Predmet: **Evolučné Algoritmy**

Vypracoval: **Jaroslav ONDRISKA**, FMFI UK, Bratislava, Informatika, Umelá inteligencia
E-mail: jaroslav.ondriska@st.fmph.uniba.sk

Téma 1.

Horolezecký algoritmus pre funkciu f(*x*) definovanú v

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

pre x $\in$ [-10,10]. Premenná *x* je binárne reprezentovaná tak v štandardnom kódovaní, ako aj v Grayovom kódovaní. Pokúste sa optimalizovať parametre c_0 a P_{mut} tak, aby ste skoro so 100% istotou dostali globálne minimum s čo najmenším počtom iteračných krokov.

Riešenie:

Teória

Teória optimalizačných problémov

Nasledujúci text je písaný na základe prednášok Evolučné algoritmy a knihy Evolučné algoritmy od Prof. Ing. Vladimíra Kvasničku, DrSc, Doc. RNDr. Jiřího Pospíchala, CSc, a Ing. Petra Tiňa, CSc. Takisto som využil elektronickú podobu týchto textov, ktoré sa dajú nájsť na internetovskej adrese: http://math.chtf.stuba.sk

Nech funkcia

$$f : D \rightarrow R$$

$$D = \prod_{i=1}^{n} [a_i,b_i]$$

zobrazuje n-rozmernú kocku D (karteziánsky súčin uzavretých intervalov $[a_i,b_i]$) na reálne číslo y$\in$R. Táto funkcia je ohraničená dvoma podmienkami:

(i) Existuje taký algoritmus, ktorý funkciu f vypočíta dostatočne rýchlo s požadovanou presnosťou pre každé x$\in$D (hovoríme, že funkcia f je dobre vypočítateľná).

(ii) Pre každú dvojicu lokálnych miním x_1,x_2 D je vzdialenosť $|x_1-x_2|$ väčšia ako dané kladné číslo $\delta>0$, $|x_1-x_2|>\delta$. Podmienka zhora ohraničuje počet lokálnych miním funkcie f, ktoré sa vyskytujú na kocke D. Nie je možné, aby sa v ľubovoľnom okolí minima funkcie vyskytovalo iné minimum, pre určité malé okolie minima funkcie by vyššie uvedená podmienka $|x_1-x_2|>\delta$ prestala platiť. Podmienka automaticky vylučuje z triedy prípustných funkcií tie funkcie, ktoré sú fraktálového typu, t.j. v každom okolí nejakého minima sa nachádza aspoň jedno iné minimum.

Globálne minimum funkcie f na kocke D je určené vzťahom

$$x_{opt} = \arg\min_{x \in D} f(x)$$

Nájdenie globálneho minima použitím klasických optimalizačných metód [1-3] (gradientových a negradientových) patrí medzi obtiažne numerické problémy pre funkcie, ktoré nie sú ohraničené ďalszími podmienkami (ako napr. že f(x) je konvexná funkcia na oblasti D). Z týchto dôvodov sa v súčasnosti pri riešení vyššie uvedeného problému často používajú tzv. evolučné optimalizačné algoritmy, ktoré poskytujú riešenia blízke globálnemu, alebo s ním totožné.

Binárna verzia funkcie f má tvar

$$f : \{0,1\}^k \rightarrow R$$

Táto funkcia je definovaná nad množinou binárnych vektorov dĺžky k, každému binárnemu vektoru zobrazenie f priradí reálne číslo z množiny R

$$y = f(\alpha)$$

Kardinalita množiny binárnych vektorov dĺžky k je určená vzťahom

$$|\{0,1\}^k| = 2^k$$

To znamená, že "dimenzia" priestoru vsźetkých možných binárnych vektorov dĺžky k rastie exponenciálne s dĺžkou k. Jednoduchá geometrická interpretácia binárnych vektorov je n-rozmerná hyperkocka.

Analóg predošlému optimalizačného problému pre binárne vektory má tvar

$$\alpha_{opt} = \arg\min_{\alpha \in \{0,1\}^{\kappa}} f(\alpha)$$

Vo vsźeobecnosti, globálne optimum α_{opt} sa nájde po preskúšaní všetkých možných binárnych vektorov dĺžky k. Obrazne povedané, problém vyriešime tak, že sa určitým systematickým spôsobom pohybujeme po vrcholoch k-rozmernej hyperkocky tak, že navštívime všetkých 2^k vrcholov. Algoritmicky tento prístup môže byť implementovaný pomocou metódy spätného prehľadávania, pozri algoritmus 2.1. CPU čas potrebný na riešenie tejto optimalizačnej úlohy je potom úmerný kardinalite priestoru riešení

