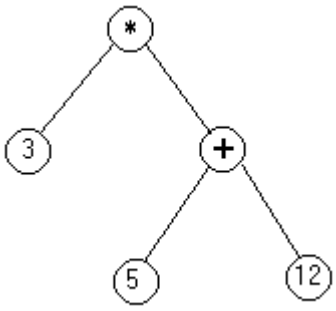
Alexandra Takáčová, FMFI UK, 4iui
Aleksandra.Takac@st.fmph.uniba.sk

Case Study z Evolučných Algoritmov

Téma 14. Genetické programovanie pre mravca hľadajúceho cestu

Úvod

Americký informatik John Koza v roku 1992 navrhol modifikáciu genetického algoritmu, ktorú nazval genetické programovanie. Na rozdiel od genetických algoritmov, ktoré pracujú s chromozómami – znakovými reťazcami, genetické programovanie nahradzuje reťazce funkciami. Funkcia je reprezentovaná stromom, kde vrcholy sú napríklad aritmetické operácie a listy stromu sú nejaké číselné hodnoty. Ale strom môže taktiež reprezentovať aj program, pričom vnútorné vrcholy stromu budú funkcie programu, a listy budú argumenty funkcií. Programovací jazyk LISP je preto vhodný pre genetické programovanie, lebo je jeho štruktúra vhodná na reprezentáciu stromom.

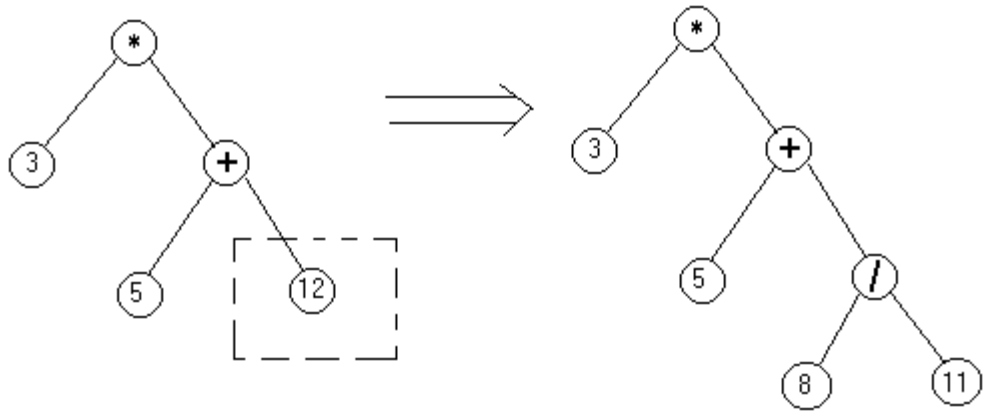


Obrázok 1. Strom, ktorý reprezentuje funkciu 3 * (5 + 12) alebo v LISPe zapísané: (* 3 (+ 5 12))

Idea, ktorá viedla k návrhu genetického programovania, bola automatické generovanie programov. Implementovaná bola v LISPe. Populácia náhodne vygenerovaných programov by geneticky evolvovala na základe princípov Darvinovskej prírodnej selekcie a biologicky motivovaných operácií. Tie operácie sú reprodukcia, kríženie a mutácia. Genetické programovanie pri každej generácii vykoná evaluáciu fitnes funkcie, Darvinovskú selekciu a genetické operácie. Každý program v populácii je ohodnotený, aby sa mohlo určiť, koľko je ten program úspešný v riešení problému. Programy sú s pravdepodobnosťou, ktorá sa určí na základe ich hodnoty fitnes funkcie, volené z populácie, pričom je povolená aj “znovu-selekcia” programu. Úspešnejší program má lepšiu šancu zúčastniť sa v rôznych genetických operáciách, ale aj neúspešný program má tiež teoretickú šancu byť zvolený.

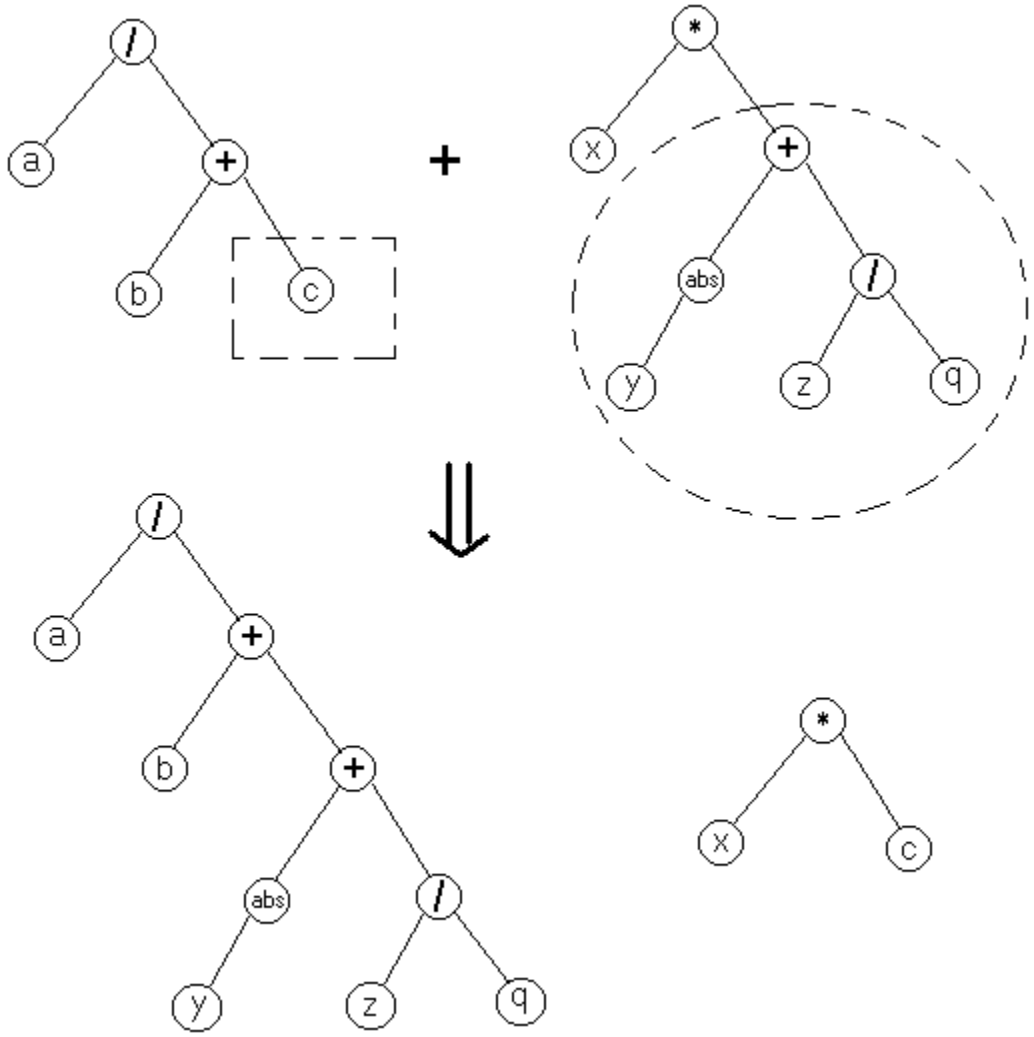
Jedince, náhodne vygenerované v prvej generácii a potomkovia, vytvorení každou genetickou operáciou, sú syntakticky korektné a vykonateľné programy. Po mnohých generáciách môže byť vygenerovaný program, ktorý rieši alebo približne rieši daný problém.

