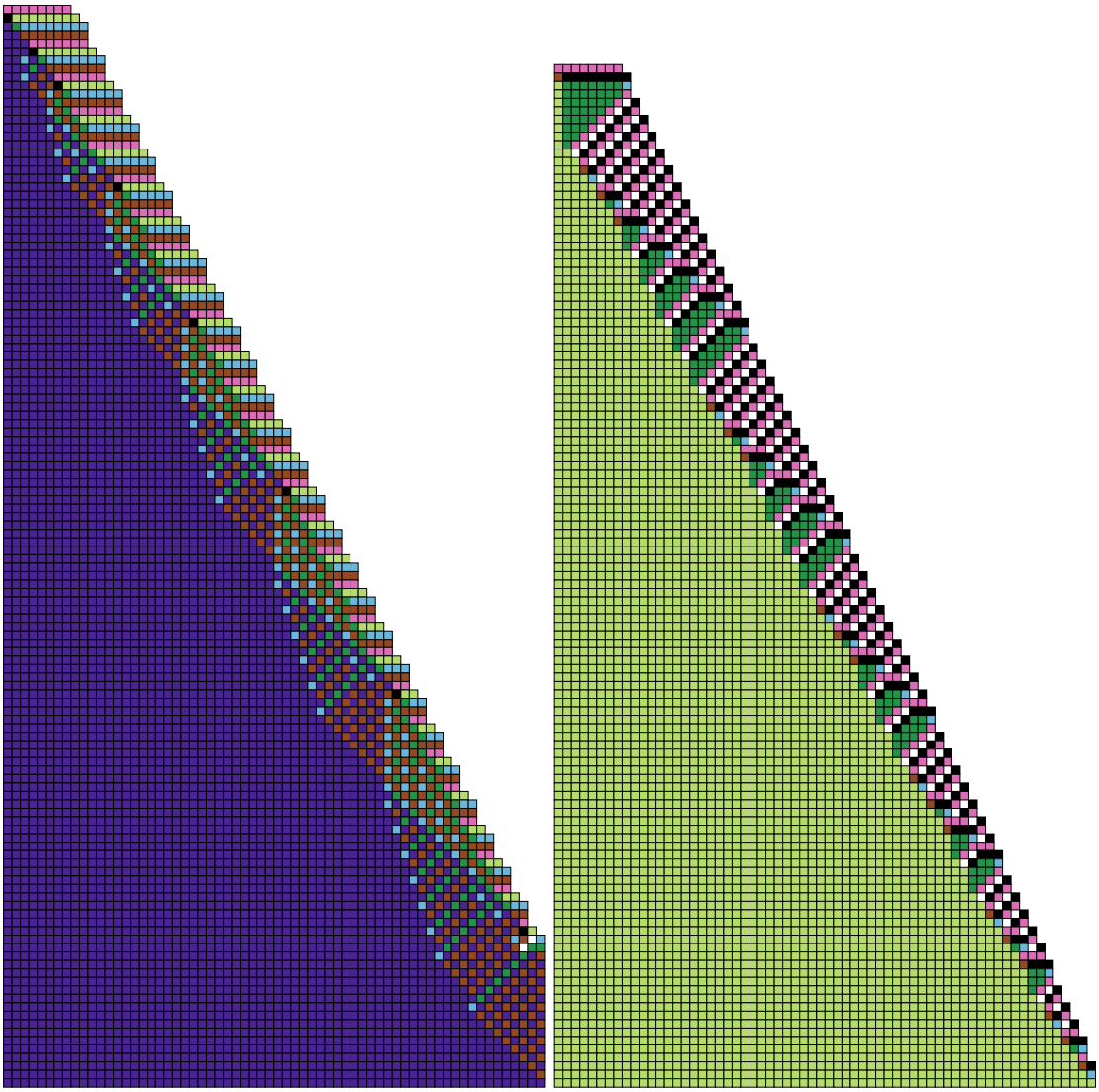
Obr. A.2: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Riešenie vľavo bolo nájdené pomocou reprezentácie CMR s dvadsiatimi podmienkovými pravidlami, množinou vstupných hodnôt 2 až 5, a počiatočnou konfiguráciou na ľavej strane automatu. Riešenie vpravo bolo nájdené s počiatočnou konfiguráciou v strede automatu.



