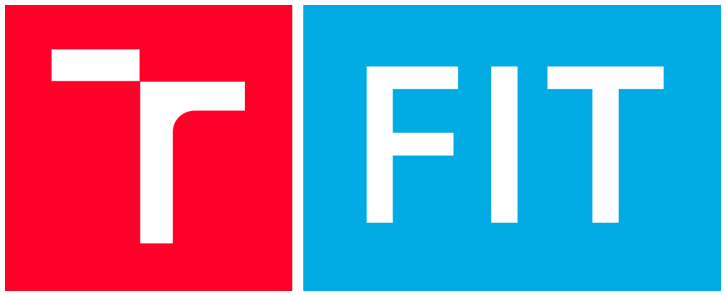
# VYSOKÉ UČENIE TECHNICKÉ V BRNE

Fakulta informačných technológií



## Aplikované a evolučné algoritmy

## 2016/2017

# Umenie v celulárnych automatoch

Filip Gulán (xgulan00) 5.5.2017

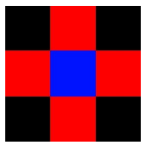
# Obsah

# 1 Úvod

Úlohou tohto projektu bolo vytvoriť evolučný algoritmus, ktorý má za úlohu nájsť prechodové funkcie celulárneho automatu, ktorých vývoj bude do istého zmyslu chápaný ako umenie. Mali byť uvažované aspoň 4 stavy na bunku a výsledky mali byť prezentované vo forme tabuliek s pravidlami alebo graficky, ako vývoj celulárneho automatu. Pre každý zaujímavý výsledok bol zdokumentovaný presný stav systému. Pre účely projektu, presnejšie pre vizualizáciu získaných pravidiel, bol použitý poskytnutý simulátor celulárnych automatov menom *BiCAS*.

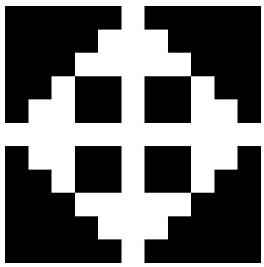
# 2 Popis použitého evolučného algoritmu

Pre účely projektu bol vytvorený genetický algoritmus, ktorý obsahuje kríženie a mutáciu pre zabezpečenie variability novovzniknutých jedincov. Na začiatku genetického algoritmu sa musí vygenerovať počiatočná populácia jedincov/chromozómov. Táto populácia je v tomto prípade implementovaná ako vektor štruktúr typu *chromozome*. Každý jedinec je teda vlastná štruktúra *chromozome*, ktorá v sebe obsahuje hodnotu *fitness* a vektor štruktúr *rule*, ktorý predstavuje pravidlá. Štruktúra *rule* v sebe obsahuje položky *sum*, *current* a *next*. *Current* obsahuje aktuálnu hodnotu bunky. *Next* obsahuje budúcu hodnotu bunky, ktorá sa stane *current* po tom, ako je hodnota okolia rovná hodnote *sum*. *Sum* predstavuje sumu hodnôt štvor-okolia. Toto štvor-okolie je znázornené na obrázku 1 červenými pixelami, aktuálna bunka je znázornená modrým pixelom.



Obrázok 1: Štvor-okolie bunky.

Po vygenerovaní počiatočnej populácie sú všetci jedinci ohodnotení funkciou *fitness*, ktorá funguje na tom princípe, že porovnáva podobnosť výsledného priestoru po simulácií celulárneho automatu na danom rozmere po *N* krokoch (kde *N* je rozmer automatu) s načítaným vzorom zo súboru. Vzor je uložený v súbore a môže byť chápaný ako akási predloha, ku ktorej chceme dospieť. Zvolený vzor je možné vidieť na obrázku 2.



Obrázok 2: Použitý vzor.

