Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný odbor: SOFTVÉROVÉ INŽINIERSTVO

GENETICKÝ ALGORITMUS

Case-Study z predmetu Evolučné Algoritmy

Autor: Bc. Tomáš Tatranský
Ročník: 2005/2006, letný semester
Dátum: 30. júna 2006

Zadanie

Téma 4. Genetický algoritmus pre funkciu

$$f(x) = 0.993851231 + e^{-0.01x^2} \sin(10x)\cos(8x)$$

pre $x \in$ [-10,10]. Premenná x je binárne reprezentovaná tak v štandardnom kódovaní, ako aj v Grayovom kódovaní. Pokuste sa nájsť minimálnu veľkosť populácie a také hodnoty pravdepodobností P_{cross} a P_{mut}, aby ste skoro so 100% istotou dostali globálne minimum s predpísaným minimálnym počtom krokov.

Riešenie

Na riešenie som použil Microsoft Visual Studio 2005 spoločne s .Net Framework 2.0. Program je napísaný v jazyku C# a disponuje prívetivým používateľským rozhraním. Na ďalších riadkoch popíšem jednotlivé komponenty programu. Textový výstup programu som použil na vytvorenie grafov v programe MS Excel 2007 Beta.

Implementácia chromozómu – Trieda Chromosome

Premenné

Trieda *Chromosome* reprezentuje jeden chromozóm obsahujúci binárnu reprezentáciu reálneho čísla. Základným prvkom je premenná *Bits* typu *BitVector32*. Táto premenná obsahuje 32 bitov, ku ktorým je možný prístup cez bitové masky. Ďalšia premenná *IsGray* označuje, či sú binárne dáta uložené pomocou Grayovho kódu alebo sú uložené štandardným spôsobom. Statické pole *Mask* zľahčuje prístup k jednotlivým bitom. Premenná *Fitness* reprezentuje hodnotu fitness daného chromozómu.

Funkcie

```
public Chromosome(double number, bool isgray)
```

Konštruktory implementujú inicializáciu chromozómov a ich vlastností.

```
public double Number
```

Vracia aktuálnu hodnotu chromozómu preloženú na reálne číslo s desatinnou čiarkou. Pomocou tejto Property je možné aj nastaviť novú hodnotu reálneho čísla do chromozómu.

```
public int BitsToInt
```

Konverzia bitového zápisu do prirodzeného čísla. Táto Property bola použitá pri

testovaní správnej funkčnosti chromozómu.

```
public String BitsToString
```

Konverzia bitového vektora na textový reťazec obsahujúci kompletný výpis bitov.

```
private BitVector32 ToGray(int p)
```

Preklad štandardného bitového zápisu do zápisu pomocou Grayovho kódovania.

```
private BitVector32 FromGray(BitVector32 gray)
```

Preklad Grayovho bitového zápisu späť do štandardného kódovania.

```
private int DoubleToInt(double input)
```

Konverzia reálneho čísla typu Double na 32 bitový integer.

```
private double BitsToDouble(BitVector32 input)
```

Konverzia bitového zápisu z chromozómu na reálne číslo.

```
public String DumpString()
```

Výpis pomocných informácií o chromozóme, teda hlavne binárneho zápisu, reprezentácie v integeri a reprezentácie reálneho čísla.

```
public ArrayList Cross(Chromosome orig, double probability, Random RandObj)
```

Táto funkcia je určená na kríženie dvoch chromozómov. Kríženie je jednobodové a prebehne so zadanou pravdepodobnosťou na náhodne zvolenom mieste chromozómov. Funkcia generuje nové chromozómy vytvorené krížením.

```
public void Mutate(double probability, Random RandObj)
```

Funkcia mutácie funguje tak, že so zadanou pravdepodobnosťou mutuje každý bit v reťazci chromozómu.

```
public int CompareTo(object obj)
```

Táto funkcia je implementovaná aby bolo možné jednoducho usporiadať zoznam chromozómov podľa ich fitness. Funkcia iba určuje podľa ktorej položky sa budú chromozómy usporiadať. Samotné triedenie je vstavané v systéme C#.

Implementácia Genetického algoritmu – Trieda GeneticAlg

Premenné

```
private ArrayList Chromosomes; – zoznam chromozómov
private ArrayList ChSelection; – dočasné úložisko novej generácie
private int Population; – veľkosť populácie
private double CrossProbability; – pravdepodobnosť kríženia chromozómov
private double MutationProbability; – pravdepodobnosť mutácie jedného bitu
private Random RandObj; – objekt na generovanie náhodných čísel
private bool GrayCode; – nastavenie štandardného alebo Grayovho kódovania
private int Generations; – maximálny počet generácií
private Chromosome bestChromosome; – najlepší chromozóm zo všetkých
generácií
private Chromosome currentBestCh; – najlepší chromozóm aktuálnej generácie
private Chromosome lastBestCh; – najlepší chromozóm minulej generácie
private double Deviation; – požadovaná maximálna odchýlka od globálneho
minima
```

Funkcie

```
public double BestX
```

Vracia hodnotu x, ktorá patrí najlepšiemu chromozómu zo všetkých generácií.

```
public double BestXFitness
```

Vracia Fitness pre najlepší chromozóm zo všetkých generácií.

```
public GeneticAlg(int p, int g, double cp, double mp, double dev, bool
gray, Random randObj)
```

Konštruktor triedy nastavuje potrebné premenné.

```
public int Run()
```

Spustenie genetického algoritmu. Najprv sa zavolá *Init*(), následne sa spočíta fitness pomocou *CountFitness*(), overí sa, či je splnená podmienka a ak áno cyklus sa končí. Inak sa generuje nová generácia pomocou funkcií *Selection*(), *Crossover*(), *Mutation*() a nakoniec táto nová generácia nahradí pôvodnú vo funkcii *Replace*(). Toto sa opakuje najviac *Generations* krát.

```
public void Init()
```

Funkcia generuje náhodné chromozómy a tak vytváran nultú generáciu.

```
private void PrintToOutput(int i, Form1 output)
```

Funkcia je použitá na výpis užitočných údajov do textového rámca.

```
public void CountFitness()
```

Výpočet hodnoty *Fitness* pre každý chromozóm. Je vyhodnotená funkcia pomocou funkcie *EvalFunction*(). Chromozómy sa usporiadajú podľa hodnoty Fitness. Určí sa najlepší chromozóm aktuálnej generácie a v prípade, že je najlepší aj pre všetky generácie, tak sa nastaví tiež.

```
public bool EndCondition()
```

Ukončovacia podmienka zisťuje, či sme našli minimum s danou presnosťou.

```
public void Selection()
```

Výber chromozómov, ktoré postupujú do procesu reprodukcie je realizovaný pomocou rulety. Chromozómy sú zoradené podľa hodnoty Fitness. Najlepší chromozóm má priradené v rulete najvyššie číslo, každý ďalší chromozóm má číslo o jedno menšie. Zo súčtu všetkých priradených čísel ktorý je vlastne súčtom radu sa náhodne vyberie jedno číslo v rozmedzí 0 až súčet. Toto číslo je následne pomocou matematickej rovnice prepočítané na konkrétny chromozóm.