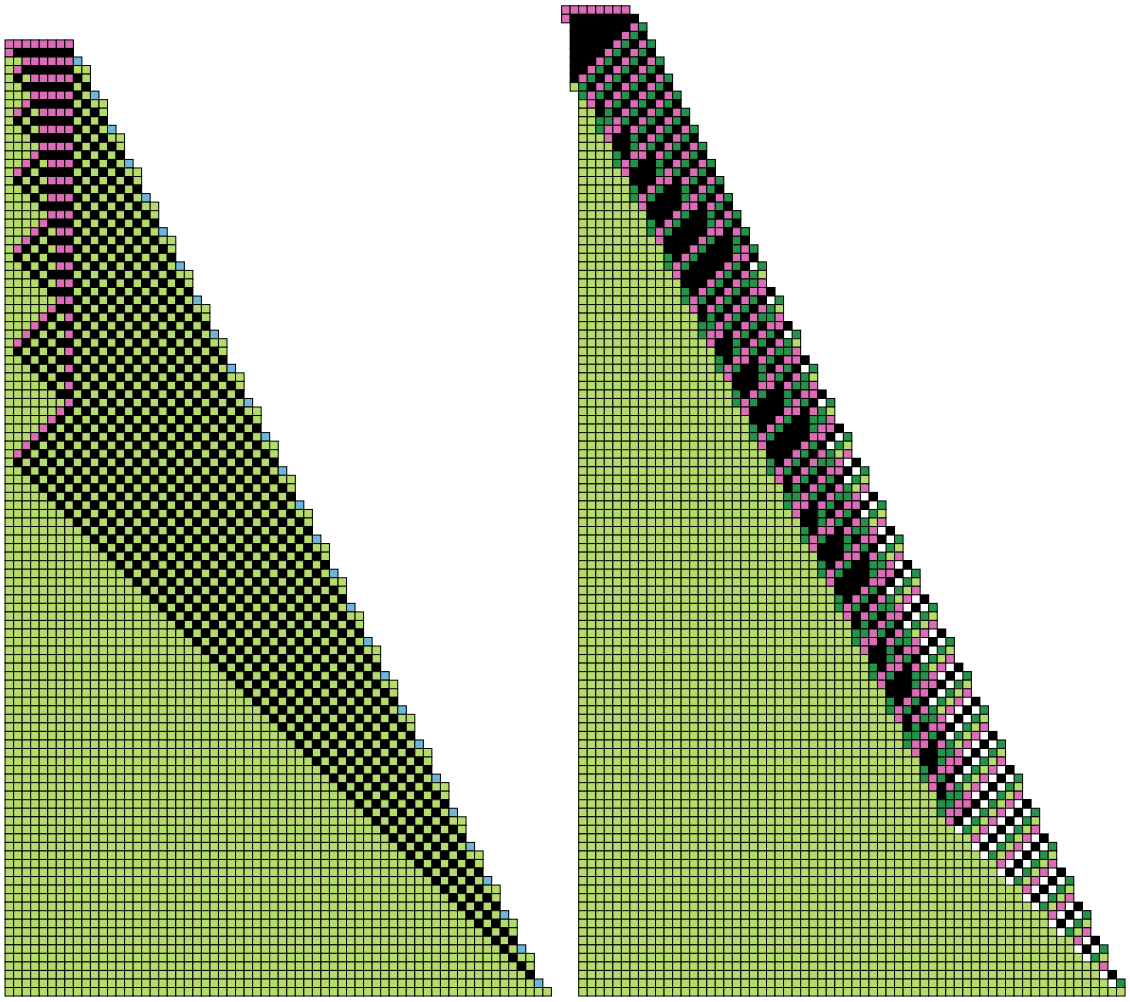
Obr. A.3: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Riešenie vľavo bolo nájdené pomocou reprezentácie CMR s dvadsiatimi podmienkovými pravidlami, množinou vstupných hodnôt 2 až 5, a počiatočnou konfiguráciou na pravej strane automatu. Riešenie vpravo bolo nájdené s počiatočnou konfiguráciou vľavo a s množinou vstupných hodnôt 2 až 6.