Pri operácii mutácie, na základe fitnes, sa s určitou pravdepodobnosťou zvolí z populácie rodičovský program. Vyberie sa náhodný bod stromu zvoleného programu, podstrom v tom bode sa zmaže a vytvorí sa na tom mieste náhodný strom (podobný ako pri inicializácii). Táto asexuálna operácia sa zvyčajne vykonáva s malou pravdepodobnosťou, napríklad 1%.



Obrázok 2. Operácia mutácie

V operácii kríženia, sa zvolia dva rodičovské programy, ktoré s danou pravdepodobnosťou získame z populácie na základe fitnes hodnoty. Zvyčajne sú dvaja rodičia rozličných veľkostí a tvaru. Náhodne sa zvolí bod kríženia v prvom rodiči a náhodne v druhom. Potom sa podstromy v daných vrcholoch vymenia navzájom. Kríženie je dôležitá operácia a preto sa vykonáva s dosť vysokou pravdepodobnosťou, napríklad 85% až 90%.



Obrázok 3. Operácia kríženia

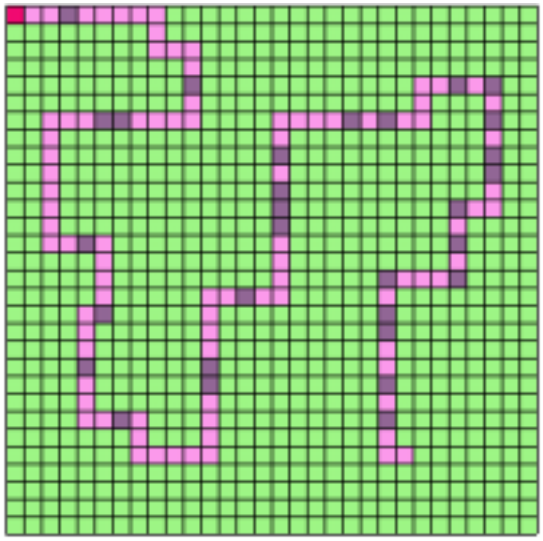
Operácia reprodukcie kopíruje jedinca s istou pravdepodobnosťou určenou na základe hodnoty fitnes funkcie do nasledovnej generácie.

Problém

Cieľom tejto práce je vyskúšať genetické programovanie na probléme hľadania pravidiel pre mravca, ktorý sa pohybuje po toroidálnej mriežke. Cieľom mravca je zozbierať čo najviac potravy za daný čas, pričom potrava je rozosiata po nie celkom súvislej dráhe a mravec vidí len políčko pred sebou. Úlohou je vyskúšať generovanie pravidiel pre jednu dráhu a pre viac dráh súčasne.

Implementácia

Prostredie, v ktorom sa mravec pohybuje, je v mojej implementácii toroidálna mriežka veľkosti 30x30. Zvolila som si počet kúskov potravy 89 a počet medzier (miesta na dráhe, kde nie je potrava) 30, hoci program vytvorený k tejto case study umožňuje vytvoriť si dráhu s ľubovoľným počtom potravy a medzier. Mravec nerozlišuje políčka medzery od políčok mimo dráhy. Mravec sa môže pohybovať dopredu, otočiť sa vpravo a otočiť sa vľavo. Ak narazí na políčko s potravou, tak ju automaticky zozbiera. Môže sa pozrieť na pole pred ním a zistiť, či tam je alebo nie je potrava a na základe toho sa rozhodnúť.



Obrázok 4. Príklad dráhy (Dráha1)

Jedna z náhodných dráh je znázornená na Obrázku 4. Červeným políčkom je reprezentovaný mravec. Na začiatku hľadania je vždy na pozícii (1,1) otočený doprava. Mravec je určený pozíciou v mriežke (x, y) a smerom, ktorým je otočený (vpravo, dolu, vľavo, hore). Ružová farba označuje potravu a fialová medzery na dráhe.

Pravidlo, t.j. program každého mravca, je strom, ktorého vnútorné vrcholy sú funkcie a listy sú terminály. Koreň stromu musí byť funkcia.

Funkcie = (IFSENSOR, PROGN)
Terminály = (TURNLEFT, TURNRIGHT, MOVE)

IFSENSOR je funkcia s dvoma argumentami. Ak políčko pred mravcom obsahuje potravu, tak sa vykoná prvý argument, inak sa vykoná druhý. Argument môže byť aj funkcia aj terminál. PROGN

je funkcia, ktorá má tri argumenty a ktoré sa vykonajú v poradí prvý, druhý, tretí bez akýchkoľvek podmienok.

Význam terminálov je zrejmý, TURNLEFT – mravec sa otočí doľava, TURNRIGHT – otočí sa doprava, MOVE – mravec urobí jeden krok dopredu.

Počet krokov mravca si môžeme zvoliť ako hodnotu ‘Simulation time’, ako krok sa počíta vykonanie každého z terminálov t.j. MOVE, TURNLEFT, TURNRIGHT.

V implementácii je tiež program reprezentovaný stromom, objektom typu TTree.

```
TTree = class
              node:TTerUFun;
              first,second,third:TTree;
          end;
```

node – označuje typ vrcholu;
first, second, third – sú podstromy;

Hlavná procedúra programu na vygenerovanie pravidiel pre jednu dráhu je procedúra genetického algoritmu:

```
procedure TEvolution.GeneticAlg (var BestAnt:integer);
var t,candidate1,candidate2,d1,d2:integer;
    i:integer;
    stop_criterion:boolean;
    new1,new2:TTree;
    pom:TPopulation;
    r:real;
begin
  t:=0;
  stop_criterion:=false;
  GenerateFirstPopulation(Population);
  while (t<number_of_epochs) and (not stop_criterion) do
    begin
      t:=t+1;
      InitializePopulation(Q);
      EvaluateFitness(Population);
      EvaluateRelativeFitness(Population);
      FindBestAnt(t, Population);
      while (Q.size<Population.size) do
        begin
          new1:=nil;
          new2:=nil;
          candidate1:=RouletteWheel(Population);
          candidate2:=RouletteWheel(Population);
          r:=random;
          if (r < probability_of_reproduction) then
             Reproduction( Population, candidate1, candidate2, new1,
                           new2, d1, d2)
          else
            begin
              Population.ants[candidate1].CopyTree(new1);
              Population.ants[candidate2].CopyTree(new2);
              d1:=Population.antDepth[candidate1];
              d2:=Population.antDepth[candidate2];
            end;
          Q.AddAnt(new1,d1);
          Q.AddAnt(new2,d2);
        end;
```

```
      pom:=Population;               //  |
      Population:=Q;                 //  |   Pop <--- Q
      DestroyPopulation(pom); //  |
      if CriteriaSatisfied(t,Population) then stop_criterion:=true;
    end;
  inc(t);
  EvaluateFitness(Population);
  FindBestAnt(t,Population);
  BestAnt:=t;
end;
```

Genetický algoritmus pre generovanie pravidiel mravca pre viac dráh súčasne je podobný predchádzajúcemu algoritmu s tým, že každého mravca simuluje na všetkých zvolených dráhach a potom na základe zozbieranej potravy vypočíta fitnes.