$$t_{CPU} \propto 2^k$$

Binárna reprezentácia reálnej premennej

Nech sa binárny vektor α dĺžky k

$$\alpha = (\alpha_1\alpha_2 \ldots\alpha_\kappa) \in \{0,1\}^k$$

interpretujeme ako celé číslo

$$int(\alpha) = \sum_{i=1}^{k} \alpha_i 2^{k-i}$$

K tomuto celému číslu jednoduchým spôsobom priradíme reálne číslo, ktoré sa môže chápať ako aproximácia reálneho čísla $x \in [a,b]$

$$x \approx real(\alpha) = a + (b-a)\, int(\alpha) / (2^k - 1)$$

V našom prípade prevodová funkcia z binárneho čísla α na reálne číslo x vyzerá nasledovne

$$a = -10,\ b = 10$$
$$x = -10 + 20/(2^k - 1) int(\alpha)$$

Táto konštrukcia racionálneho čísla $a \le real(\alpha) \le b$ z binárneho reťazca α dĺžky k sa formálne interpretuje ako transformácia binárnej reprtezentácie na reálnu reprezentáciu, kde zostrojené recionálne číslo $real(\alpha)$ aproximuje požadované reálne číslo x s presnosťou $(b-a)/(2^k-1)$. Interval [a,b] obsahuje $m=2^k$ bodov $x_1=a$, $x_2=a+(b-a)/(2^k-1)$, ..., $x_i=a+(i-1)(b-a)/(2^k-1)$, ..., $x_n=b$, pozri Tab1.

Tabuľka 1

No.	α	int(α)	real(α')	α'
1	000	0	0	000
2	001	1	1/7	001
3	010	2	2/7	011
4	011	3	3/7	010
5	100	4	4/7	110
6	101	5	5/7	111
7	110	6	6/7	101
8	111	7	1	100

Inverzná transformácia k predošlému vzorcu má tvar

$$int(\alpha) = \lceil (x-a)*(2^k-1)/(b-a) \rceil$$

kde symbol $\lceil x \rceil$ označuje celú časť reálneho čísla x (napr. $\lceil 1.1 \rceil = 1$, $\lceil 1.9 \rceil = 1$). V dôsledku toho, že $a \le x \le b$, zlomok $(x-a)/(b-a)$ leží v uzavretom intervale [0,1], ak $int(\alpha) = 0$, potom x=a; ak $int(\alpha) = 2^k-1$, potom x=b.

Vyššie uvedená binárna reprezentácia má jednu podstatnú nevýhodu. Dvojica binárnych reťazcov, ktoré sa líšia vo všetkých polohách bitových premenných, môže odpovedať dvom susedným celým číslam (pozri napr. Tab.1, kde komplementárne binárne reťazce α_1=(011) a α_2=(100) sa interpretujú ako celé čísla int(α_1)=4, resp. int(α_2)=5). Táto nevýhoda štandardného binárneho kódu sa odstraňuje použitím tzv. Grayovho kódu. Jeho základná myšlienka spočíva v tom, že kóduje binárne čísla tak, že dve susedné čísla sú binárne reprezentované reťazcami, ktoré sú rôzne len v jednej polohe binárneho reťazca(pozri Tab1, kde napr. binárne reťazce α_1'=(010) a α_2'=(110), ktoré sa líšia len v prvej polohe reťazca, odpovedajú susedným celým číslam int(α_1')=4, resp. int(α_2')=5). Konštrukcia racionálneho čísla z intervalu [a,b](ktoré, ako už bolo povedané vyššie, aproximuje reálne číslo a≤x≤b) sa zakladá na tom, že binárne číslo v Grayovom kóde najprv pretransformujeme do štandardného kódu a až potom sa použije vzťah na aproximáciu čísla

$$x \approx real(\alpha) = a + (b-a)\, int(\alpha') / (2^k - 1)$$

kde α' = (α_1', α_2', ..., α_k') je binárny reťazec zostrojený pomocou Grayovho kódu a int(α') je odpovedajúce celé číslo priradené reťazcu α'

$$int(\alpha') = int(\alpha)$$

kde sa jednotlivé komponenty reťazca α = (α_1, α_2, ..., α_k) určia pomocou zložiek reťazca α' = (α_1', α_2', ..., α_k')

$$\alpha_1 = \alpha_1'$$
$$\alpha_2 = \alpha_1' + \alpha_2' = \alpha_1 + \alpha_2'$$
$$\alpha_3 = \alpha_1' + \alpha_2' + \alpha_3' = \alpha_2 + \alpha_3'$$
$$...$$
$$\alpha_k = \alpha_1' + \alpha_2' + ... + \alpha_k' = \alpha_{k-1} + \alpha_k'$$