Pôvodne som ruletu riešil tak že, každý chromozóm mal priradenú hodnotu pomocou prevrátenej hodnoty fitness. V tomto prípade bol ale výber veľmi slabý a často sa stávalo, že lepší chromozóm sa vôbec nedostal do novej generácie. Spôsob, ktorý som implementoval ako druhý sa ukázal byť podstatne optimálnejší.

```
public void Crossover()
```

Funkcia prechádza cez pole chromozómov vybraných na reprodukciu a vždy dva chromozómy pošle na kríženie funkcii *Chromosome.Cross*(). Vrátené skrížené alebo iba skopírované chromozómy vkladá do poľa pripravenej nasledujúcej generácie.

```
public void Mutation()
```

Funkcia necháva zmutovať každý chromozóm z poľa chromozómov novej generácie.

```
public void Replace()
```

Výmena poľa starej generácie z apole s novými chromozómami.

```
public double EvalFunction(double x)
```

Výpočet funkčných hodnôt funkcie zadanej v zadaní.

Trieda prehľadávania priestoru – SearchingSpace

Poslednou triedou, okrem triedy, ktorá implementuje používateľské rozhranie, je trieda použitá na prehľadanie priestoru nastavení rôznych parametrov, najmä však parametrov pravdepodobnosti kríženia a mutácie, počtu chromozómov v populácií a nastavenia kódovania binárnej informácie. Trieda je veľmi jednoduchá. Ide v podstate o jedinú funkciu, ktorá postupne mení parametre a spúšťa stále znova a znova genetický

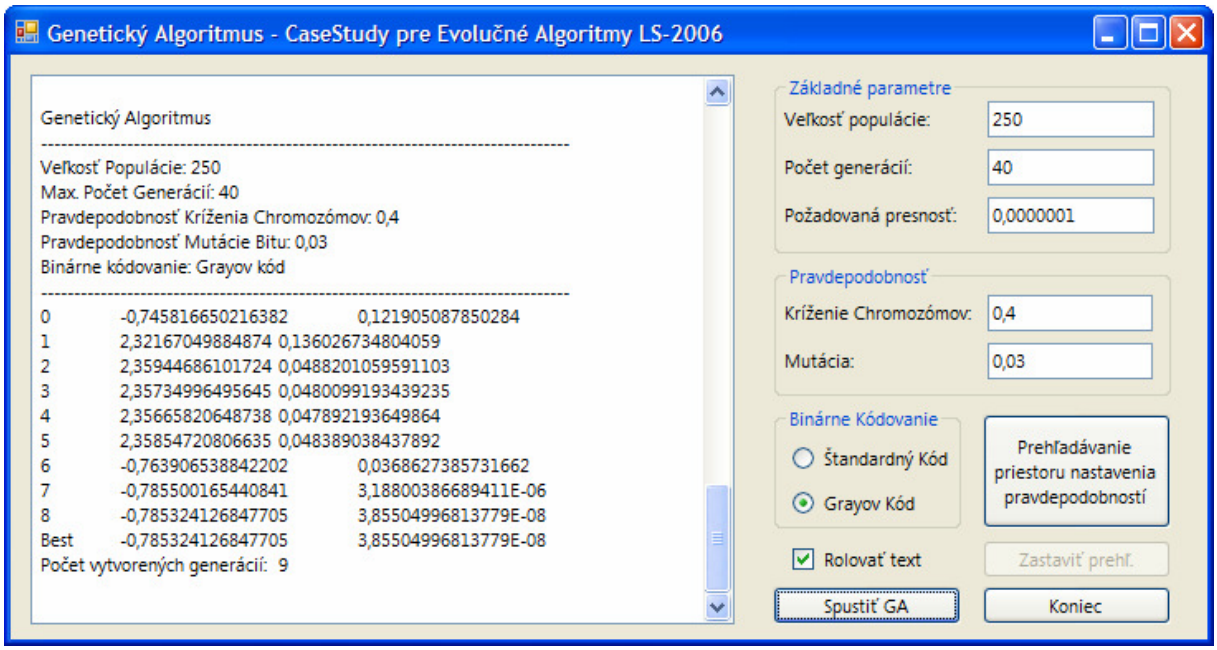
algoritmus. Štatistiku vypisuje do textového rámca na obrazovku.

Takou zaujímavosťou je, že táto trieda sa spúšťa ako samostatné vlákno (thread) a neblokuje tak používateľské prostredie. Nie je problém ani ukončiť prehľadávanie alebo ukončiť celý program. Toto programové vlákno sa vytvára v triede implementujúcej používateľské rozhranie, kde bolo potrebné aj riešiť zápis do textového rámca vytvoreného v inom vlákne a teda bolo potrebné vytvoriť tzv. delegáta, ktorý preberie text z dcérskeho vlákna a už v pôvodnom vlákne tento text potom vloží do textového rámca.

Trieda generuje textový výpis v podobe riadkov obsahujúcich aktuálne nastavenie parametrov genetického algoritmu a percentuálnu úspešnosť dosiahnutia minima. Ďalšie číslo reprezentuje najvyšší dosiahnutý počet generácií, ktorý ešte viedol k úspešnému nájdeniu minima funkcie. V prípade stopercentnej úspešnosti teda vlastne udáva koľko najmenej generácií je potrebných na túto úspešnosť.

Používateľské prostredie – Trieda Form1

Na obrázku č. 1 je zobrazené používateľské prostredie programu. Program je možné ovládať pomocou tlačítok a je možné voliť vlastné hodnoty pre hlavné parametre. Je možné buď spustenie genetického algoritmu so zadanými hodnotami alebo spustenie prehľadávania pravdepodobnostných hodnôt, kedy sa použijú používateľom zvolené základné parametre a priradia sa k postupne sa meniacim hodnotám pravdepodobností, ktoré sú určované z množín definovaných v programe. Následne sa pre každé priradené hodnoty parametrov opakovanými spusteniami genetického algoritmu zisťuje percentuálna úspešnosť nájdenia minima funkcie. Výpis výsledkov sa zobrazuje v textovom okne, odkiaľ je možné tieto výsledky kopírovať.



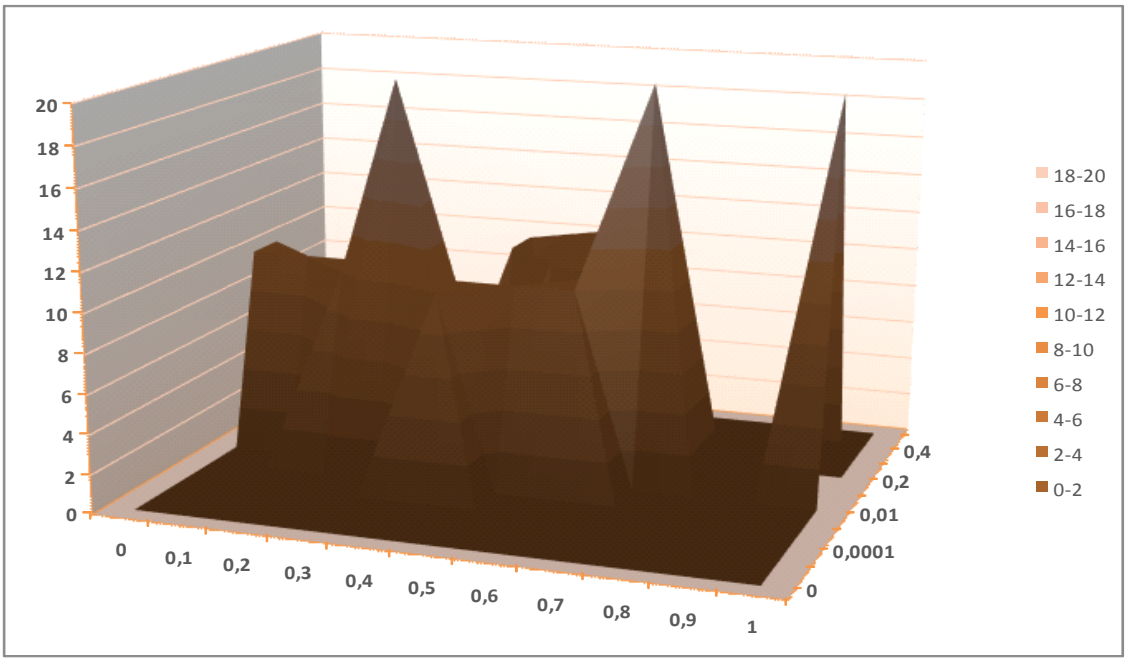
Obr. 1 Používateľské prostredie programu