Ako fitnes funkciu som zvolila najskôr jednoduchú – počet kúskov zozbieranej potravy (Fitness1:= food). Vylepšená funkcia by zohľadňovala aj počet krokov mimo dráhy, čo by nám pomohlo v nájdení riešenia – mravca, ktorý sa nemýlil a sledoval správne dráhu. Použitá funkcia je nasledovná:

$$Fitness2 := 2 * food + coef / (miss + 1)$$

food – počet kúskov potravy
miss – políčka mimo dráhy, ktoré mravec navštívil
coef – vhodný koeficient (v programe je to hodnota 40), ktorý nám umožní, aby počet potravy bol dôležitejší ako zlé políčka (inak by sa mravcovi „oplatilo" byť stále na jednom mieste), ale taktiež aby funkcia „nútila" mravca sledovať dráhu

Používanie programu

- Najskôr si treba zvoliť dráhu, na ktorej chceme učiť mravca. Existujú 2 spôsoby zvolenia. Buď si načítame už hotovú mapu zo súboru (**Load Trail** tlačidlo) alebo si vytvoríme novú (**Create Trail** tlačidlo)
- Na vytvorenie novej mapy si naklikáme políčka do prázdnej mapy, na ukončenie mapy sa musí použiť tlačidlo **Finish Creating Trail**, ktorý uloží mapu s aktuálnym číslom
- **Start Evolution** tlačidlo spustí generovanie pravidiel genetickým programovaním pre aktuálnu mapu, pričom sa po každých 10 krokov vykreslí cesta najlepšieho mravca
- **Save Best Ants** uloží do súboru najlepších mravcov vo všetkých epochách
- Ak chceme generovať pravidlá pre viac dráh súčasne, tak si najprv pridáme do zoznamu dráhy pomocou tlačítka **Add Trail to List**, potom spustíme genetický algoritmus na zozname použitím **Evolve on Trail List** tlačítka
- Tlačidlo **Rem** vymaže všetky dráhy zo zoznamu, **Simulate Best Ant** simuluje najlepšieho mravca z poslednej evolúcie na vybranej dráhe, **Exit** použijeme na ukončenie programu
- Hodnoty ako pravdepodobnosť mutácie, kríženia, veľkosť populácie atď. upravujeme v TrackBaroch, maximálnu dĺžku simulácie pohybu mravca na mape a maximálnu povolenú hĺbku stromu reprezentujúceho pravidla pohybu mravca po dráhe je možné zmeniť v príslušnom Edit políčku

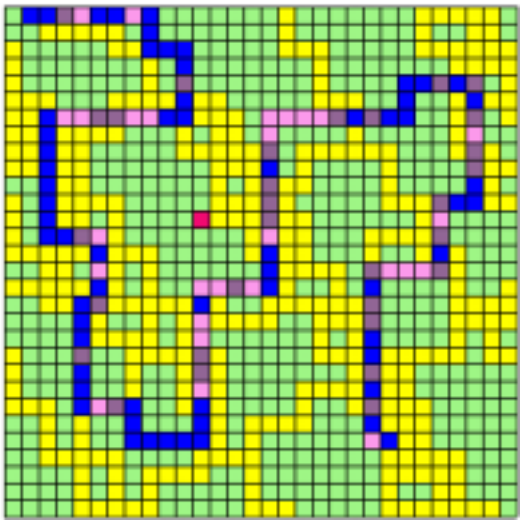
Výsledky

Hodnoty, ktoré sa mi ukázali ako najvhodnejšie, necháme fixované:

- Čas simulácie pravidiel jedného mravca na vypočítanie fitnes: 700
- Maximálna hĺbka stromu: 14
- Pravdepodobnosť reprodukcie: 1
- Pravdepodobnosť kríženia: 0.6
- Pravdepodobnosť mutácie: 0.2
- Počet epoch: 100
- Veľkosť populácie: 380

Je prípustné, že inak zvolené hodnoty tiež dávajú dobré výsledky. Uvedené parametre som zvolila na základe skúšania ako aj po oboznámení sa s výsledkami iných štúdií a literatúry. Veľkosť populácie je určená zohľadňovaním časovej zložitosti výpočtu, ale prípustné je, že menšia populácia nám tiež môže dať pekné výsledky.

- Najskôr sú ukázané výsledky s Fitness1:

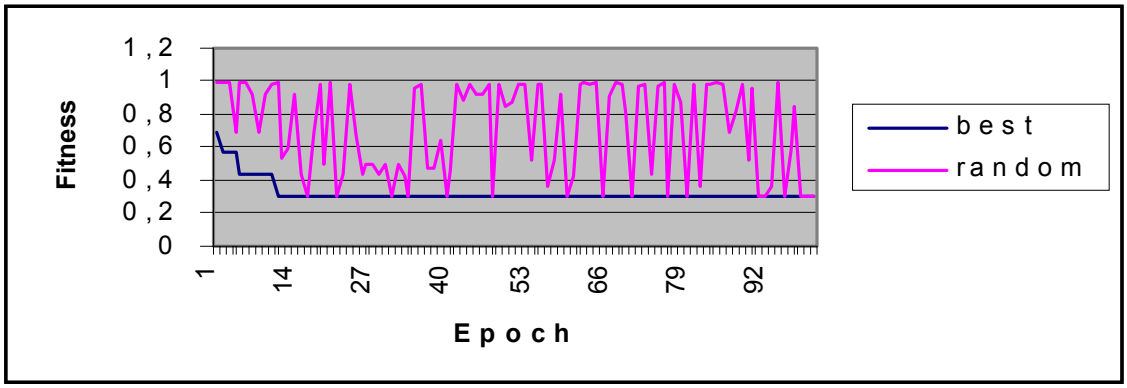


Obrázok 5.