Tento vzor bol zvolený experimentálne, kedy dával najzaujímavejšie výsledky a pritom nebol časovo náročný na nájdenie maximálnej *fitness* funkcie. Podľa *fitness* funkcie je následne vygenerovaná populácia zoradená. Po vygenerovaní a ohodnotení počiatočnej populácie jedincov nasleduje tvorba nových generácií jedincov a následné operácie s nimi. Tvorba nových generácií jedincov prebieha vo *for* cykle, kde sa deje to, že sa do novej generácie zoberie ⅓ najlepších jedincov z predchádzajúcej generácie. Zostávajúce ⅔ jedincov novej generácie je vytvorených na základe kríženia 2 náhodných jedincov vybraných z už pridanej ⅓ najlepších. Po krížení je nakoniec ešte na novo-vziknutého jedinca aplikovaná mutácia. Kríženie funguje na tom princípe, že sa vygeneruje náhodné číslo, ktoré predstavuje bod kríženia (ide teda o jednobodové kríženie). Do tohoto vygenerovaného bodu sa berú pravidlá z prvého jedinca, od tohto bodu zase pravidlá z druhého jedinca. Mutácia je zase implementovaná tak, že sa prechádzajú pravidlá daného jedinca a pre každú položku *sum/current/next* je vykonaná mutácia podľa pravdepodobnosti p. Nová mutovaná hodnota môže byť buď *stará hodnota + 1*, alebo *stará hodnota - 1*. Smer je taktiež zvolený náhodne. *+1* a *-1* je zvolené kvôli tomu, aby mutácia nezapríčinila úplnu zmenu jedinca, ale iba jeho čiastočnú zmenu. Po vytvorení novej populácie je táto populácia znovu ohodnotená *fitness* a aj podľa nej zoradená. Nakoniec z poslednej generácie je vybraný najlepší jedinec a vrátený ako riešenie.

Algoritmus generuje pravidlá aditívneho celulárneho automatu, ktorý vykonáva prechod na základe sumy okolia, ako ale už bolo spomenuté. Pre simuláciu a následnú vizualizáciu pravidiel bol použitý poskytnutý nástroj *BiCAS*. Keďže *BiCAS* nevie pracovať s pravidlami aditívneho automatu, tak pravidlo aditívneho automatu musí byť prevedené pre pravidlá, ktoré akceptuje *BiCAS*. Toto zabezpečuje funkcia *aditiveToNonAditive()*. Táto funkcia robí v skratke to, že zoberie aditívne pravidlo, ktoré obsahuje iba *sum, current, next* a vygeneruje z neho všetky možné neaditívne pravidlá.

# 3 Popis spustenia algoritmu

Algoritmus je implementovaný ako konzolová aplikácia jazyku *C++* a je možné špecifikovať chovanie algoritmu pomocou daných prepínačov. Program má jediný povinný prepínač a tým je prepínač *-x*. Všetky ostatné parametre sú voliteľné a v prípade že nie sú špecifikované, tak sa použijú východzie hodnoty.

Parametre sú nasledovné:

- *-f <súbor>* špecifikuje, kde budú uložené výstupné pravidlá. K menu súboru je pridaná koncovka *.tab*. Východzia hodnota je *"experiment"*.
- *-x <súbor>* špecifikuje, kde je uložený vstupný vzor, ktorý bude použitý (odovzdaný archív obsahuje súbor *pattern*).
- *-p <číslo>* veľkosť populácie. Východzia hodnota *20*.
- *-g <číslo>* počet generácií. Východzia hodnota *500*.
- *-m <0-100>* pravdepodobnosť mutácie v percentách. Východzia hodnota *10*.
- *-s <číslo>* počet stavov bunky (najlepšia hodnota *4-13*). Východzia hodnota *10*.

- *-r <číslo>* počet aditívnych pravidiel. Východzia hodnota *200*.
- *-b* zapína mód, v ktorom sa na *stdin* zobrazuje iba konečná maximálna hodnota *fitness* funkcie.
- *-h* zobrazí nápovedu.

Preklad programu je zabezpečený pomocou *make*, a *make run* zase zabezpečí testovací beh programu a následné spustenie získaných pravidiel v simulátore *BiCAS*.

# 4 Experimenty

Hlavnou podstatou experimentov vykonaných v tejto kapitole bolo zistiť, ako parametre systému, presnejšie parameter pravdepodobnosti mutácie a veľkosti populácie, ovplyvňujú rýchlosť a kvalitu nájdenia výsledku . Experimenty prebehli tak, že sa pre každé nastavenie systému spustilo vykonávanie *20x* a z tejto hodnoty bol vytvorený priemer. Teda napríklad pre hodnotu bunky v druhom riadku druhom stĺpci *14,6* tabuľky 1 bol algoritmus spustený *20x*. Minimálna dosiahnuteľná hodnota *fitness* bola *0*, maximálna dosiahnuteľná hodnota *fitness* pre daný vzor v experimente bola *45*.