Zisťovanie vplyvu parametrov

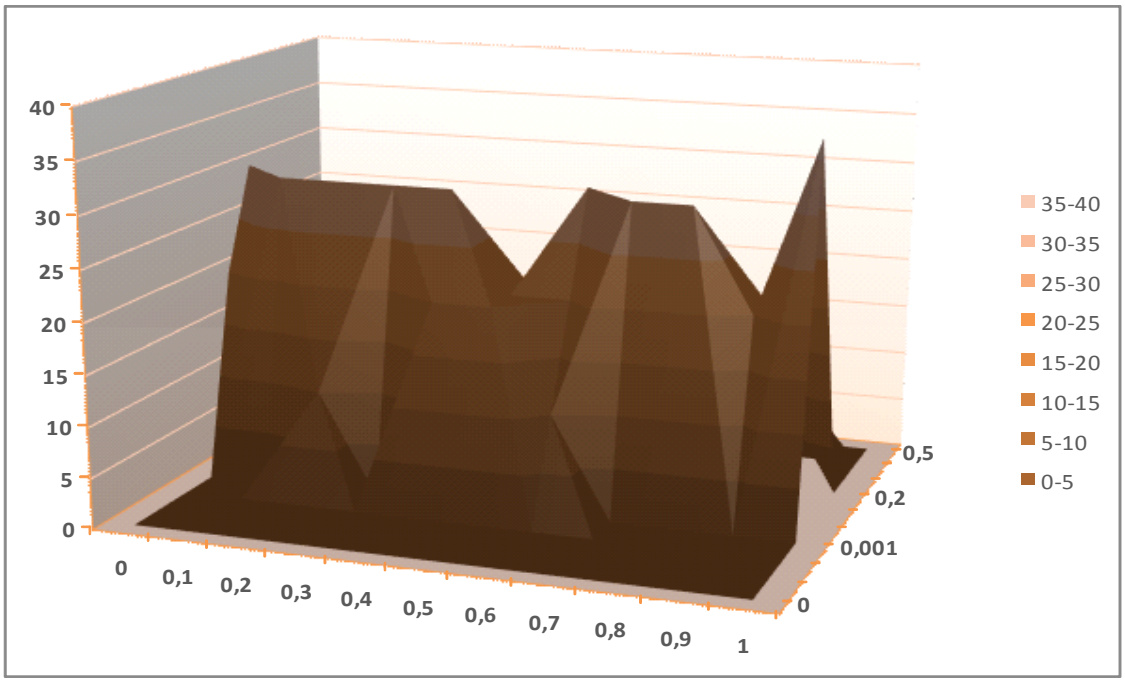
Ako zo zadania vyplýva, zaoberal som sa najmä hľadaním správnych hodnôt pre pravdepodobnosti kríženia a mutácie a tiež pre počet chromozómov v populácií a to rovnako pre štandardné aj grayovo kódovanie binárnej repprezentácie. Na zistenie vplyvu parametrov som vytvoril program, ktorý zozbieral údaje pre rozsah počtu chromozómov v populácií od 10 až po 1000, pre rozsah pravdepodobnosti mutácie bitu od 0 až po 0,5 a pre rozsah pravdepodobnosti prekríženia od 0 až po 1. Presné použité hodnoty pravdepodobností je možné vidieť v tabuľkách na konci kapitoly.

Získané hodnoty úspešnosti som zaniesol do grafov, ktoré sú zobrazené na nasledujúcich stranách. Vzhľadom na to, že získaných údajov bolo veľké množstvo, grafy sú vytvorené iba pre vybrané veľkosti populácií.

Na prvých dvoch grafoch môžeme vidieť výsledky pre štandardné a grayovo kódovanie pri počte 10 chromozómov v populácií. Výsledná nanesená hodnota je úspešnosť dosiahnutia minima počas 400 generácií v percentách. Povolenú odchýlku od minima som určil na 10^{-7}.