Mapa ukazuje pohyb najlepšieho mravca, modré políčka označujú miesta, kde mravec zozbieral potravu, žlté miesta označujú pohyb mravca mimo dráhy a červené políčko je posledná pozícia mravca.
Program tohto mravca je nasledovný:

```
( PROGN  ( PROGN  ( IFSENSOR  ( PROGN  ( PROGN  ( IFSENSOR MOVE  (
PROGN TURNLEFT  ( PROGN  ( PROGN MOVE  ( IFSENSOR TURNLEFT TURNRIGHT
)TURNRIGHT ) ( PROGN MOVE TURNRIGHT  ( IFSENSOR MOVE MOVE ))TURNLEFT
)MOVE ))MOVE TURNRIGHT ) ( IFSENSOR  ( PROGN MOVE TURNLEFT  ( IFSENSOR
TURNRIGHT TURNRIGHT ))MOVE ) ( PROGN TURNLEFT  ( IFSENSOR MOVE
TURNRIGHT )TURNLEFT ))TURNRIGHT ) ( IFSENSOR MOVE MOVE )MOVE )MOVE  (
IFSENSOR  ( IFSENSOR  ( IFSENSOR TURNLEFT TURNLEFT ) ( IFSENSOR
TURNRIGHT  ( PROGN TURNRIGHT MOVE  ( IFSENSOR MOVE MOVE )))) ( IFSENSOR
( PROGN  ( IFSENSOR MOVE MOVE )TURNLEFT  ( PROGN MOVE TURNRIGHT  (
IFSENSOR  ( PROGN  ( PROGN  ( PROGN TURNRIGHT  ( IFSENSOR  ( IFSENSOR
MOVE  ( IFSENSOR TURNRIGHT  ( PROGN TURNRIGHT  ( IFSENSOR MOVE MOVE
)TURNLEFT )))TURNLEFT ) ( IFSENSOR TURNLEFT  ( PROGN TURNLEFT TURNLEFT
TURNRIGHT )))TURNLEFT  ( IFSENSOR MOVE MOVE ))TURNRIGHT TURNRIGHT ) (
IFSENSOR  ( IFSENSOR TURNRIGHT  ( PROGN MOVE  ( IFSENSOR  ( IFSENSOR  (
IFSENSOR MOVE  ( PROGN  ( PROGN MOVE TURNRIGHT TURNRIGHT )MOVE MOVE ))
( IFSENSOR MOVE TURNRIGHT ))TURNLEFT )TURNLEFT )) ( PROGN MOVE  ( PROGN
MOVE  ( PROGN  ( PROGN  ( PROGN  ( IFSENSOR TURNRIGHT TURNLEFT
)TURNRIGHT  ( PROGN  ( PROGN TURNRIGHT TURNLEFT MOVE )TURNRIGHT  (
PROGN TURNLEFT TURNRIGHT TURNLEFT )))TURNRIGHT MOVE )MOVE TURNRIGHT ) (
IFSENSOR TURNLEFT MOVE ))TURNLEFT ))))) ( PROGN TURNLEFT MOVE MOVE ))))
```

Ukazuje sa, že táto voľba fitnes funkcie nie je vhodná, lebo nenúti mravca sledovať dráhu potravy. Mravec sa náhodne pohybuje po mape. Na grafe môžeme vidieť, ako klesá hodnota najlepšieho mravca, druhý mravec (random) je náhodný, t.j. prvý mravec v populácii.

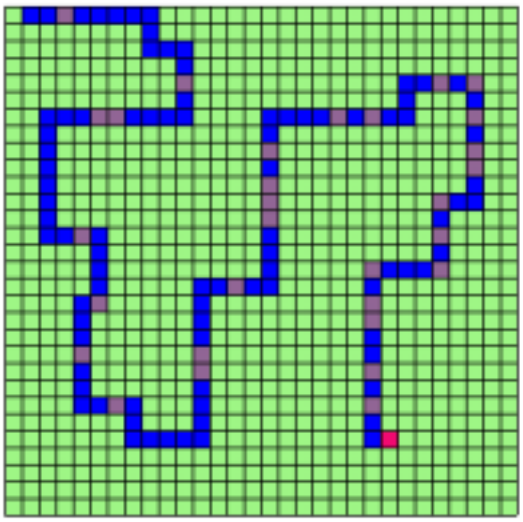


Graf 1.

Keďže voľba takej fitnes funkcie nedala zaujímavé výsledky, budeme sa naďalej zaoberať funkciou Fitness2 (Fitness2 := 2 * food + coef / (miss + 1)), ktorú sme už skôr spomínali. Hodnoty parametrov necháme také isté ako v predchádzajúcom prípade a použijeme znovu tú istú dráhu.

- Druhý pokus (použijeme funkciu Fitness2)

Trasa najlepšieho mravca:
(pridali sme mu niečo krokov navyše, vtedy prešiel celú dráhu, ale pravidlá sú správne)

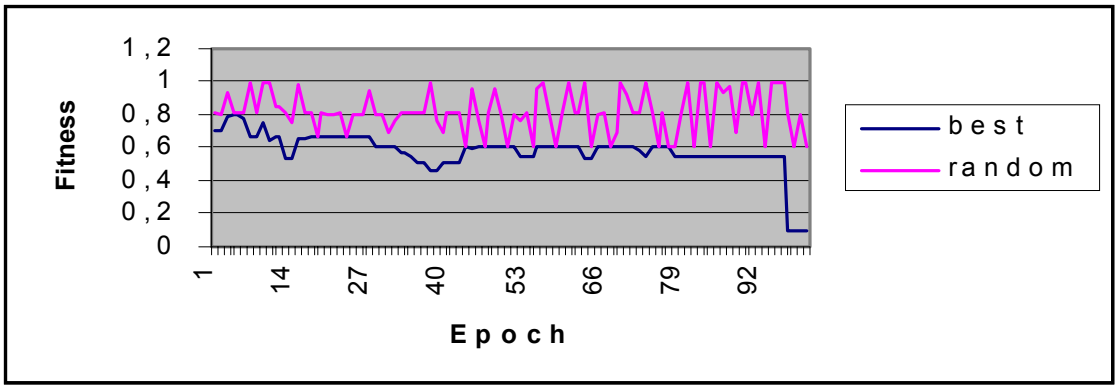


Obrázok 6.

Program najlepšieho mravca:

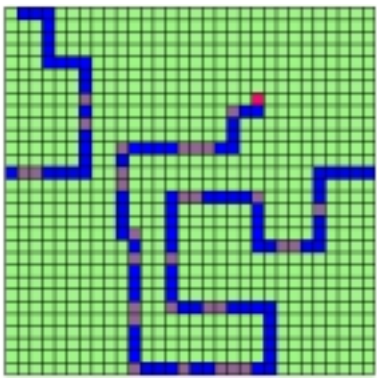
```
( IFSENSOR TURNRIGHT  ( PROGN  ( PROGN  ( PROGN TURNRIGHT TURNLEFT
TURNLEFT ) ( IFSENSOR MOVE  ( PROGN MOVE TURNLEFT TURNRIGHT ))TURNLEFT
) ( IFSENSOR  ( IFSENSOR MOVE  ( PROGN TURNRIGHT MOVE  ( IFSENSOR
TURNLEFT MOVE )))TURNRIGHT )TURNRIGHT ))
```

Graf najlepšieho a náhodného mravca:

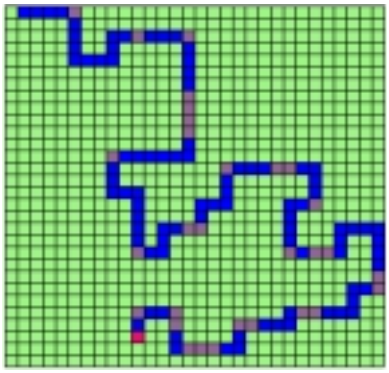


Graf 2.