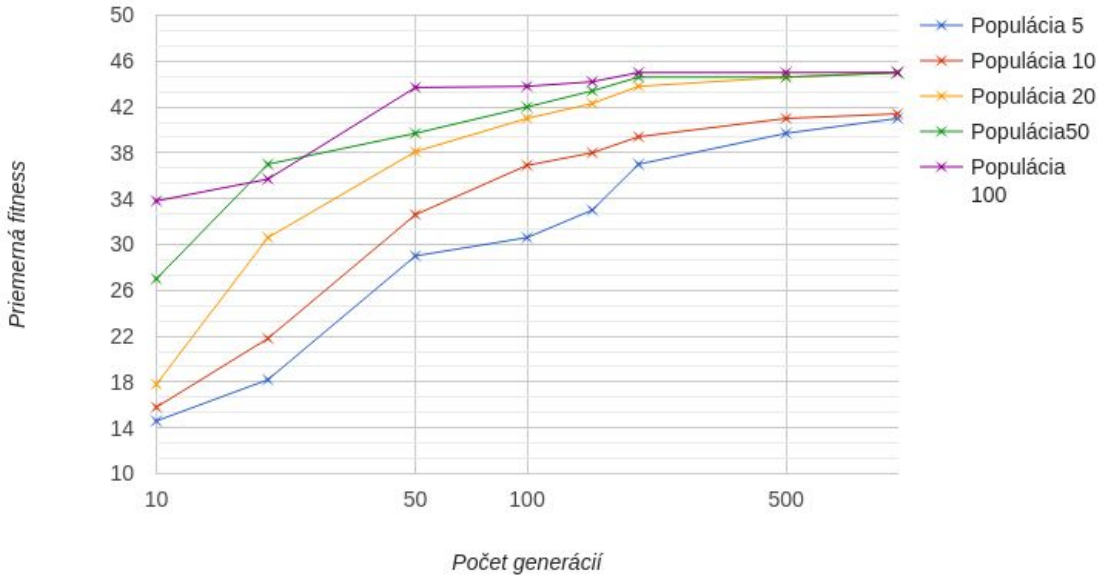
Prvý experiment bol zameraný na to, či a ako veľkosť populácie ovplyvňuje kvalitu a rýchlosť nájdeného riešenia. Systém bol spúšťaný ako *./eca -x pattern -p <P> -m 10 -g <G> -s 6 -r 50 -b, kde <P>* je veľkosť populácie a *<G>* počet generácií. Veľkosť populácie bola zvolená *5, 10, 20, 50* a nakoniec *100*. Každá táto populácia bola skúmaná v po určitom počte generácií, v prípade tohoto experimentu po *10, 20, 50, 100, 150, 200, 500* a nakoniec po *1000* generácií. Namerané hodnoty tohoto experimentu je možné vidieť v tabuľke 1 a pre názornosť boli tieto hodnoty vynesené do grafu, ktorý sa nachádza na obrázku 3.

| Počet generácií | Populácia 5 | Populácia 10 | Populácia 20 | Populácia50 | Populácia 100 |
|---|---|---|---|---|---|
| 10 | 14,6 | 15,8 | 17,8 | 27 | 33,8 |
| 20 | 18,2 | 21,8 | 30,6 | 37 | 35,7 |
| 50 | 29 | 32,6 | 38,1 | 39,7 | 43,7 |
| 100 | 30,6 | 36,9 | 41 | 42 | 43,8 |
| 150 | 33 | 38 | 42,3 | 43,4 | 44,2 |
| 200 | 37 | 39,4 | 43,8 | 44,6 | 45 |
| 500 | 39,7 | 41 | 44,6 | 44,6 | 45 |
| 1000 | 41 | 41,4 | 45 | 45 | 45 |

Tabuľka 1: Výsledky experimentu 1.

Z tabuľky 1 je možné vidieť, že najlepšie výsledky, alebo výsledky kedy sa v *20* behoch vždy dostalo maximálne možnej *fitness* hodnoty 45, boli získané s populáciou *20* v *1000* generácií. Totožne dobré výsledky boli dosahované aj pri populácií *100* a v generácií už *200*, alebo populácií *50* a generácií *500*. Pritom čas potrebný na nájdenie maximálneho riešenia pri takomto nastavení systému bol okolo *30* sekúnd. Z tabuľky 1 alebo lepšie z grafu

na obrázku 3 je ďalej možné vidieť, že populácia o jedincoch *5* a *10* maximálnu *fitness* do *1000* generácií nachádzala, ale nedialo sa tak vždy ako pri ostatných populáciách.