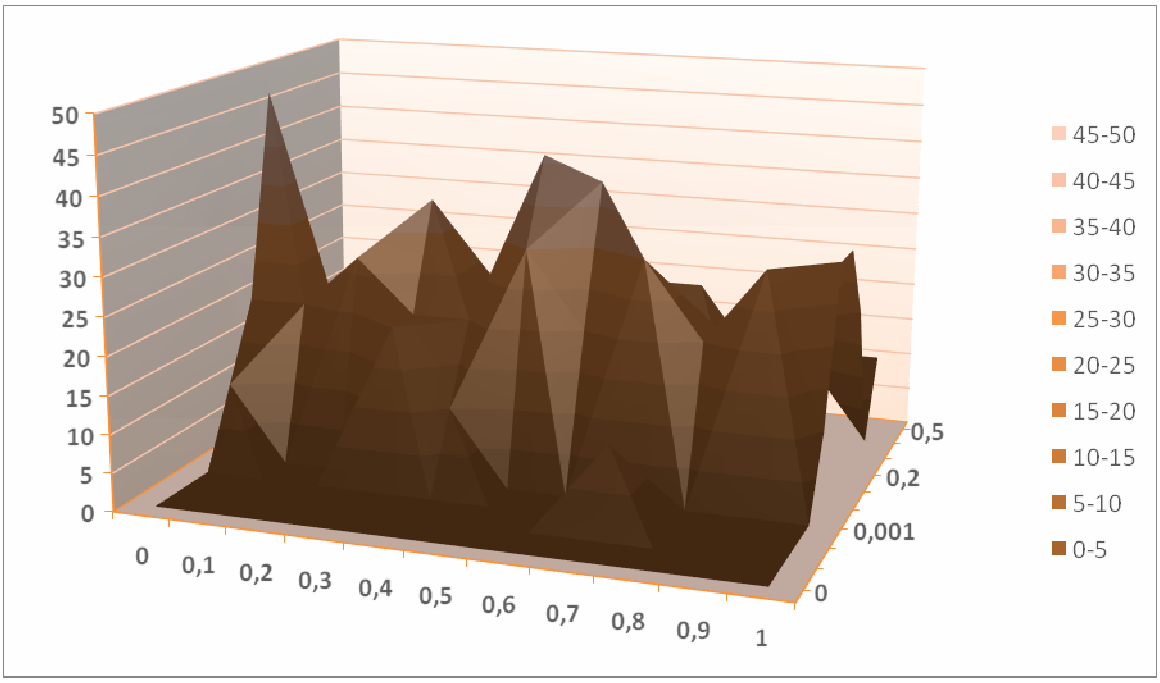


Úspešnosť GA v percentách, 10 chromozómov, štandardné kódovanie

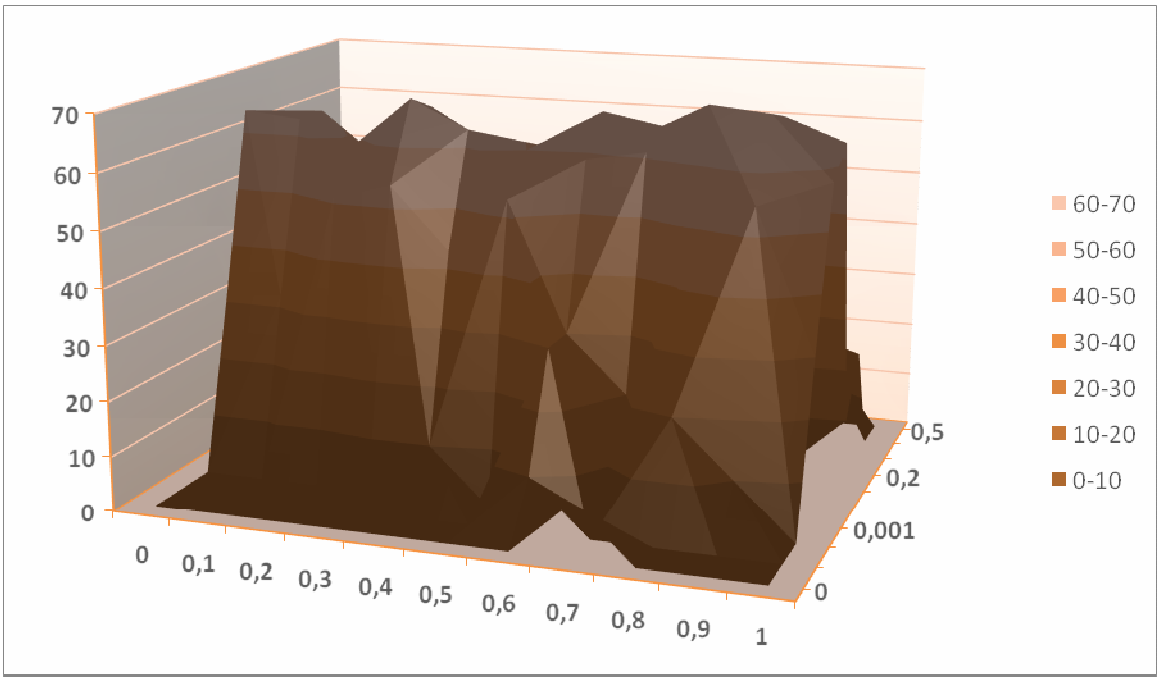


Úspešnosť GA v percentách, 10 chromozómov, Grayovo kódovanie

Získavanie údajov úspešnosti pre každú sadu parametrov bolo rozdelené na dve časti. V prvej časti sa vyhodnotilo prvých 10 pokusov genetického algoritmu. Ak z nich boli úspešné aspoň 4 pokusy, potom bolo vykonaných ďalších 90 pokusov s rovnakými parametrami. Výsledné číslo teda reprezentuje presný počet úspešných pokusov zo 100 vykonaných. Ak z prvej desiatky neboli úspešné ani 4 pokusy, ďalšími sa už nepokračovalo aby sa tak ušetril čas na získavanie užitočnejších výsledkov.



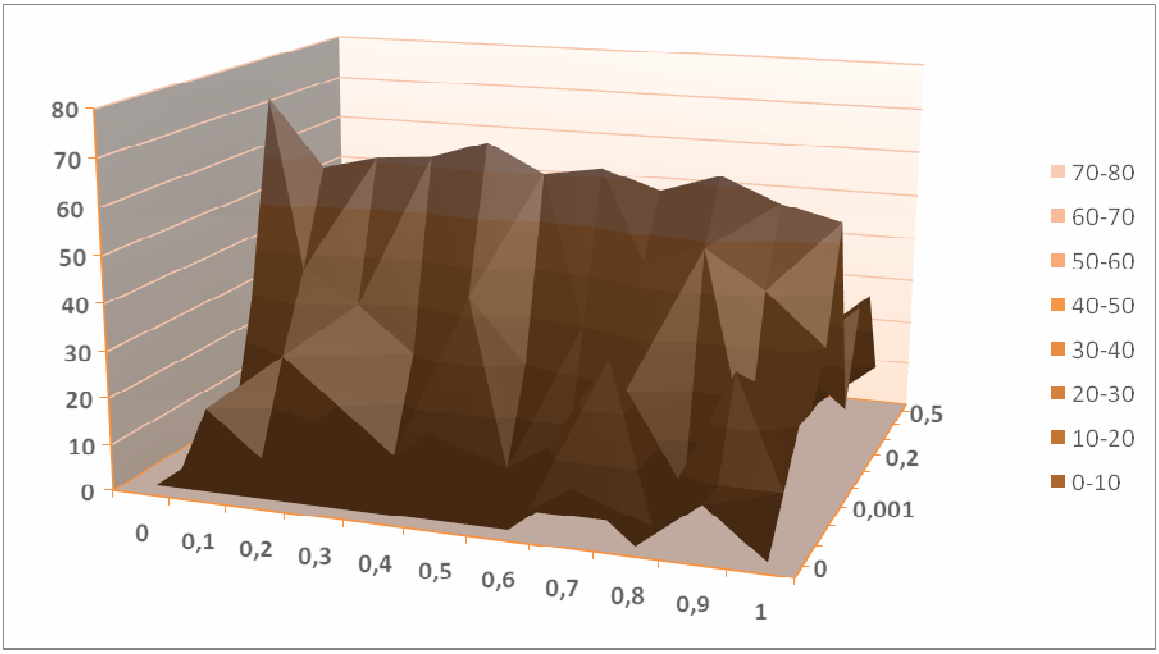
Úspešnosť GA v percentách, 50 chromozómov, štandardné kódovanie



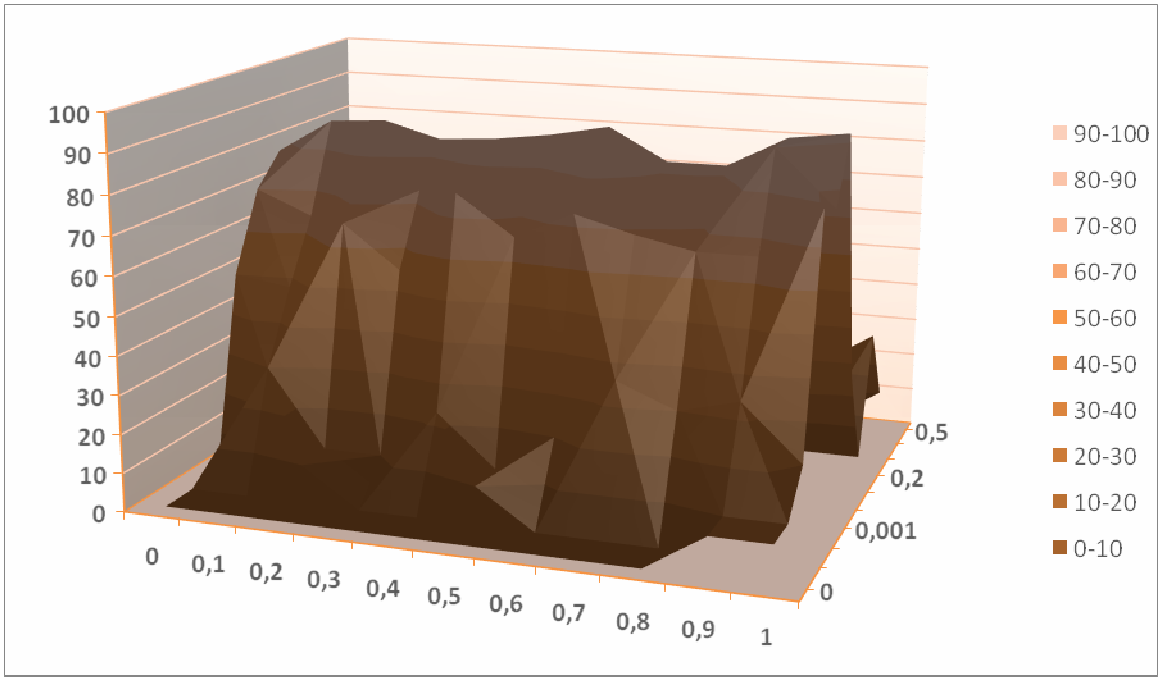
Úspešnosť GA v percentách, 50 chromozómov, Grayovo kódovanie