kde binárna operácia "+" je definovaná ako "xor" súčet binárnych čísel, 0+0=0, 0+1=1, 1+0=1, 1+1=0. Tieto vťahy možno jednoducho verifikovať na príkladoch z tabuľky Tab1. Posledný stĺpec tejto tabuľky (Grayov kód) sa použitím vzťahov pretransformuje na druhý stĺpec (štandardný kód). Inverzná transformácia (t.j. konštrukcia Grayovho kódu zo štandardného kódu) sa jednoducho určí vzťahmi

$$\alpha_1' = \alpha_1$$
$$\alpha_2' = \alpha_1 + \alpha_2$$
$$\alpha_3' = \alpha_2 + \alpha_3$$
$$...$$
$$\alpha_k' = \alpha_{k-1} + \alpha_k$$

Transformácia spojitého optimalizačného problému na binárny optimalizačný problém

Našou hlavnou úlohou je riešiť spojitý optimalizačný problém pre globálne minimum funkcie f(x) nad oblasťou D. Pre ďalšie úvahy o použití evolučných algoritmov na riešenie tohto problému bude užitočné pretransformovať tento spojitý optimalizačný problém na binárny optimalizačný problém. Predpokladajme, že každá z n premenných vektora x∈D je vyjadrená v binárnej reprezentácii bitovým vektorom dĺžky k. To znamená, že vektor x∈D je v binárnej reprezentácii vyjadrený bitovým vektorom dĺžky kn . Nech funkcia f vyhovuje druhej podmienke z úvodného odseku pre danú kladnú konštantu δ. Budeme predpokladať, že dĺžka binárnej reprezentácie k je zvolená tak, že platí

$$\delta >> \frac{(b_i - a_i)}{(2^k - 1)} \quad \text{pre i = 1,2,...,n}$$

Táto podmienka vyžaduje, aby minimálna vzdialenosť medzi dvoma minimami funkcie f(x) nad oblasťou D bola omnoho väčšia ako presnosť binárnej reprezentácie pre každú premennú vektora x∈D.

Prechod od binárneho vektora α = (α_1, α_2, ..., α_{kn}) ∈{0,1}^{kn} k spojitému vektoru x = (x_1, x_2, ..., x_n) ∈D sa môže formálne chápať ako transformácia

$$\Gamma=\{0,1\}^{kn} \rightarrow D$$

$$x = \Gamma(\alpha)$$

ktorá zobrazuje množinu binárnych vektorov dĺžky kn na body – n-tice reálnych čísel z kocky D. Ináč povedané, konečná množina (2^{kn}) binárnych vektorov dĺžky kn je reprezentovaná pomocou zobrazenia Γ bodmi, ktoré môžu byť v oblasti D usporiadané do ortogonálnej mriežky. Spojitý minimalizačný problém pri použití binárnej reprezentácie n-rozmerných vektorov x sa teda realizuje na konečnej množine diskrétnych bodov. Označme α_{opt}' binárny vektor dĺžky kn, ktorý bol získaný riešením binárneho optimalizačného problému, pričom funkcia f tohto problému je totožná s funkciou f v spojitom optimalizačnom probléme

$$\alpha_{opt}' = \underset{\alpha \in \{0,1\}^{kn}}{arg\ min}\ f(\Gamma(\alpha))$$

Vektor reálnych premenných priradený tomuto binárnemu vektoru je $x_{opt}' = \Gamma(\alpha_{opt}')$. Budeme predpokladať, že presnosť binárnej reprezentácie (t.j počet bitov k rezervovaných pre každú reálnu premennú) je taká, že vektor x_{opt}' je blízky presnému riešeniu x_{opt} spojitého optimalizačného problému funkcie f nad oblasťou D.

$$x_{opt} \approx x_{opt}' = \Gamma(\alpha_{opt}')$$

Presnosť riešenia spojitého optimalizačného problému pri prechode zo spojitej reprezentácie do binárnej reprezentácie závisí od konštanty k, ktorá určuje dĺžku binárnych vektorov reprezentujúcich jednotlivé reálne premenné. Ak funkcia f obsahuje málo miním, ktoré sú dostatočne navzájom izolované (konštanta δ je veľká) a široké, konštanta k nemusí byť veľká. Avšak ak funkcia f obsahuje množstvo miním, ktoré ležia blízko seba (konštanta δ musí byť malá), potom konštanta k musí byť pomerne veľká. Ortogonálna mriežka bodov nad oblasťou D, ktoré sú generované zvolenou binárnou reprezentáciou reálnych premenných, musí byť dostatočne jemná, aby sa pokryli a odlíšili blízko seba ležiace minimá funkcie f.

V našom prípade máme len jednorozmerný vektor, teda reálne číslo, preto nemusíme uvažovať takto všeobecne a stačí rozdeliť reálnu os na primerané dieliky.