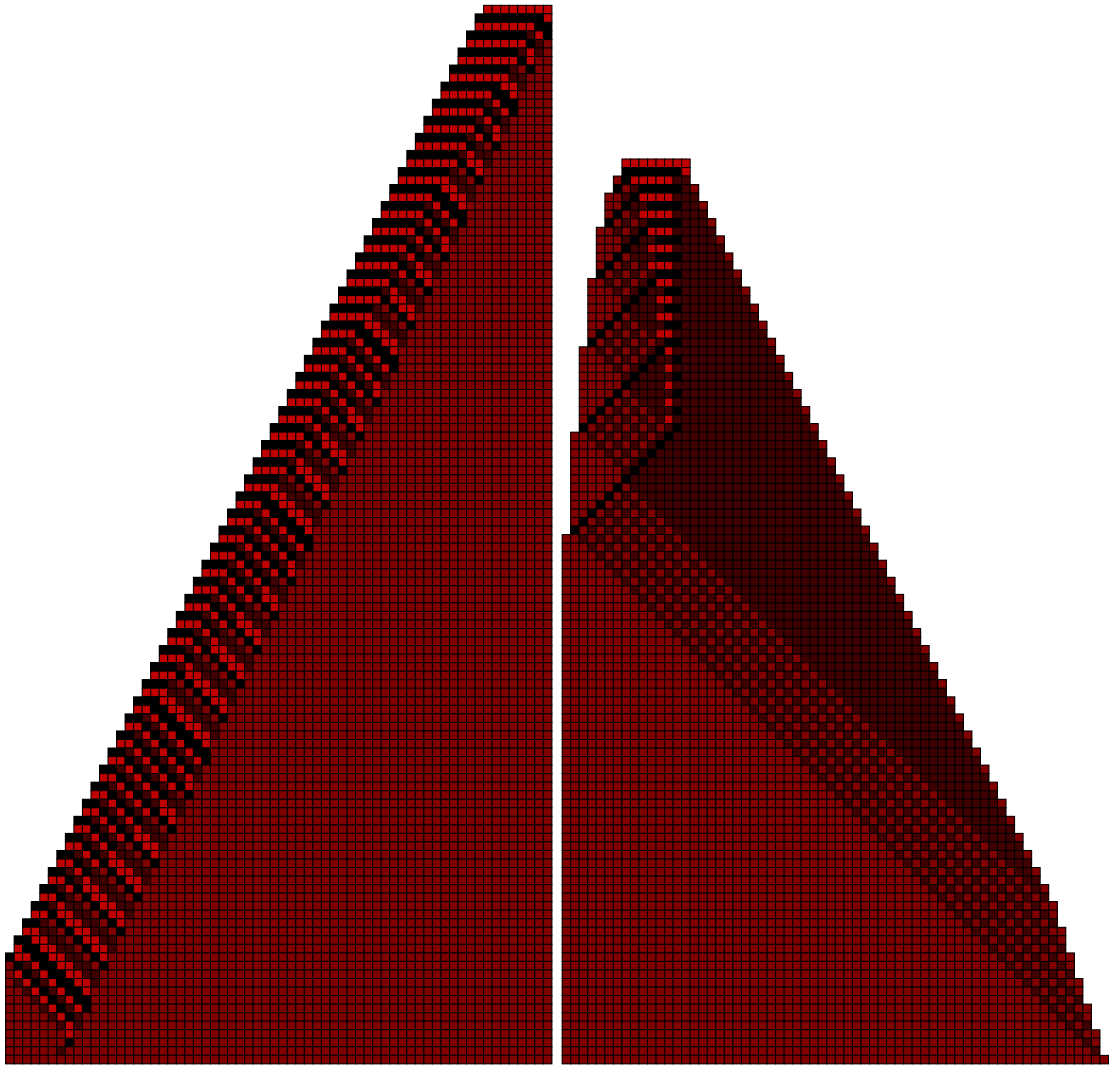
Obr. A.4: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Riešenie vľavo bolo nájdené pomocou reprezentácie CMR s dvadsiatimi podmienkovými pravidlami, množinou vstupných hodnôt 2 až 6, a počiatočnou konfiguráciou v strede automatu. Riešenie vpravo bolo nájdené s počiatočnou konfiguráciou vpravo.