Simulovaním najlepšieho mravca na iných dráhach, ktoré majú podobné vlastnosti ako dráha, ktorú sme doteraz používali, dostávame tiež výborné výsledky:



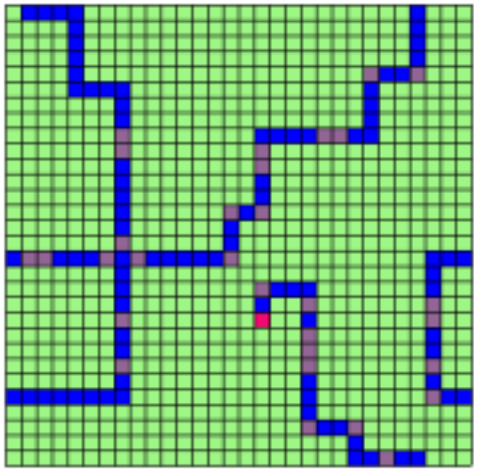
Dráha2



Dráha3

Obrázok 7,8.

- Ďalší pokus urobíme s trochu zložitejšou dráhou. Pohyb najlepšieho mravca:

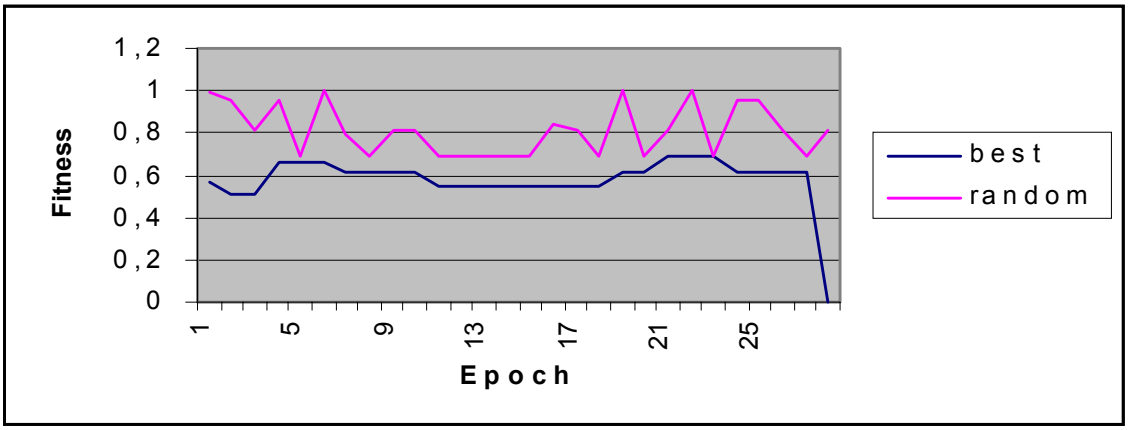


Obrázok 9. -Dráha4

Program mravca:

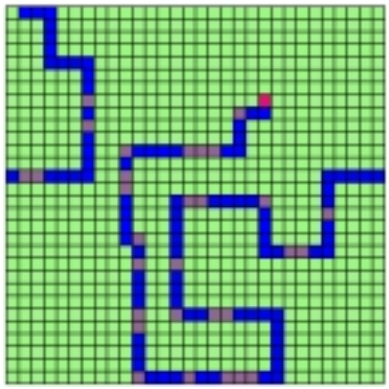
```
( PROGN  ( IFSENSOR MOVE  ( PROGN TURNLEFT TURNLEFT  ( IFSENSOR MOVE
TURNLEFT )))MOVE TURNLEFT )
```

Program celkom slušne fungoval aj na iných dráhach, i keď nie celkom presne ako predchádzajúci program mravca.

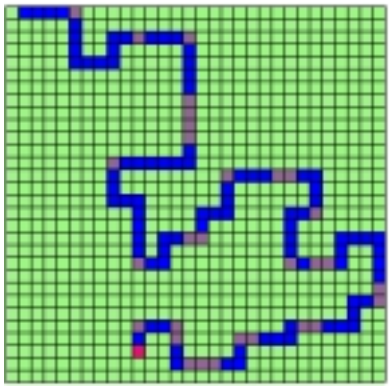


Graf 3.

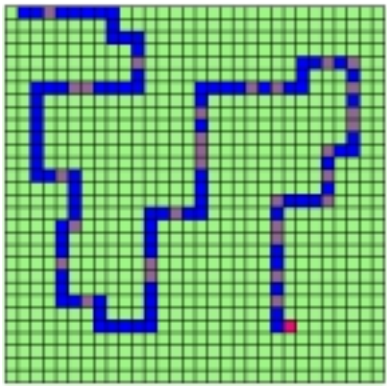
- Generovanie pravidiel pre tri dráhy:



Dráha1



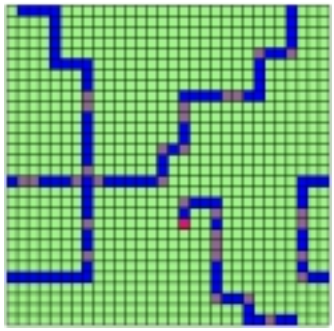
Dráha2



Dráha3

Obrázok 10,11,12.

Taktiež program dobre fungoval aj na dráhe, na ktorej nebol učený.

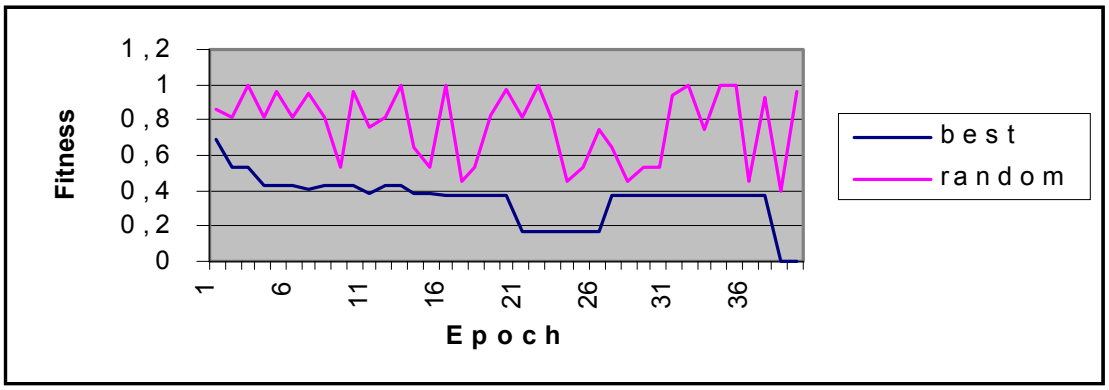


Obrázok 13.