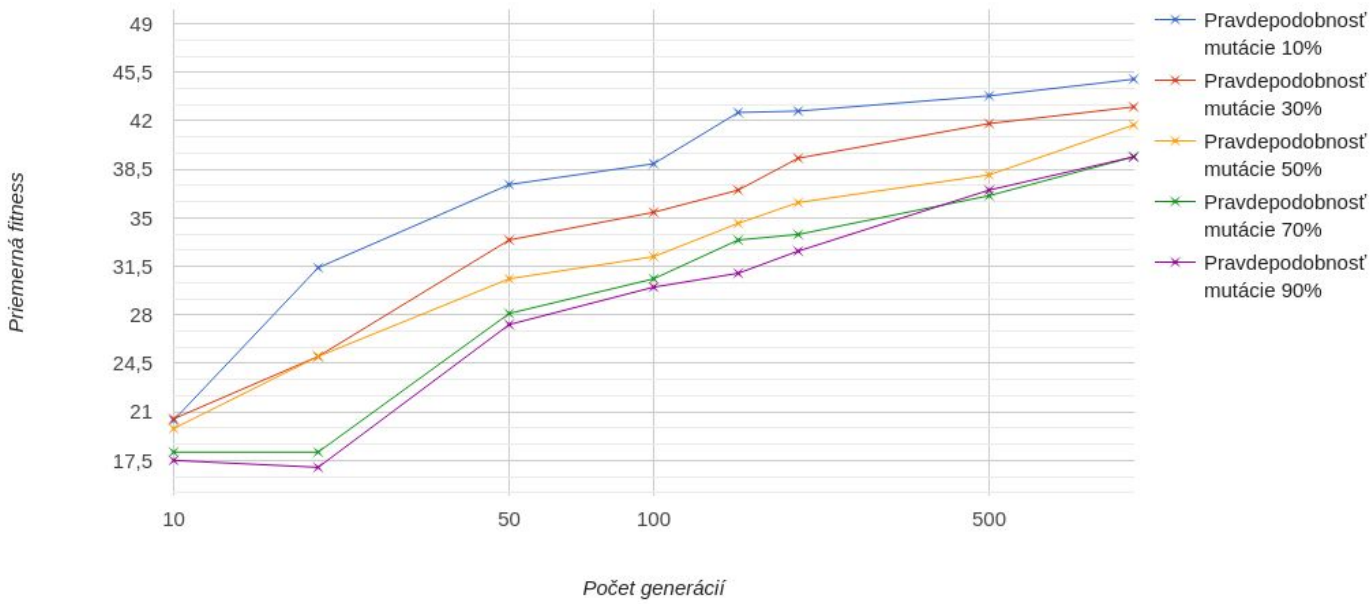

Obrázok 3: Výsledky experimentu 1 zanesené do grafu s logaritmocikou mierkou.

V poradí druhý experiment bol zameraný na to, ako a či ovplyvňuje mutácia rýchlosť a kvalitu riešenia. Systém bol tentokrát spúšťaný ako *./eca -x pattern -p 20 -m <M> -g <G> -s 6 -r 50 -b,* kde *<G>* je opäť počet generácií a *<M>* pravdepodobnosť mutácie. Experiment 2 bol vykonávaný podobne ako experiment 1. Pravdepodobnosť mutácie bola zvolená *10%, 30%, 50%, 70%* a nakoniec *90%*. Každá táto populácia bola skúmaná v po určitom počte generácií, v prípade tohoto experimentu po *10, 20, 50, 100, 150, 200, 500* a nakoniec po *1000* generácií. Namerané hodnoty tohoto experimentu je možné vidieť v tabuľke 2 a pre názornosť boli tieto hodnoty vynesené do grafu, ktorý sa nachádza na obrázku 4.

| **Počet generácií** | **Pravdepodobnosť mutácie 10%** | **Pravdepodobnosť mutácie 30%** | **Pravdepodobnosť mutácie 50%** | **Pravdepodobnosť mutácie 70%** | **Pravdepodobnosť mutácie 90%** |
|---|---|---|---|---|---|
| 10 | 20,4 | 20,5 | 19,8 | 18,1 | 17,5 |
| 20 | 31,4 | 25 | 25 | 18,1 | 17 |
| 50 | 37,4 | 33,4 | 30,6 | 28,1 | 27,3 |
| 100 | 38,9 | 35,4 | 32,2 | 30,6 | 30 |
| 150 | 42,6 | 37 | 34,6 | 33,4 | 31 |
| 200 | 42,7 | 39,3 | 36,1 | 33,8 | 32,6 |
| 500 | 43,8 | 41,8 | 38,1 | 36,6 | 37 |
| 1000 | 45 | 43 | 41,7 | 39,4 | 39,4 |

Tabuľka 2: Výsledky experimentu 2.