Horolezecké algoritmy

Uvedieme dva základné typy stochastických optimalizačných algoritmov, ktoré aj keď neobsahujú evolučné rysy, budú slúžiť ako základ pre formuláciu evolučných optimalizačných algoritmov. Slepý algoritmus je základný stochastický algoritmus, ktorý opakovane generuje náhodne riešenie z oblasti D a zapamätá si ho len vtedy, ak bolo získané lepšie riešenia ako to, ktoré už bolo zaznamenané v predchádzajúcej histórii algoritmu. Z dôvodov kompatibility tohto algoritmu s evolučnými algoritmami uvedieme jeho implementáciu pre binárnu reprezentáciu vektorov - riešení, pozri algoritmus 1.

Algoritmus 1

```
procedure Blind_Algoritmus(input: t_max,k,n; output: a_fin,f_fin);
begin
        f_fin:=∞;
        t:=0;
        while t< t_max do
        begin
                t:=t+1;
                α:=randomly generated binary vector of the length kn;
                if f(Γ(a))<f_fin then
                begin
                        α_fin:=α;
                        f_fin:=f(Γ(α))
                end;
        end;
end;
```

Jednoduchými úvahami dá sa dokázať, že tento jednoduchý stochastický optimalizačný algoritmus poskytuje korektné globálne minimum optimalizačného problému realizovaného nad ortogonálnou mriežkou bodov z oblasti D za predpokladu, že parameter procedúry t_{max} asymptoticky rastie do nekonečna

$$\lim_{t_{max} \to \infty} P(t_{max} \mid \alpha_{fin} = \alpha_{opt}) = 1$$

kde $P(t_{max} \mid \alpha_{fin} = \alpha_{opt})$ je pravdepodobnosť toho, že slepý algoritmus po t_{max} iteračných krokoch poskytne výstupné riešenie, ktoré je totožné s presným riešením (globálne minimum).

Vo všeobecnosti môžeme povedať, že slepý algoritmus neobsahuje žiadnu stratégiu konštrukcie riešení (t.j. binárnych vektorov dĺžky kn) na základe predchádzajúcej histórie algoritmu. Každé riešenie je zostrojené úplne nezávisle (t.j. plne náhodne) od predchádzajúcich riešení. Zaznamenáva sa to riešenie, ktoré v priebehu aktivácie procedúry poskytuje zatiaľ najnižšiu funkčnú hodnotu. Po ukončení aktivácie procedúry je toto riešenie výstupným parametrom.

Slepý algoritmus môže byť jednoducho zovšeobecnený na tzv. horolezecký algoritmus (hill climbing), kde sa iteračne hľadá najlepšie lokálne riešenie v určitom okolí, a toto riešenie je v ďalšom kroku použité ako "stred" novej oblasti. K formalizácii horolezeckého algoritmu zavedieme niektoré základné pojmy, ktoré sú dôležité pre jeho jednoduchý popis. Operácia mutácie stochasticky transformuje binárny vektor α na nový binárny vektor α' , pričom stochastičnosť toho procesu je určená pravdepodobnosťou P_{mut}

$$\alpha' = O_{mut}(\alpha)$$

kde α a α' sú dva binárne vektory rovnakej dĺžky kn

$$\alpha = (\alpha_1, \alpha_2, ..., \alpha_{kn}) \text{ a } \alpha' = (\alpha_1', \alpha_2', ..., \alpha_{kn}')$$

kde jednotlivé komponenty α' sú určené takto

$$\alpha_i' = \begin{cases} 1-\alpha_i \text{ (pre random } < P_{mut}) \\ \alpha_i \text{ (ostatné prípady)} \end{cases}$$

kde random je náhodné číslo z intervalu [0,1) generované s rovnomernou distribúciou (pozri algoritmus 1.3). Pravdepodobnosť P_{mut} určuje stochastičnosť operátora mutácie, v limitnom prípade, ak $P_{mut} \to 0$, potom operátor O_{mut} nemení binárny vektor

$$\lim_{P_{mut} \to 0} O_{mut}(\alpha) = \alpha$$

Štúdie ukázali, že mutácia pre binárne vektory so štandardným kódovaním vytvára nové binárne vektory s číselnými hodnotami, ktoré sú rozdelené diskrétnym spôsobom okolo nulovej hodnoty. Ak sa použije Grayovo kódovanie, číselné hodnoty nových binárnych vektorov vytvárané mutáciou sú rozdelené pomerne "spojito" okolo nulovej hodnoty približne s Gaussovou distribúciou pravdepodobnosti. Môžeme teda povedať, že mutácia v rámci štandardného kódovania poskytuje nové binárne vektory, ktoré vzhľadom k pôvodným (t.j. nemutovaným) binárnym vektorom majú číselné hodnoty, ktoré sú relatívne diskrétne rozdelené okolo nuly. Táto "diskrétnosť rozdelenia" je potlačená, ak sa použije Grayov kód pri binárnej reprezentácii reálnych čísel. Na základe tohto výsledku možno konštatovať, že Grayov kód je asi vhodnejší pre binárnu reprezentáciu reálnych premenných, mutáciou vytvorené binárne vektory sa vyskytujú "spojito" na celej oblasti prípustných hodnôt.