Obr. A.5: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Riešenie vľavo bolo nájdené pomocou reprezentácie CMR so štrnástimi podmienkovými pravidlami, množinou vstupných hodnôt 2 až 6, a počiatočnou konfiguráciou v strede automatu. Riešenie vpravo bolo nájdené pomocou CMR so šestnástimi pravidlami a s počiatočnou konfiguráciou vľavo.



Obr. A.6: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Riešenie vľavo bolo nájdené pomocou reprezentácie CMR s ôsmimi podmienkovými pravidlami, množinou vstupných hodnôt 2 až 6, a počiatočnou konfiguráciou vľavo. Riešenie vpravo bolo nájdené s množinou vstupných hodnôt 2 až 7.



Obr. A.7: Evolučne nájdené riešenia problému druhej mocniny v jednorozmernom celulárnom automate; Riešenie vľavo bolo nájdené pomocou reprezentácie CMR s piatimi podmienkovými pravidlami, množinou vstupných hodnôt 2 až 6, a počiatočnou konfiguráciou v strede automatu. Riešenie vpravo bolo nájdené reprezentáciou CMR so šiestimi podmienkovými pravidlami, s množinou vstupných hodnôt 2 až 7 a s počiatočnou konfiguráciou automatu vľavo.

Príloha B

Prechodové funkcie zaujímavých riešení

≤ 6	≥ 2	≥ 3	4
≥ 7	$\neq 0$	≤ 0	3
$= 0$	≤ 4	≤ 4	0
≤ 3	≤ 5	≤ 0	6
≥ 6	≥ 0	$= 0$	4
≥ 5	$= 0$	$= 0$	7
≥ 6	≤ 5	$\neq 0$	7
$= 0$	≤ 3	$= 0$	3
≤ 3	$= 0$	$= 0$	5
≤ 4	$= 0$	≥ 6	5
≤ 2	$= 0$	≤ 6	0
≤ 0	≤ 4	$\neq 0$	5
≥ 1	≥ 2	≥ 1	1
$\neq 0$	$= 0$	≤ 6	0
$\neq 0$	≤ 1	$\neq 0$	7
≤ 1	$\neq 0$	$= 0$	5
≥ 6	≥ 7	≤ 6	4
$= 0$	$= 0$	$\neq 0$	6
≥ 0	≥ 1	≤ 7	4
$\neq 0$	$\neq 0$	≤ 6	1

≤ 0	≥ 2	≤ 5	2
≥ 5	$\neq 0$	≤ 5	5
$= 0$	≥ 7	$= 0$	6
≥ 2	$\neq 0$	≤ 5	2
$= 0$	$\neq 0$	$\neq 0$	3
≤ 3	$= 0$	≤ 2	0
≥ 4	$= 0$	≤ 2	1
≥ 4	$\neq 0$	$\neq 0$	1
$\neq 0$	≤ 5	≤ 0	7
≥ 1	$= 0$	$= 0$	1
≥ 6	$\neq 0$	≤ 5	7
$\neq 0$	$\neq 0$	$\neq 0$	6
$\neq 0$	$= 0$	$= 0$	0
≤ 2	$= 0$	≥ 7	1
≤ 1	≤ 6	≤ 5	2
$\neq 0$	≥ 3	$\neq 0$	4
$\neq 0$	≥ 0	≤ 2	7
$\neq 0$	≤ 6	$\neq 0$	0
$\neq 0$	$= 0$	$= 0$	6
≥ 0	$\neq 0$	$\neq 0$	7

Tabuľka B.1: Prechodové funkcie evolučne nájdených riešení problému druhej mocniny v jednorozmernom celulárnom automate pre riešenia na obrázku A.2