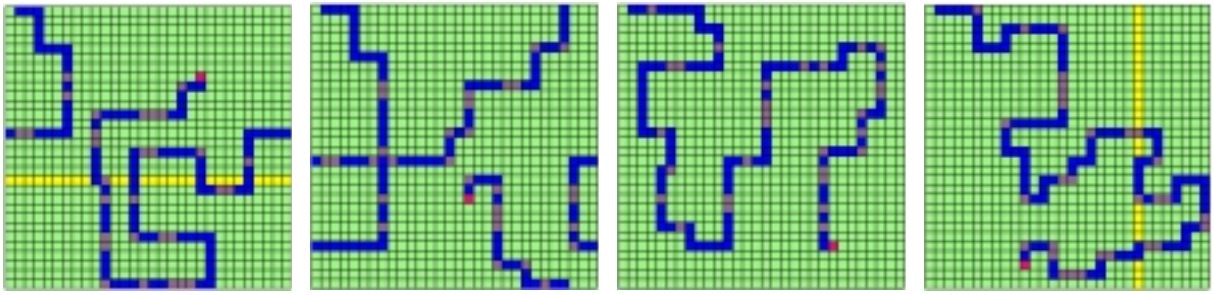
Program úspešného mravca:

```
( IFSENSOR TURNRIGHT  ( PROGN TURNLEFT MOVE  ( IFSENSOR TURNRIGHT  (
PROGN TURNLEFT  ( IFSENSOR TURNRIGHT TURNLEFT ) ( IFSENSOR  ( PROGN
MOVE  ( PROGN  ( IFSENSOR  ( IFSENSOR TURNRIGHT  ( PROGN  ( PROGN  (
IFSENSOR TURNLEFT  ( PROGN TURNRIGHT TURNRIGHT  ( PROGN MOVE TURNRIGHT
TURNLEFT ))) ( IFSENSOR TURNRIGHT MOVE ) ( IFSENSOR  ( IFSENSOR  (
PROGN TURNRIGHT TURNLEFT TURNRIGHT )TURNLEFT )TURNRIGHT )) ( PROGN
TURNLEFT  ( IFSENSOR TURNLEFT TURNRIGHT ) ( PROGN  ( PROGN MOVE  (
PROGN MOVE TURNLEFT TURNLEFT ) ( PROGN TURNLEFT TURNLEFT TURNRIGHT )) (
IFSENSOR  ( PROGN MOVE TURNLEFT MOVE ) ( IFSENSOR MOVE MOVE )) (
IFSENSOR  ( IFSENSOR TURNRIGHT TURNLEFT )TURNLEFT )))TURNLEFT )) (
IFSENSOR  ( PROGN TURNLEFT TURNRIGHT  ( PROGN  ( IFSENSOR TURNRIGHT
TURNRIGHT )TURNLEFT  ( PROGN MOVE MOVE  ( IFSENSOR TURNRIGHT  (
IFSENSOR TURNRIGHT TURNLEFT ))))) ( PROGN TURNLEFT  ( PROGN MOVE  (
PROGN  ( IFSENSOR  ( PROGN MOVE TURNRIGHT TURNLEFT )TURNLEFT )TURNRIGHT
MOVE )MOVE )MOVE )))MOVE  ( IFSENSOR MOVE  ( IFSENSOR  ( PROGN  ( PROGN
( PROGN TURNRIGHT TURNLEFT MOVE )TURNRIGHT MOVE ) ( PROGN MOVE  ( PROGN
MOVE TURNRIGHT TURNLEFT )TURNLEFT ) ( IFSENSOR  ( IFSENSOR  ( PROGN  (
IFSENSOR TURNRIGHT TURNRIGHT )TURNRIGHT TURNLEFT ) ( IFSENSOR  (
IFSENSOR MOVE MOVE ) ( IFSENSOR TURNRIGHT MOVE ))) ( IFSENSOR TURNLEFT
TURNRIGHT )))MOVE ))) ( PROGN TURNRIGHT MOVE MOVE ))TURNLEFT )))))
```

Graf 4.



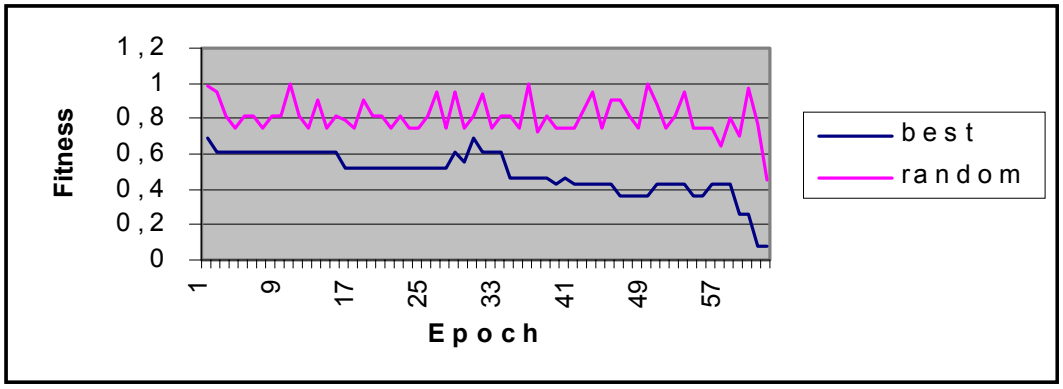
- Štvrtý pokus generovania pravidiel je na všetkých štyroch dráhach, ktoré sme mali doteraz: (pri niektorých som (už po vygenerovaní pravidiel) pri simulácii pridala kroky navyše a vtedy bol najlepší mravec úspešný)



Obrázok 14,15,16,17

Program:

```
( IFSENSOR TURNRIGHT  ( PROGN  ( PROGN TURNRIGHT TURNRIGHT  ( IFSENSOR
( IFSENSOR MOVE TURNLEFT ) ( IFSENSOR MOVE TURNRIGHT ))TURNRIGHT  (
PROGN TURNLEFT MOVE TURNRIGHT )))
```



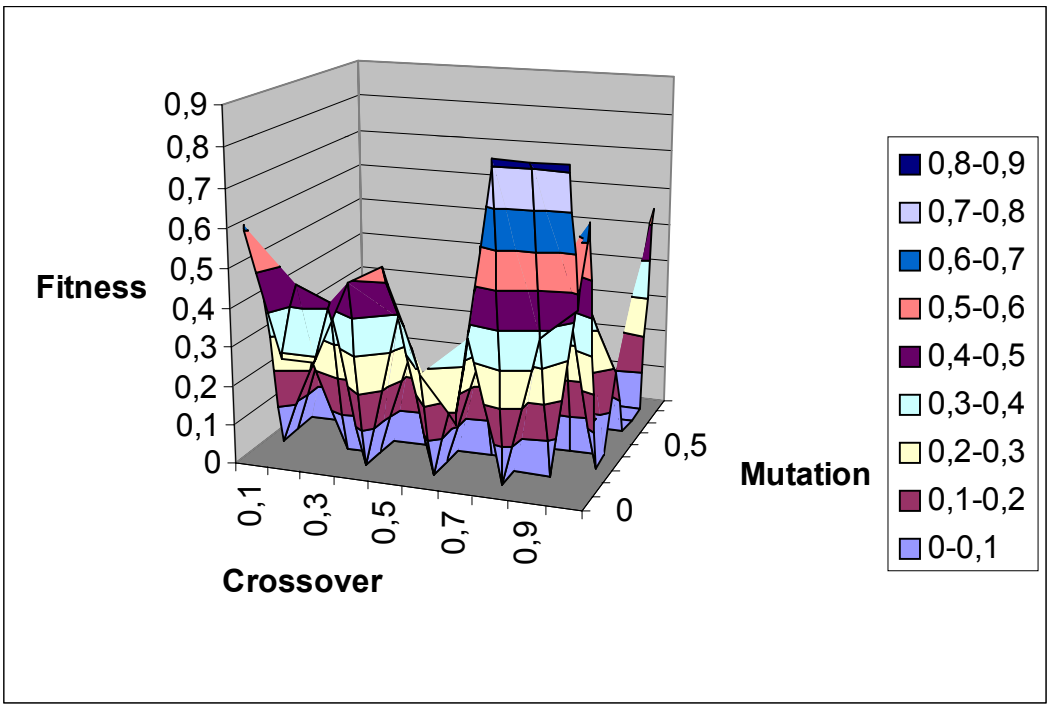
Graf 5.