Základná idea horolezeckého algoritmu spočíva v tom, že vzhľadom k určitému zvolenému riešeniu zostrojíme náhodne predpísaný počet nových riešení tak, že vo zvolenom riešení sa náhodne zmenia bitové premenné (hovoríme, že zvolené riešenie je stred oblasti z neho náhodne generovaných riešení). Z tejto oblasti vyberieme najlepšie riešenie (t.j. s minimálnou funkčnou hodnotou nad bodmi z daného okolia), ktoré sa použije v nasledujúcom iteračnom kroku ako stred novej oblasti. Tento proces sa opakuje predpísaný počet-krát, pričom sa zaznamenáva najlepšie riešenie, ktoré sa vyskytlo v priebehu histórie algoritmu. (Možná modifikácia tohto algoritmu spočíva v prehľadávaní všetkých riešení, ktoré sa líšia v jedinom bite od aktuálneho riešenia.)

Algoritmus 2

```
procedure Mutation_Bin(input: α; output α');
begin
        for i:=1 to kn do
                if random<P_mut then α_i' := 1-α_i
                                else α_i' := α_i;
end;
```

Okolie $U(\alpha)$ binárneho vektora α sa zostrojí pomocou vektorov $\alpha' = O_{mut}(\alpha)$

$$U(\alpha) = \{\alpha' = O_{mut}(\alpha)\}$$

pričom budeme predpokladať, že kardinalita (počet elementov) sa rovná predpísanej hodnote, $|U(\alpha)| = c_0$, kde c_0 je dané kladné celé číslo. Poznamenajme, že v dôsledku stochastičnosti aplikácie operácie mutácie na daný binárny vektor α má zloženie okolia $U(\alpha)$ tiež stochastický charakter. To, či nejaký vektor α' patrí alebo nepatrí do okolia $U(\alpha)$, je určené len pravdepodobnostne, a nie deterministicky. Najlepšie riešenie v okolí $U(\alpha)$ je určené takto

$$\alpha^* = \arg\min_{\alpha' \in U(\alpha)} f(\Gamma(\alpha'))$$

V horolezeckom algoritme sa takto získané riešenie α^* použije ako stred v ďalšom iteračnom kroku algoritmu. Implementácia horolezeckého algoritmu v pseudopascale je v algoritme 2.

Analógia vzorca zo slepého algoritmu, ktorá hovorí, že tento jednoduchý algoritmus je asymptoticky schopný nájsť globálne minimum, platí aj v horolezeckom algoritme. V tomto prípade sa ale kardinalita okolia $c_0 = |U(\alpha)|$ musí asymptoticky zväčšovať do nekonečna

$$\lim_{c_0 \to \infty} P(c_0 \mid \alpha_{fin} = \alpha_{opt}) = 1$$

Potom je ale zbytočné opakovať iteračné kroky horolezeckého algoritmu pre nové lokálne optimálne riešenia, už v rámci jedného iteračného kroku získame pre $c_0 \to \infty$ globálne riešenie.
Ako naznačujú jednoduché numerické aplikácie, horolezecký algoritmus, aj keď neobsahuje explicitne evolučnú stratégiu, je pomerne efektívny a robustný stochastický optimalizačný algoritmus, ktorý je schopný pre jednoduchšie úlohy nájsť globálne minimum.

Algoritmus 3

```
procedure Hill_Climbing(input:t_max,c_0,P_mut; output: f_fin, α_fin);
begin
        α:=randomly generated binary vector of the length kn;
        f_fin:=A_0; t:=0;
        while t<t_max do
        begin
                t:=t+1;
                α* := arg min f(Γ(α'));
                          α'∈U(α)
                if  f (Γ(α*))<f_fin then
                begin
                        f_fin := f(Γ(α));
                        α_fin := α*;
                end;
                α := α*;
        end;
end;
```