$= 0$	$= 0$	$= 0$	0
$= 0$	≤ 1	$= 0$	5
≤ 1	$\neq 0$	$= 0$	7
$\neq 0$	$= 0$	$\neq 0$	3
≤ 7	≥ 5	≥ 6	7
≤ 2	$= 0$	$= 0$	7
≤ 7	≥ 3	≤ 2	7
$\neq 0$	≥ 5	$= 0$	2
≥ 0	≥ 2	$= 0$	2
$= 0$	≤ 1	≤ 4	7
$= 0$	≤ 2	≤ 6	3
≥ 5	$\neq 0$	≥ 1	5
$= 0$	$\neq 0$	≤ 6	6
≥ 7	$\neq 0$	$= 0$	2
$\neq 0$	≥ 2	$\neq 0$	2
$= 0$	≤ 5	$\neq 0$	0
≥ 5	$\neq 0$	≤ 5	7
$= 0$	≥ 3	$= 0$	4
$= 0$	$= 0$	≥ 7	0
$\neq 0$	$= 0$	≥ 2	2

≥ 4	$= 0$	≤ 7	2
≤ 4	≥ 3	≥ 6	7
≥ 7	$\neq 0$	$\neq 0$	5
≤ 4	$= 0$	≤ 6	0
≤ 2	≥ 2	≤ 5	2
$= 0$	≥ 2	≤ 6	1
≥ 1	≤ 5	$= 0$	4
$\neq 0$	$= 0$	$= 0$	0
≤ 1	≤ 6	≥ 2	2
$\neq 0$	$= 0$	≥ 6	7
≤ 3	≤ 2	$\neq 0$	6
$= 0$	≥ 5	$= 0$	3
$\neq 0$	$= 0$	≤ 2	3
≤ 0	≥ 6	$= 0$	3
$\neq 0$	≤ 3	≥ 0	7
$= 0$	$= 0$	≥ 5	0
≤ 4	≤ 7	$\neq 0$	4
$= 0$	$= 0$	$\neq 0$	2
≥ 1	≤ 0	$\neq 0$	4
≤ 3	≥ 0	≥ 6	1

Tabuľka B.2: Prechodové funkcie evolučne nájdených riešení problému druhej mocniny v jednorozmernom celulárnom automate pre riešenia na obrázku A.3

≥ 5	≤ 5	≤ 0	2
≥ 2	$= 0$	$\neq 0$	3
≤ 5	$\neq 0$	≥ 3	5
$\neq 0$	$\neq 0$	≥ 5	1
≤ 5	≥ 1	≥ 2	2
≤ 3	$\neq 0$	$= 0$	2
$\neq 0$	$\neq 0$	$\neq 0$	6
$\neq 0$	$\neq 0$	≥ 0	0
$\neq 0$	≥ 6	$= 0$	2
≤ 1	$\neq 0$	≤ 6	6
$\neq 0$	≥ 4	≥ 3	4
≤ 4	≥ 0	≥ 5	3
≥ 0	$= 0$	≥ 4	5
≥ 6	$= 0$	≥ 5	5
≤ 4	$= 0$	≥ 7	2
$= 0$	$\neq 0$	≥ 0	2
≤ 0	≤ 7	≥ 3	1
≥ 4	≥ 2	$\neq 0$	1
≤ 5	$\neq 0$	≤ 6	5
≤ 3	$\neq 0$	≥ 6	3

≤ 5	≥ 6	$= 0$	4
$\neq 0$	≥ 6	≥ 1	1
$\neq 0$	$\neq 0$	≥ 7	2
$\neq 0$	$= 0$	$\neq 0$	5
≤ 3	≥ 2	≤ 5	3
$\neq 0$	$\neq 0$	$= 0$	2
$\neq 0$	≥ 5	$\neq 0$	7
$= 0$	≤ 7	≥ 7	2
$= 0$	≥ 7	≤ 1	4
$\neq 0$	$\neq 0$	$= 0$	4
≤ 6	≤ 5	$= 0$	0
$\neq 0$	≤ 6	≤ 1	6
$= 0$	≥ 2	≤ 5	6
≤ 0	$= 0$	≥ 5	7
$= 0$	$\neq 0$	$\neq 0$	7
≤ 7	$\neq 0$	≤ 2	2
$\neq 0$	≤ 4	$\neq 0$	6
≥ 0	$\neq 0$	$= 0$	0
$= 0$	$= 0$	≤ 2	3
$= 0$	≥ 7	$= 0$	6