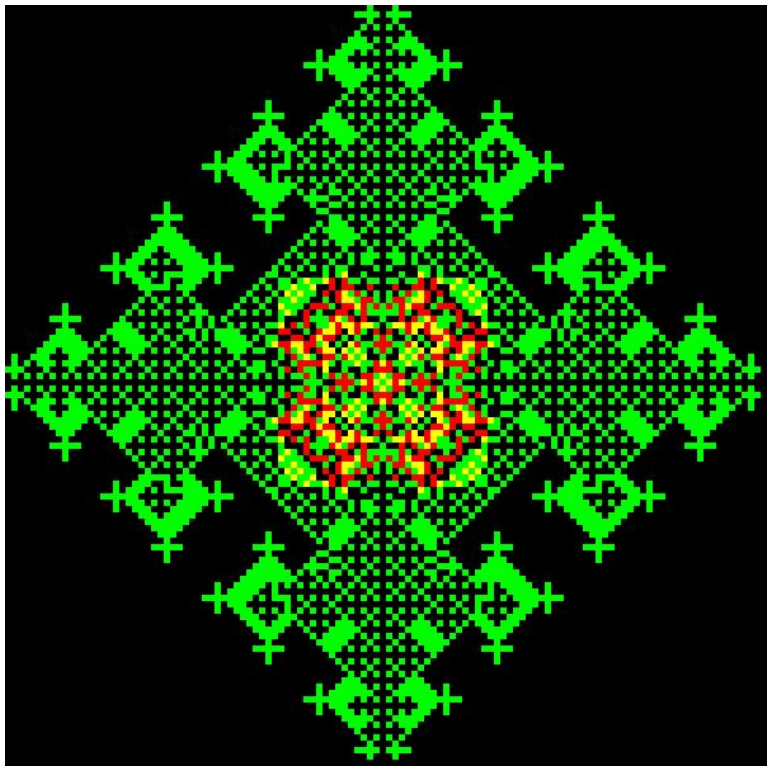
Z tabuľky 2 je možné vidieť, že jediný prípad kedy sa dosahovalo vždy v *20* behoch maximálnej *fitness* hodnoty, bolo iba keď bola zvolená mutácia najmenšia, teda *10* percent a to až v *1000* generácii. Taktiež z grafu na obrázku 4 je možné vidieť, že čím bola pravdepodobnosť mutácie väčšia, tým sa nájdenie lepšieho riešenia spomaľovalo. Tento jav môže byť vysvetlený tým, že pravdepodobnosti mutácie boli príliš veľké a jedinec, ktorý vznikol krížením 2 jedincov dobrej *fitness* bol mutáciou nakoniec tak razantne zmenený, že jeho *fitness* bola takmer vždy zhoršená oproti jeho nemutovanej alternatíve. Slovom razantne je v tomto prípade myslené to, že jedinec obsahuje v sebe pravidlá a každé pravidlo podlieha mutácii. Avšak ak sa zmenilo každé pravidlo v danom jedincovi, je prirodzené, že aj jeho *fitness* bola úplne iná.