Program

Program je priložený k tejto case-study. Je naprogramovaný v Borland C++ Builderi 5.0. Po spustení programu sa ukáže okno, kde užívateľ môže zadať vstupné parametre a spustiť výpočet. Medzi voliteľné parametre patria : počet krokov, ako dlho sa výpočet bude vykonávať (t_{max}), dĺžka bitového reťazca reprezentujúceho reálne číslo k, kardinalita okolia $U(\alpha)$ (c_0), pravdepodobnosť mutácie (P_{mut}) a výber medzi štandardným a Grayovým kódovaním čísiel. Ďalej je tam ešte ShowEvery textbox, kde si užívateľ môže nastaviť, po koľko krokoch sa majú výsledky vypisovať do textového okienka. Po stlačení tlačítka Start sa nakreslí graf funkcie zo zadania čiernou farbou a osi šedou farbou. Počas výpočtu užívateľ môže sledovať priebežnú najlepšiu hodnotu $\Gamma(\alpha)$, ktorá je znázornená zelenou farbou. Popri tom sa do textového okienka vypisujú hodnoty $\Gamma(\alpha)$ a $f(\Gamma(\alpha))$ zatiaľ najlepšieho binárneho vektora α. Tlačítko Clear Results po stlačení vyčistí obsah tohto textového okna.

Pozn. V programe som nepočítal okolie vektora ako množinu(tak, ako to je v teórii) ale ako multimnožinu (kvôli zjednodušeniu a zefektívneniu programu).

Výsledky

Keďže minimálna vzdialenosť medzi dvoma minimami funkcie f(x) nad oblasťou D má byť omnoho väčšia ako presnosť binárnej reprezentácie pre každú premennú vektora $x \in D$ a funkcia f má relatívne veľa lokálnych miním na intervale [-10,10], volil som k aspoň 10. S väčším k narastala presnosť výsledku. Znázorňuje to nasledovná tabuľka

k	$\Gamma(\alpha)=x$	$f(\Gamma(\alpha))$
10	-0,791788856304985	0,00342532030583883
11	-0,786516853932584	0,000120208322914431
12	-0,783882783882784	0,00016425076425608
13	-0,785007935539006	7,06711692212111E-6
14	-0,785570408350119	5,85470026297392E-6
15	-0,785241248817408	3,04793813559118E-7
16	-0,785381857022965	5,14703725131406E-7
17	-0,785299570461811	7,07103260872474E-10

V tabuľke je dobre vidno ako presnosť narastá s podrobnejšou reprezentáciou reálneho čísla x. Dá sa predpokladať, že globálnym minimom funkcie f je 0. Rozhodol som sa reprezentovať x binárnym vektorom dĺžky 11, nakoľko väčšia dĺžka vektora už na presnosti až tak nepridá. Prehľad výsledkov zobrazujú nasledovné tabuľky. Počet významných číslic je 4-5, ale pre úplnosť uvádzam presné čísla, ako ich počítal program.

Tabuľka 1 pre k = 11, $t_{max} = 100$, $c_0 = 50$, $P_{mut} = 0.1$, code = normal

Iterácia	α	$\Gamma(\alpha)=x$	$f(\Gamma(\alpha)) = f(x)$
5	10011110001	2,35955056179775	0,048877851151881
10	10011110001	2,35955056179775	0,048877851151881
15	10011110001	2,35955056179775	0,048877851151881
20	10011110001	2,35955056179775	0,048877851151881
25	10011110001	2,35955056179775	0,048877851151881
30	10011110001	2,35955056179775	0,048877851151881
35	10011110001	2,35955056179775	0,048877851151881
40	10011110001	2,35955056179775	0,048877851151881
45	01110101111	-0,786516853932584	0,000120208322914431
50	01110101111	-0,786516853932584	0,000120208322914431
...	...	...	...
100	01110101111	-0,786516853932584	0,000120208322914431

Tabuľka 2 pre k = 11, $t_{max} = 100$, $c_0 = 50$, $P_{mut} = 0.3$, code = normal

Iterácia	α	$\Gamma(\alpha)=x$	$f(\Gamma(\alpha)) = f(x)$
5	10011110000	2,34978016609673	0,0507582784535198
10	01110101101	-0,806057645334636	0,0347054958499099
15	01110101101	-0,806057645334636	0,0347054958499099
20	01110101101	-0,806057645334636	0,0347054958499099
25	01110101101	-0,806057645334636	0,0347054958499099
30	01110101101	-0,806057645334636	0,0347054958499099
35	01110101111	-0,786516853932584	0,000120208322914431
40	01110101111	-0,786516853932584	0,000120208322914431
45	01110101111	-0,786516853932584	0,000120208322914431
50	01110101111	-0,786516853932584	0,000120208322914431
...	...	...	...
100	01110101111	-0,786516853932584	0,000120208322914431

Tabuľka 3 pre k = 11, $t_{max} = 100$, $c_0 = 100$, $P_{mut} = 0.1$, code = normal