Na pvých grafoch zobrazujúcich výsledky pre 10 a 50 chromozómov v populácií vidíme značný rozdiel v úspešnostiach medzi použitím štandardného binárneho kódovania a použitím Grayovho kódovania, ktoré je prispôsobené tak, že susediace hodnoty sa v binárnom zápise líšia vždy iba v jednom bite.

Ukázalo sa, že Grayovo kódovanie podstatne zlepšuje výkonnosť genetického algoritmu, pričom zároveň dochádza k určitej výpočtovej penalizácií pri výpočte hodnôt fitness funkcie pretože je potrebný preklad z Grayovho kódovania na štandardné kódovanie. Táto penalizácia ale nie je vysoká a je prevážená výhodami v podobe nižšieho počtu chromozómov v generácií a nižšieho maximálneho počtu generácií na dosiahnutie rovnakých hodnôt úspešnosti ako pri modeli so štandardným kódovaním.

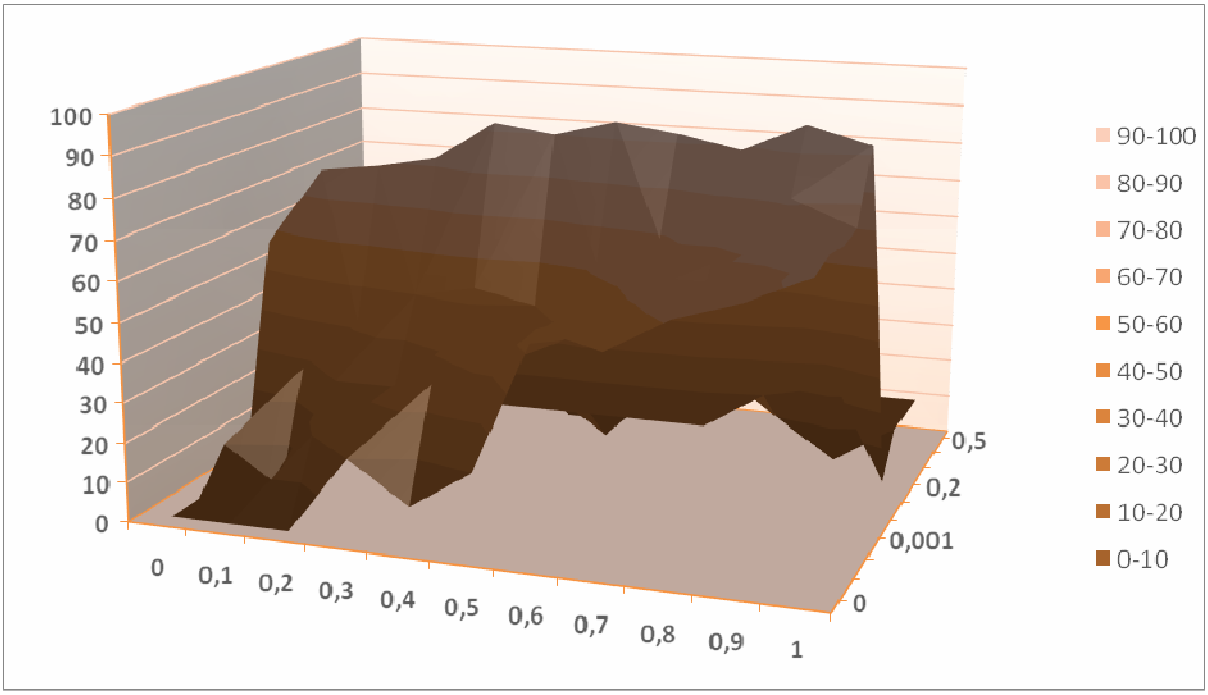


100 chromozómov, štandardné kódovanie

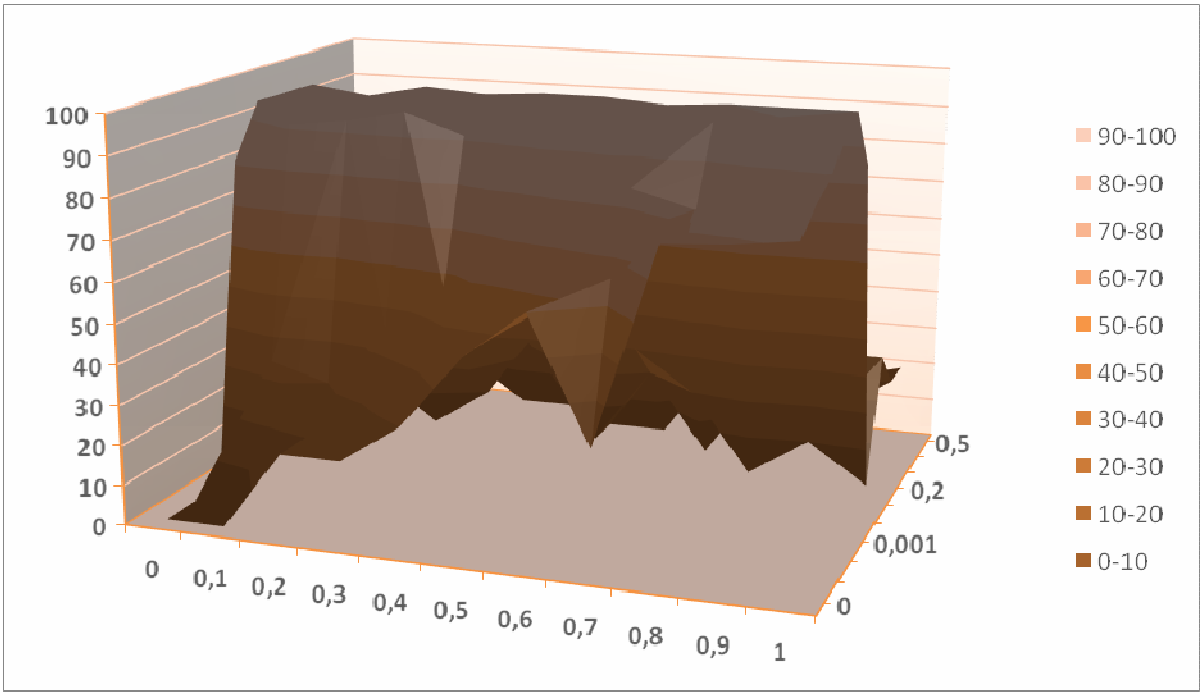


100 chromozómov, Grayovo kódovanie