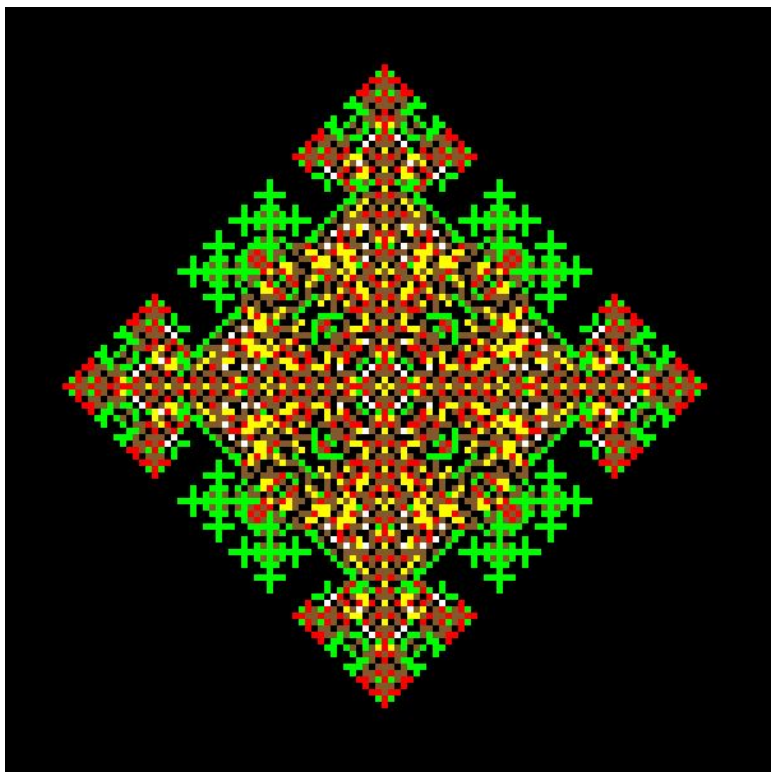

Obrázok 4: Výsledky experimentu 2 zanesené do grafu s logaritmickou mierkou.

# 5 Výsledky

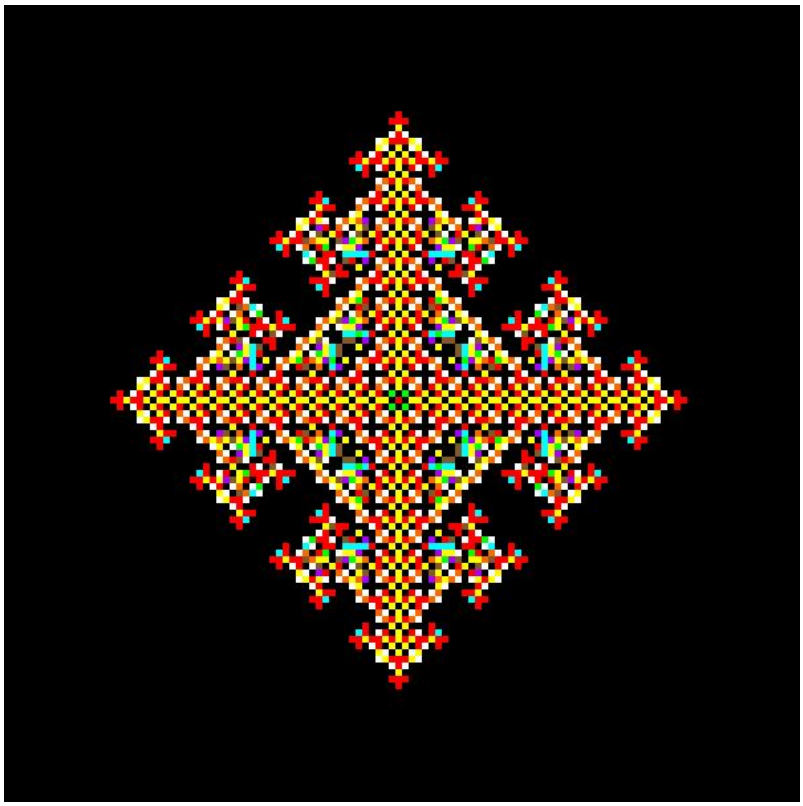
Táto kapitola obsahuje vybrané diela, ktoré boli vytvorené v simulátore *BiCAS* pomocou evolučným algoritmom získaných pravidiel. Každé dielo obsahuje popis nastavenia systému, vrátane *seedu* pseudonáhodných čísel, pre potenciálne zopakovanie získaného výsledku.