Iterácia	α	$\Gamma(\alpha)=x$	$f(\Gamma(\alpha)) = f(x)$
5	01001101110	-3,9228138739619	0,137706369603999
10	10011110010	2,36932095749878	0,0617378109078603
15	10011110010	2,36932095749878	0,0617378109078603
20	10011110010	2,36932095749878	0,0617378109078603
25	10011110010	2,36932095749878	0,0617378109078603
30	10011110010	2,36932095749878	0,0617378109078603

35	10011110010	2,36932095749878	0,0617378109078603
40	10011110010	2,36932095749878	0,0617378109078603
45	10011110010	2,36932095749878	0,0617378109078603
50	10011110010	2,36932095749878	0,0617378109078603
...			...
100	01110101111	-0,786516853932584	0,000120208322914431

Tabuľka 4 pre k = 11, t_{max} = 100, c_0 = 100, P_{mut} = 0.3, code = normal

Iterácia	α	$\Gamma(\alpha)=x$	f ($\Gamma(\alpha)$) = f(x)
5	10011101111	2,3400097703957	0,0673288415006917
10	10011101111	2,3400097703957	0,0673288415006917
15	01110110000	-0,776746458231558	0,00595518960102675
20	01110110000	-0,776746458231558	0,00595518960102675
25	01110110000	-0,776746458231558	0,00595518960102675
30	01110110000	-0,776746458231558	0,00595518960102675
35	01110110000	-0,776746458231558	0,00595518960102675
40	01110110000	-0,776746458231558	0,00595518960102675
45	01110110000	-0,776746458231558	0,00595518960102675
50	01110101111	-0,786516853932584	0,000120208322914431
...	...	...	...
100	01110101111	-0,786516853932584	0,000120208322914431

Tabuľka 5 pre k = 11, t_{max} = 100, c_0 = 50, P_{mut} = 0.1, code = Gray

Iterácia	α	$\Gamma(\alpha)=x$	f ($\Gamma(\alpha)$) = f(x)
5	01101011001	-3,9228138739619	0,137706369603999
10	11010001011	2,36932095749878	0,0617378109078603
15	11010001011	2,36932095749878	0,0617378109078603
20	11010001011	2,36932095749878	0,0617378109078603
25	11010001011	2,36932095749878	0,0617378109078603
30	01001111000	-0,786516853932584	0,000120208322914431
35	01001111000	-0,786516853932584	0,000120208322914431
40	01001111000	-0,786516853932584	0,000120208322914431
45	01001111000	-0,786516853932584	0,000120208322914431
50	01001111000	-0,786516853932584	0,000120208322914431
...	...	...	...
100	01001111000	-0,786516853932584	0,000120208322914431

Tabuľka 6 pre k = 11, t_{max} = 100, c_0 = 50, P_{mut} = 0.3, code = Gray

Iterácia	α	$\Gamma(\alpha)=x$	f ($\Gamma(\alpha)$) = f(x)
5	11010001110	2,38886174890083	0,129646646133414
10	11010001110	2,38886174890083	0,129646646133414
15	01000111011	-0,444553004396678	0,112803749873346
20	01000111011	-0,444553004396678	0,112803749873346
25	01001101001	-0,766976062530533	0,0271323835719571
30	01001111001	-0,79628724963361	0,00980285067762631
35	01001111001	-0,79628724963361	0,00980285067762631
40	01001101000	-0,776746458231558	0,00595518960102675
45	01001101000	-0,776746458231558	0,00595518960102675
50	01001101000	-0,776746458231558	0,00595518960102675
...	...	...	...
100	01001111000	-0,786516853932584	0,000120208322914431

Tabuľka 7 pre k = 11, t_{max} = 100, c_0 = 100, P_{mut} = 0.1, code = Gray

Iterácia	α	$\Gamma(\alpha)=x$	f ($\Gamma(\alpha)$) = f(x)
5	01001111100	-0,854909623839766	0,346336435450663
10	01001101011	-0,757205666829507	0,0630086193585556
15	01001101001	-0,766976062530533	0,0271323835719571
20	01001101000	-0,776746458231558	0,00595518960102675
25	01001101000	-0,776746458231558	0,00595518960102675
30	01001111000	-0,786516853932584	0,000120208322914431
35	01001111000	-0,786516853932584	0,000120208322914431
40	01001111000	-0,786516853932584	0,000120208322914431