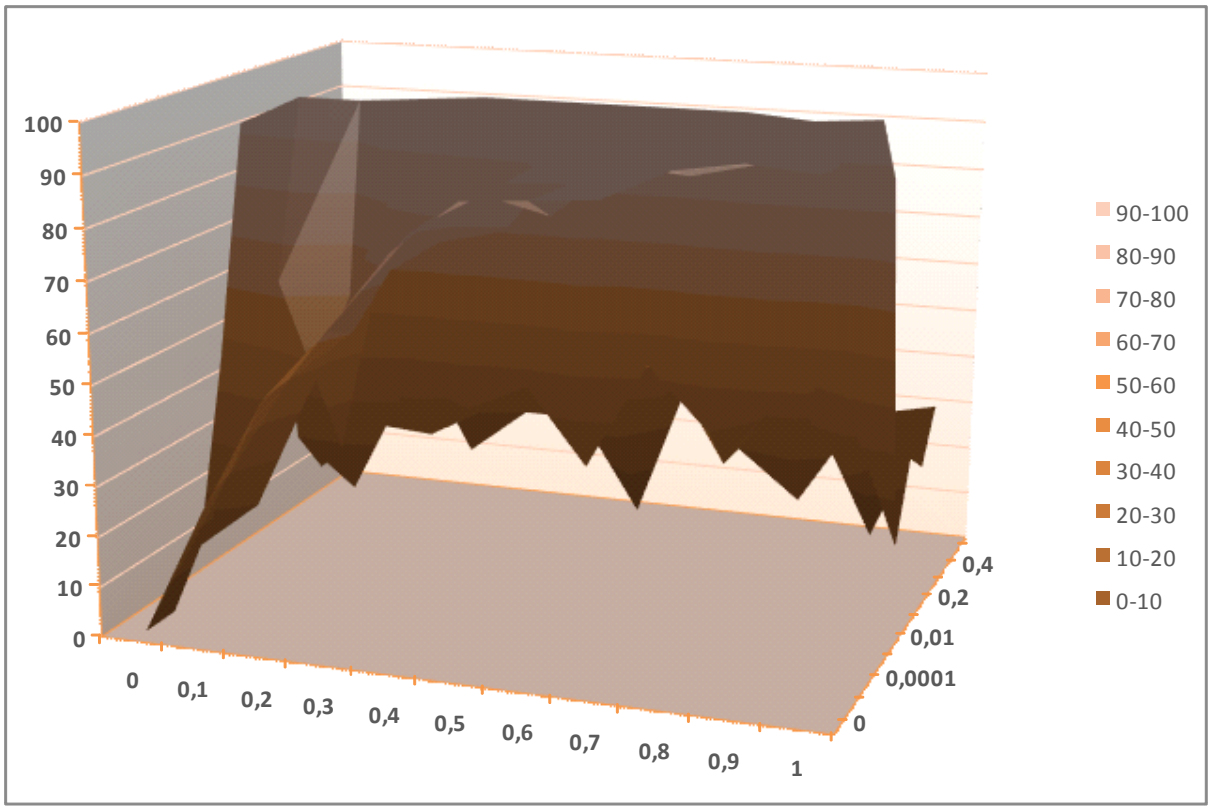
Pri 200 a viac chromozómoch v populácií som znížil maximálny počet generácií zo 400 na 100 aby bol zber údajov urýchlený. Výsledkov sa to ale príliš nedotklo vzhľadom na to, že konvergencia nastáva obyčajne pred hranicou 100 generácií.



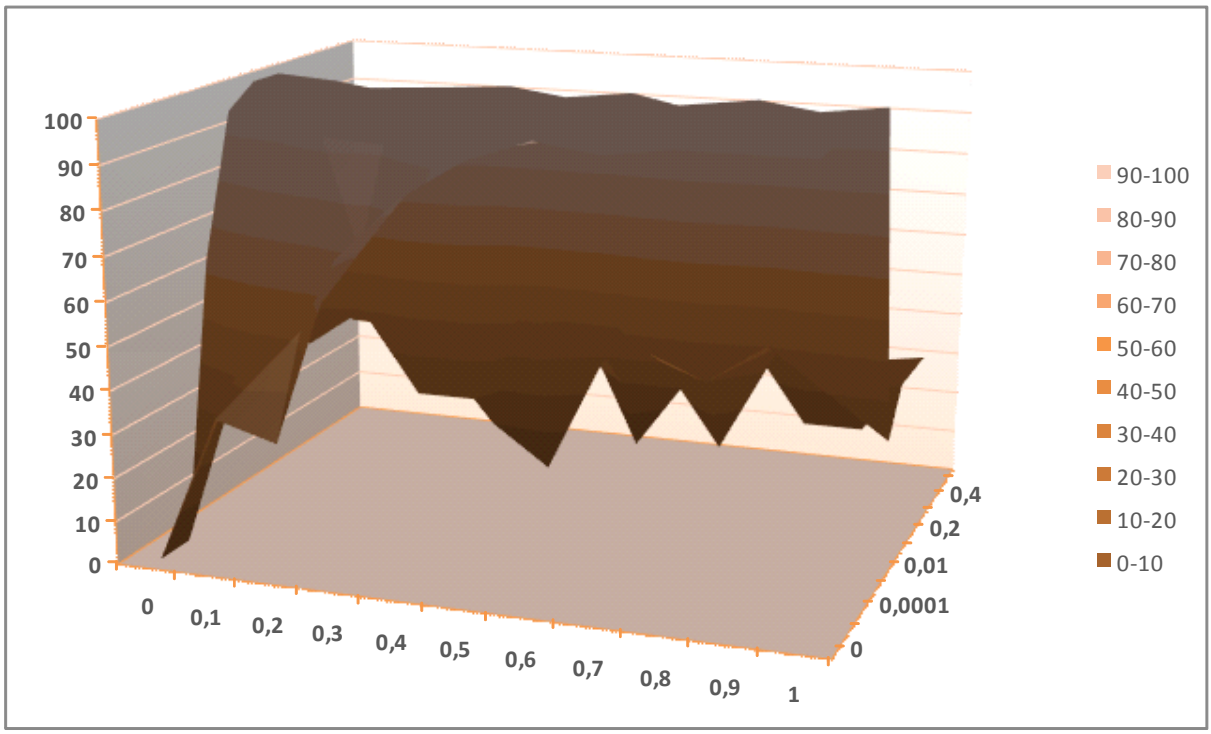
400 chromozómov, štandardné kódovanie



400 chromozómov, Grayovo kódovanie



1000 chromozómov, štandardné kódovanie



1000 chromozómov, Grayovo kódovanie

Pravd. mutácie	Pravdepodobnosť kríženia										
	0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
0	0	26	49	63	79	89	90	91	97	99	98
1,00E-05	0	30	50	60	81	88	85	95	94	97	97
0,0001	10	20	47	68	76	79	96	97	99	99	100
0,001	46	63	30	86	88	90	94	95	95	99	100
0,01	92	98	98	99	100	100	100	100	100	99	100
0,1	86	88	89	95	86	90	88	91	85	86	87
0,2	20	10	35	30	36	20	43	30	35	44	10
0,3	10	21	20	27	39	35	41	20	38	20	25
0,4	37	30	32	33	20	20	34	20	10	30	20
0,5	20	33	10	20	20	0	35	32	26	0	30