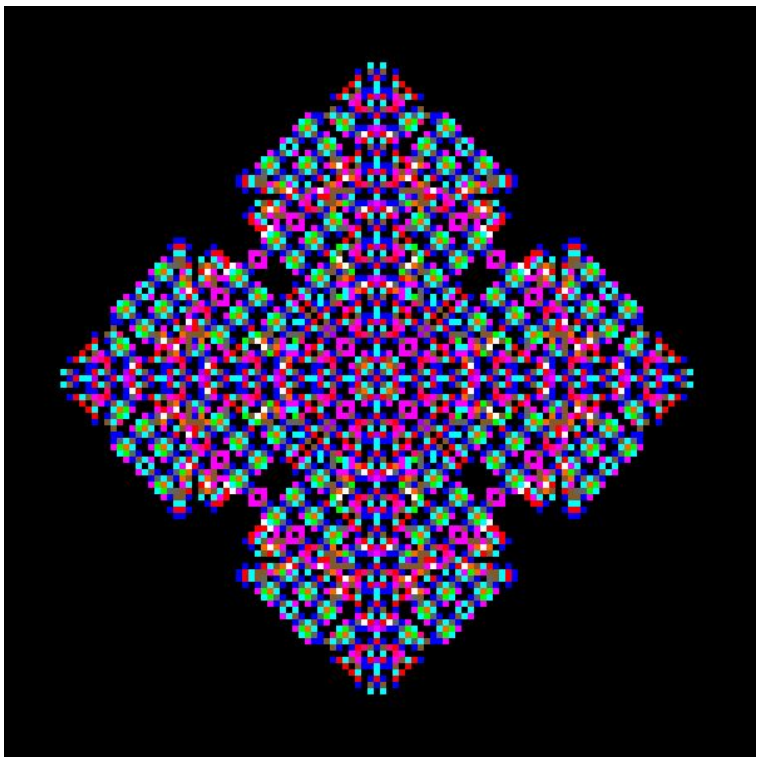
Obrázok 5: populácia 20 generácií 500 mutácia 10 stavy 4  pravidlá 30 seed 1494088068



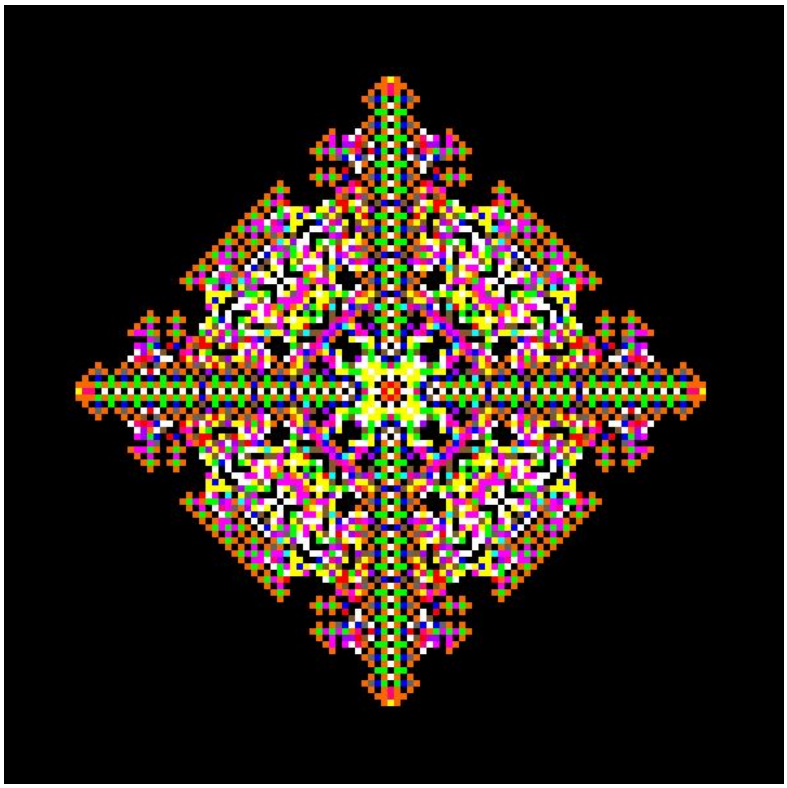
Obrázok 6: populácia 20 generácií 500 mutácia 10 stavy 6  pravidlá 50 seed 1494087900



Obrázok 7: populácia 20 generácií 500 mutácia 10 stavy 9 pravidlá 100 seed 1494088465



Obrázok 8: populácia 20 generácií 500  mutácia 10 stavy 13 pravidlá 200 seed 1494087936



Obrázok 9: populácia 20 generácií 500  mutácia 10 stavy 13 pravidlá 200 seed 1494088118

# 6 Záver

Algoritmus, ktorý bol implementovaný v rámci tohoto projektu je schopný nájsť a generovať pravidlá celulárneho automatu, ktoré vedú k estetickému výsledku, ktorý môže byť do určitej miery považovaný za umenie.

Program bol riadne otestovaný na operačnom systéme Ubuntu 16.04 LTS a na operačnom systéme CentOS. Behom testovania sa nevyskytli žiadne chyby a všetky testy dopadli úspešne.