Nasledujúce grafy ukazujú vplyv parametrov pravdepodobnosti kríženia a mutácie na úspešnosť genetického programovania, vyjadrenú normalizovanou fitnes funkciou (z intervalu <0,1>). Optimálny výsledok dosahuje hodnotu 0.

Všetky nasledujúce testy sú robené na Dráhe1, maximálna hĺbka stromu je 14, čas simulácie je 700. Pre každý vyber hodnôt je genetický algoritmus spustený 5 krát, v prvom prípade pozeráme len najlepší výsledok z piatich (Best), v druhom priemer výsledkov (Average).
V grafoch 6. a 7, je zvolená veľkosť populácie 300 a počet epoch 80.

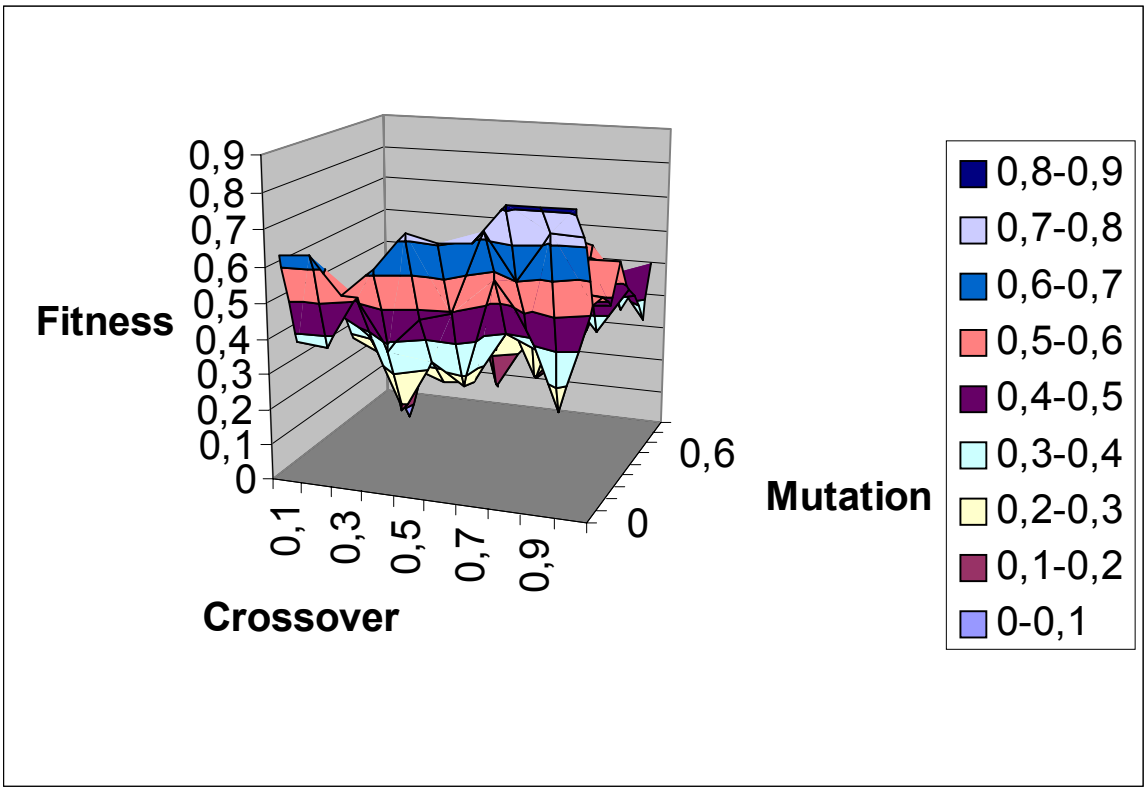
M\K	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
0	0,61468	0,28899	0,28899	0,49541	0,54128	0,28899	0,17431	0,81651	0,81651	0,81651
0,1	0,44037	0,44954	0,41284	0	0,39449	0	0,3578	0	0,38073	0,45872
0,2	0	0,22477	0	0	0,28899	0	0,27982	0	0	0,64679
0,3	0	0,2844	0,28899	0	0	0	0	0	0,51376	0
0,4	0	0	0	0	0	0	0	0	0	0
0,5	0	0	0	0	0	0	0	0	0,39449	0,19266
0,6	0	0	0	0	0	0	0	0	0	0
0,7	0	0,2844	0	0	0	0	0	0	0	0
0,8	0	0	0	0	0	0,36697	0	0	0	0
0,9	0	0	0	0	0	0	0,06422	0	0	0,54128

Tabuľka 1. Najlepšie výsledky



M\K	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
0	0,6367	0,6422	0,54128	0,61835	0,72294	0,70275	0,71101	0,81651	0,81651	0,81651
0,1	0,37248	0,36514	0,5156	0,37431	0,47156	0,49266	0,72477	0,6	0,73211	0,72661
0,2	0,36147	0,43394	0,37982	0,17798	0,32477	0,26605	0,57615	0,44404	0,22752	0,52844
0,3	0,33578	0,46055	0,33027	0,27156	0,24128	0,25138	0,4	0,36147	0,53761	0,52936
0,4	0,41835	0,33211	0,15963	0,23486	0,26972	0,48807	0,44679	0,50642	0,47523	0,48165
0,5	0,28624	0,24587	0,0578	0,38899	0,35963	0,29541	0,4422	0,37798	0,58349	0,58257
0,6	0,35963	0,21835	0,32202	0,28807	0,3367	0,36697	0,19358	0,45688	0,35046	0,4844
0,7	0,19633	0,20092	0,24771	0,12110	0,31376	0,32110	0,18349	0,47523	0,42385	0,44037
0,8	0,17982	0,09725	0,38532	0,41468	0,09358	0,22936	0,52018	0,27339	0,48073	0,34862
0,9	0,31927	0,28991	0,47156	0,45138	0,40917	0,40183	0,5633	0,53945	0,34495	0,49908