Tabuľka údajov z predposledného grafu, 1000 chromozómov, štandardné kódovanie, zobrazuje percentuálnu úspešnosť nájdenia minima pre jednotlivé pravdepodobnosti

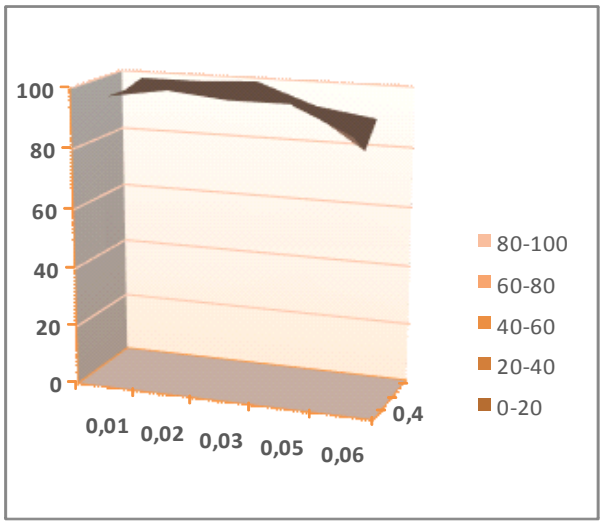
Pravd. mutácie	Pravdepodobnosť kríženia										
	0	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
0	0	34	30	72	86	95	100	98	100	100	100
1,00E-05	0	43	52	71	89	95	98	100	100	100	100
0,0001	63	81	93	92	96	100	100	100	100	100	100
0,001	95	100	100	100	100	100	100	100	100	100	100
0,01	100	100	100	100	100	100	100	100	100	100	100
0,1	100	99	97	99	100	98	100	94	100	98	100
0,2	30	40	20	20	50	47	36	30	40	30	20
0,3	38	35	30	10	0	30	30	10	37	33	30
0,4	30	30	20	27	29	20	20	30	43	31	30
0,5	26	20	26	30	30	0	20	30	10	10	30

Tabuľka údajov z posledného grafu, 1000 chromozómov, Grayovo kódovanie, zobrazuje percentuálnu úspešnosť nájdenia minima pre jednotlivé pravdepodobnosti

Hľadanie najlepších parametrov

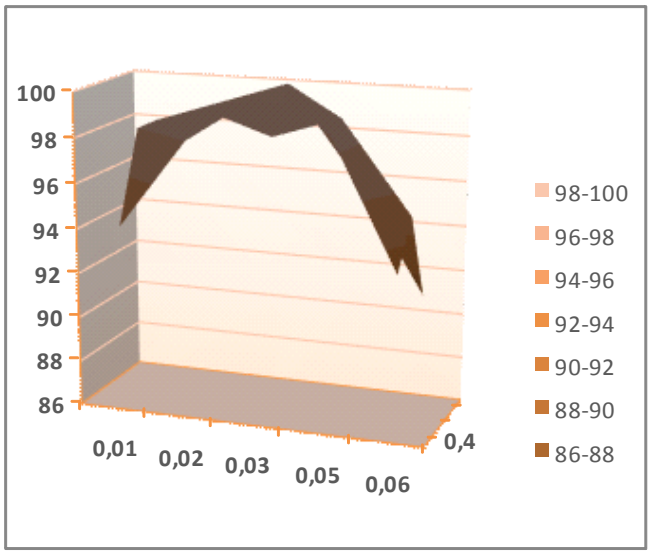
Po úvodnej dávke testov a vyhodnotení grafov som zistil, že Grayovo kódovanie získava naozaj podstatne lepšie výsledky. Pre parametre pravdepodobnosti sa ukázala ako najdôležitejšia hodnota pravdepodobnosti mutácie, kde do úvahy prichádzali najmä hodnoty 0,1 a 0,01. Pre pravdepodobnosť kríženia boli rozdiely minimálne, aj keď najlepšie výsledky dosahoval algoritmus s hodnotami 0,4, 0,5, a 0,6.

V ďalších testoch som upravil množiny testovaných parametrov, tak aby som sa zameral najmä na rozpätie pravdepodobnosti mutácie 0,1 až 0,01. V tomto rozpätí nakoniec výsledky ukázali, že voľba 0,03 je najlepšia.



Pravd. mutácie	Pravdepodobnosť kríženia		
	0,4	0,5	0,6
0,01	97	97	99
0,02	100	99	99
0,03	98	100	100
0,05	98	95	93
0,06	89	82	90

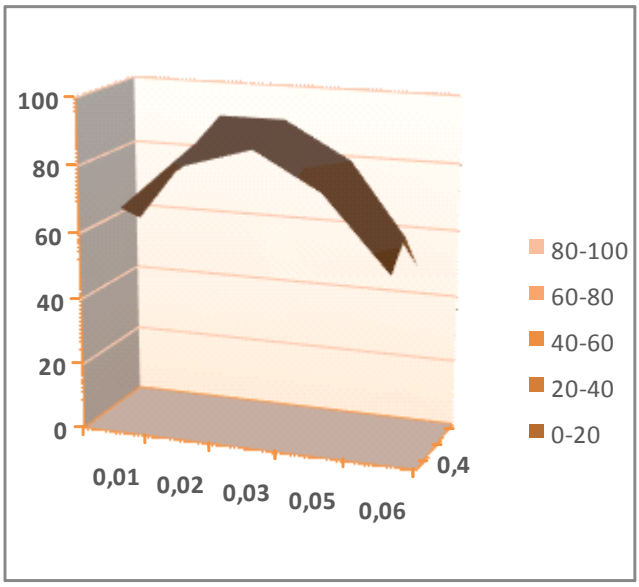
Úspešnosť GA pre populáciu 300 chromozómov, max. počet generácií 40, grayovo kódovanie



Pravd. mutácie	Pravdepodobnosť kríženia		
	0,4	0,5	0,6
0,01	94	98	98
0,02	98	99	99
0,03	100	98	100
0,05	99	99	96
0,06	93	95	91

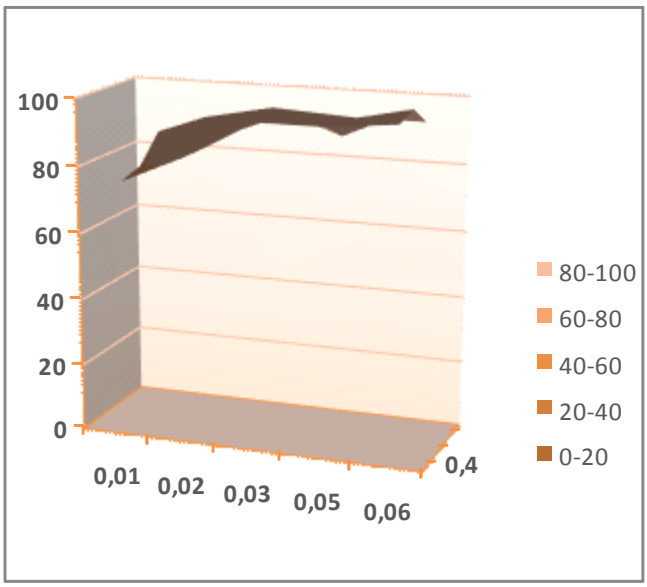
Úspešnosť GA pre populáciu 250 chromozómov, max. počet generácií 40, grayovo kódovanie

Pri Grayovom kódovaní som sa nakoniec dopracoval k minimálnym hodnotám populácie a počtu generácií pri takmer stopercentnej istote nájdenia minima. Populáciu som stanovil na 250 chromozómov a maximálny počet generácií na 40, pri pravdepodobnosti mutácie 0,03 a pravdepodobnosti kríženia 0,4.