Tabuľka B.3: Prechodové funkcie evolučne nájdených riešení problému druhej mocniny v jednorozmernom celulárnom automate pre riešenia na obrázku A.4

$= 0$	$\neq 0$	≤ 3	2
$= 0$	≤ 6	≤ 7	4
≤ 1	≥ 2	≥ 1	5
≤ 5	$= 0$	$\neq 0$	7
≤ 3	$= 0$	$\neq 0$	1
$= 0$	≥ 1	≤ 5	5
≥ 5	$\neq 0$	≥ 5	4
≥ 3	≥ 4	$\neq 0$	3
$= 0$	$= 0$	$= 0$	4
≥ 7	≥ 6	≤ 1	5
≥ 3	$\neq 0$	$= 0$	0
≥ 2	≤ 7	≤ 6	6
$\neq 0$	≥ 4	$= 0$	6
$= 0$	$\neq 0$	$= 0$	1

$= 0$	$\neq 0$	$\neq 0$	7
≥ 5	$\neq 0$	$= 0$	1
≥ 1	$= 0$	≤ 2	4
≤ 3	≤ 3	≥ 6	2
$\neq 0$	$= 0$	≥ 5	6
≥ 5	$= 0$	≥ 3	1
≥ 4	$= 0$	≥ 6	1
$= 0$	≥ 2	≤ 5	2
≤ 6	≥ 3	≤ 1	1
$\neq 0$	$= 0$	$\neq 0$	1
$\neq 0$	≥ 6	≤ 7	6
≥ 4	≤ 1	≤ 4	0
≤ 2	$= 0$	≤ 6	3
≤ 7	≥ 6	$= 0$	2
$\neq 0$	≤ 7	$= 0$	7
≥ 5	≥ 2	≤ 2	7

Tabuľka B.4: Prechodové funkcie evolučne nájdených riešení problému druhej mocniny v jednorozmernom celulárnom automate pre riešenia na obrázku A.5

$= 0$	≥ 7	$\neq 0$	1
$= 0$	$\neq 0$	$= 0$	1
≤ 5	≥ 2	≥ 0	2
$= 0$	$\neq 0$	≥ 0	2
≤ 0	$= 0$	≤ 4	4
$\neq 0$	≥ 1	≥ 0	7
≥ 3	≤ 2	≤ 7	4
≤ 5	≤ 3	≥ 3	2

$= 0$	$\neq 0$	$\neq 0$	7
≥ 3	≥ 2	≤ 3	1
$= 0$	$\neq 0$	≤ 6	6
$\neq 0$	≥ 7	≤ 0	0
$= 0$	≤ 3	≥ 2	2
$= 0$	$\neq 0$	≥ 5	6
$\neq 0$	≥ 2	≤ 3	2
≤ 4	≤ 5	$\neq 0$	0

Tabuľka B.5: Prechodové funkcie evolučne nájdených riešení problému druhej mocniny v jednorozmernom celulárnom automate pre riešenia na obrázku A.6

≤ 2	≥ 2	≤ 2	2
$\neq 0$	≤ 2	≥ 4	3
$\neq 0$	≤ 4	$= 0$	4
≥ 2	$= 0$	$\neq 0$	4
≥ 0	≥ 3	≥ 3	1

≥ 3	≥ 1	≤ 3	3
$= 0$	≤ 1	≥ 3	2
$= 0$	≤ 3	≤ 3	2
$\neq 0$	≥ 2	$\neq 0$	2
≤ 2	≥ 1	≥ 1	4
≤ 4	≥ 2	≥ 3	1

Tabuľka B.6: Prechodové funkcie evolučne nájdených riešení problému druhej mocniny v jednorozmernom celulárnom automate pre riešenia na obrázku A.7