Tabuľka 2. Priemerné výsledky

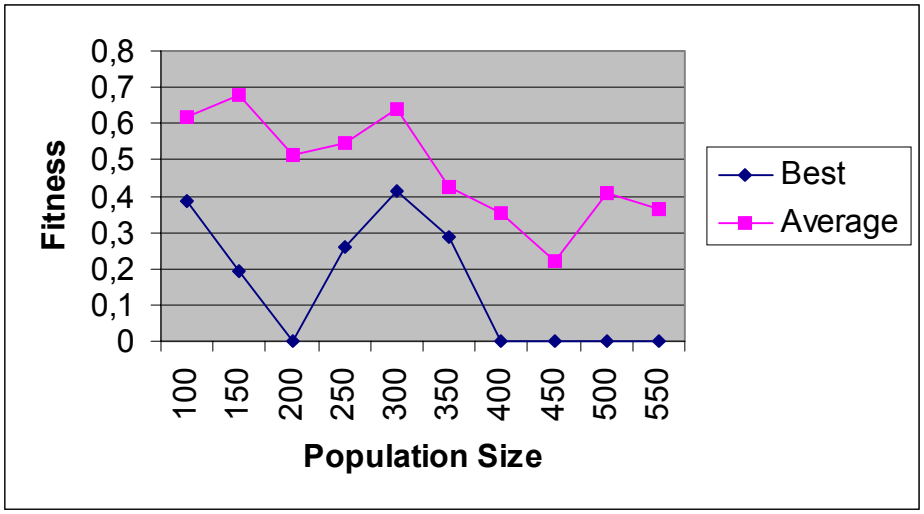


Graf 7.

V grafoch 8. a 9 je zvolená pravdepodobnosť mutácie 0.2 a pravdepodobnosť kríženia 0.8, počet epoch v grafe 8 je 80, a veľkosť populácie v grafe 9 je 350. Iné parametre sú také iste ako v predchádzajúcich pokusoch.

F\VP	100	150	200	250	300	350	400	450	500	550
Best	0,38532	0,19266	0	0,25688	0,41284	0,2844	0	0	0	0
Aver.	0,61835	0,67706	0,5156	0,54495	0,64128	0,42385	0,35229	0,22202	0,41101	0,36605

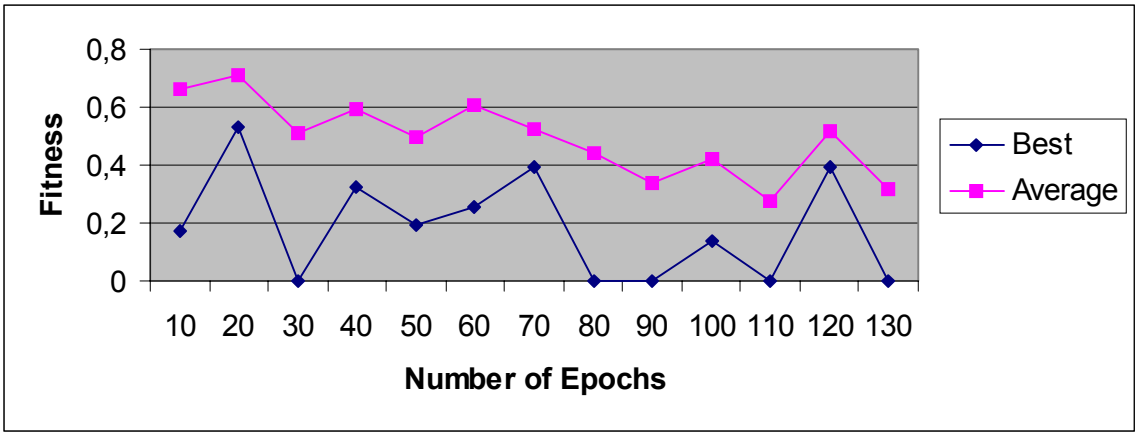
Tabuľka 3. Vplyv veľkosti populácie na úspešnosť metódy



Graf 8.

F\PE	10	20	30	40	50	60	70	80	90	100	110	120	130
Best	0,1743	0,5321	0	0,3257	0,1927	0,2569	0,3945	0	0	0,1376	0	0,3945	0
Aver.	0,6615	0,7138	0,5083	0,5899	0,4936	0,6055	0,5229	0,4385	0,3349	0,4239	0,2725	0,5174	0,3165

Tabuľka 4. Vplyv počtu epoch na výsledky



Graf 9.

Záver

Na záver môžeme povedať, že pri vhodne zvolených dráhach, pre problém generovania pravidiel pre mravca, metóda genetického programovania dala dobré výsledky. Pravidlá, pre pohyb a rozhodovanie mravca v mriežke, sú vhodné na reprezentáciu stromom a samým tým je prirodzené aj použitie genetického programovania na ich generovanie. Na prvý pohľad najlepšie riešenia nevyzerajú zrejmé a jednoduché, ale je veľmi pozitívne, že vygenerované pravidla sú robustné a všeobecné, t.j. vo väčšine prípadov fungujú správne aj na iných dráhach s podobnými vlastnosťami.
Nie moc prekvapujúce výsledky dostávame pri generovaní pravidiel pre viac dráh súčasne. Výsledky sú podobné ako pri generovaní pravidiel pre jednu dráhu, až na to, že je prípad pre viac dráh výpočtovo náročnejší.
Ostáva otázka, čí je možné vyriešiť aj komplikovanejšie problémy použitou metódou.

Vzhľadom na testovanie vplyvu vybraných parametrov na výsledky, výsledky testovania vplyvu pravdepodobnosti mutácie a kríženia v danom probléme sú trochu prekvapujúce. V priemernom prípade najlepší výsledok dala kombinácia Mutacia=0.5, Krizenie=0.3 i keď by sa mohlo očakávať, že pravdepodobnosť kríženia by mala byť väčšia ako pravdepodobnosť mutácie. Z druhej strany ak pozrieme tabuľku 1, väčšina kombinácii parametrov dosiahla minimum v piatich pokusoch, z čoho môžeme povedať, že nezáleží až tak veľmi na výberu parametrov mutácie a kríženia.
Čo sa týka veľkosti populácie a počtu epoch, s narastaním týchto hodnôt má fitnes funkcia tendenciu klesať, čo sa dalo aj očakávať. Ale zase aj s nízkymi hodnotami je možné niekedy dosiahnuť optimálny výsledok (ale je určite potrebné urobiť i viac pokusov, okrem toho aj náhoda hrá svoju rolu). Keďže tieto parametre značne vplývajú na výpočtovú zložitosť, treba nájsť nejaký kompromis medzi náročnosťou výpočtu a úspešnosťou výsledkov.

Literatúra:

1. V. Kvasnička, J. Pospíchal, P. Tiňo: Evolučné algoritmy, Vydavateľstvo STU, Bratislava, 2000
2. http://www2.informatik.uni-erlangen.de/~jacob/Evolvica/GP/Java/html/ant/ant.html
3. http://www.genetic-programming.com
4. H.-P. Schwefel, R. Männer (Eds.): Parallel Problem Solving from Nature, Springer-Verlag Berlin Heidelberg, 1991
5. D. Andre: Evolution of Mapmaking: Learning, Planning, and Memory Using Genetic Programming, 1994
6. J. R. Koza, F. H. Bennet III, D. Andre, M. A. Keane: Genetic Programming: Biologically Inspired Computation that Creatively Solves Non-Trivial Problems, 1999