Pravd. mutácie	Pravdepodobnosť kríženia		
	0,4	0,5	0,6
0,01	67	61	68
0,02	81	87	91
0,03	87	85	91
0,05	76	75	80
0,06	54	62	50

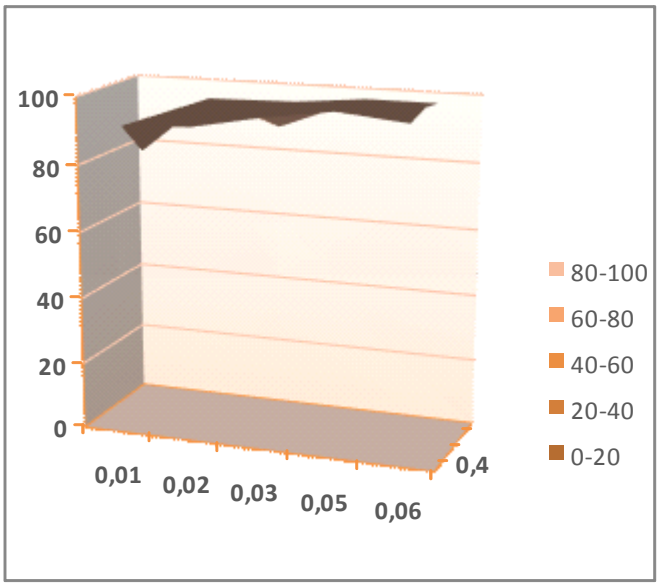
Úspešnosť GA pre populáciu 250 chromozómov, max. počet generácií 40, štandardné kódovanie



Pravd. mutácie	Pravdepodobnosť kríženia		
	0,4	0,5	0,6
0,01	75	77	85
0,02	84	93	85
0,03	95	97	93
0,05	95	90	93
0,06	97	99	93

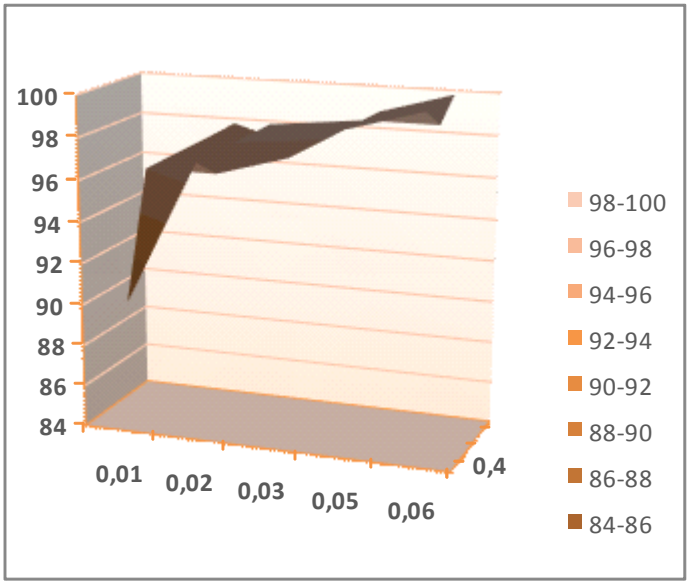
Úspešnosť pre populáciu 300 chromozómov, max. počet generácií 100, štandardné kódovanie

Pre štandardné binárne kódovanie boli výsledky podstatne horšie. Pri rovnakom nastavení dosahovala úspešnosť tohto modelu iba okolo 87%, čo je poriadne vzdialené od ideálnej stovky. Pri populácií 300 chromozómov a zvýšení počtu generácií na 100 bola najlepšia úspešnosť stále iba okolo 95%.



Pravd. mutácie	Pravdepodobnosť kríženia		
	0,4	0,5	0,6
0,01	91	81	85
0,02	92	98	93
0,03	96	91	96
0,05	99	99	98
0,06	97	99	98

Úspešnosť pre populáciu 400 chromozómov, max. počet generácií 100, štandardné kódovanie



Pravd. mutácie	Pravdepodobnosť kríženia		
	0,4	0,5	0,6
0,01	90	96	95
0,02	97	96	98
0,03	99	97	97
0,05	99	99	99
0,06	100	99	100

Úspešnosť pre populáciu 500 chromozómov, max. počet generácií 100, štandardné kódovanie

Ani populácia 400 chromozómov nepriniesla vytúženú stovku. Až pri populácií 500 kusov sa stovka ukázala na dvoch miestach: Pri pravdepodobnosti mutácie 0,06 a pravdepodobnosti kríženia 0,4 a 0,6. Opäť bolo nutných aspoň 100 generácií.

Záver

S pomocou vytvoreného programu sa mi podarilo nájsť najlepšie parametre potrebné pre takmer stopercentnú istotu nájdenia minima funkcie. Ako najvýhodnejšie parametre pre pravdepodobnosti kríženia sa ukázali 0,4, 0,5 a 0,6, pričom ich zmena ovplyvňovala výsledky iba v obmedzenej miere.

Oveľa väčší dopad na získanie dobrých výsledkov mala voľba pravdepodobnosti mutácie. Jej najvýhodnejšia poloha pre grayovo kódovanie sa ukázala hodnota 0,03, pričom táto reprezentuje, že v každom mutovanom chromozóme, ktorý je tvorený 32 bitmi, sa zmení približne práve jeden bit. Pre štandardné kódovanie sa ako najlepšia ukázala hodnota 0,06, ktorá reprezentuje pravdepodobnosť pre zmenu približne dvoch bitov v jednom chromozóme.

Najväčší vplyv na výpočtovú zložitosť aj úspešnosť nájdenia minima mal parameter veľkosti populácie, ktorého ideálna hodnota sa líši pre prípady štandardného a grayovho kódovania. V tom prvom prípade je minimálna hodnota potrebná na dosiahnutie takmer stopercentnej pravdepodobnosti 500 kusov. Pri použití grayovho kódovania je táto hodnota polovičná a teda 250 kusov.

Počet potrebných generácií som pri štandardnom kódovaní stanovil na aspoň 100 a pri grayovom kódovaní na 40.

Ako je vidieť z výsledkov najdramatickejší vplyv na kvalitu genetického algoritmu mala voľba vnútornej binárnej reprezentácie reálneho čísla. V prípade použitia grayovho kódu bolo možné znížiť veľkosť populácie na polovicu a znížiť potrebný počet generácií dokonca na menej ako polovicu. Preto odporúčam používať grayov kód všade tam, kde sa reprezentujú reálne čísla pomocou binárnych reťazcov a kde na týchto reťazcoch prebieha náhodná bitová mutácia.

Použitá Literatúra

[1] V. Kvasnička, J.Pospíchal, P. Tiňo: *Evolučné algoritmy*, STU, Bratislava 2000.

[2] V. Kvasnička, J.Pospíchal: *Darwinovská evolúcia ako algoritmus*, STU, Bratislava 2003