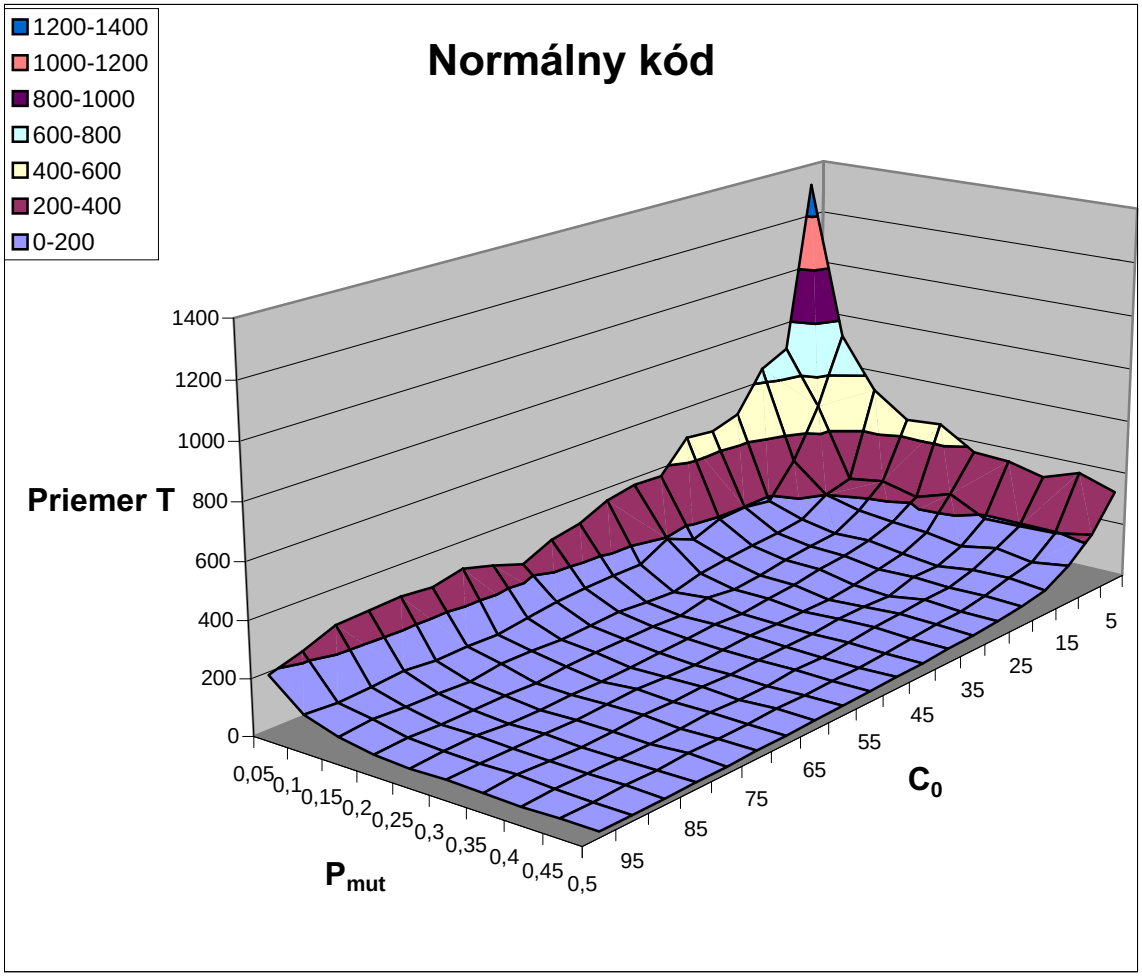
45	01001111000	-0,786516853932584	0,000120208322914431
50	01001111000	-0,786516853932584	0,000120208322914431
...	...	...	...
100	01001111000	-0,786516853932584	0,000120208322914431

Tabuľka 8 pre k = 11, t_{max} = 100, c_0 = 100, P_{mut} = 0.3, code = Gray

Iterácia	α	Γ(α)=x	f (Γ(α)) = f(x)
5	01000111010	-0,434782608695652	0,113686894262512
10	01001101000	-0,776746458231558	0,00595518960102675
15	01001111000	-0,786516853932584	0,000120208322914431
20	01001111000	-0,786516853932584	0,000120208322914431
25	01001111000	-0,786516853932584	0,000120208322914431
30	01001111000	-0,786516853932584	0,000120208322914431
35	01001111000	-0,786516853932584	0,000120208322914431
40	01001111000	-0,786516853932584	0,000120208322914431
45	01001111000	-0,786516853932584	0,000120208322914431
50	01001111000	-0,786516853932584	0,000120208322914431
...	...	...	...
100	01001111000	-0,786516853932584	0,000120208322914431

V nasledujúcich grafoch a tabuľkách je zobrazený priemerný počet krorkov, na priemerne koľko krokov sa algoritmus dostal do globálneho minima funkcie f. To znamená, že keď chceme, aby sa tam dostal s istotou, musíme pošet krokov T_{max} nastaviť o niečo viac. Priemerne znamená toľko, že táto hodnota bola vypočítaná zo 100 pokusov pri daných nastaveniach C_0 a P_{mut} a potom zpriemerovaná.

Graf 1

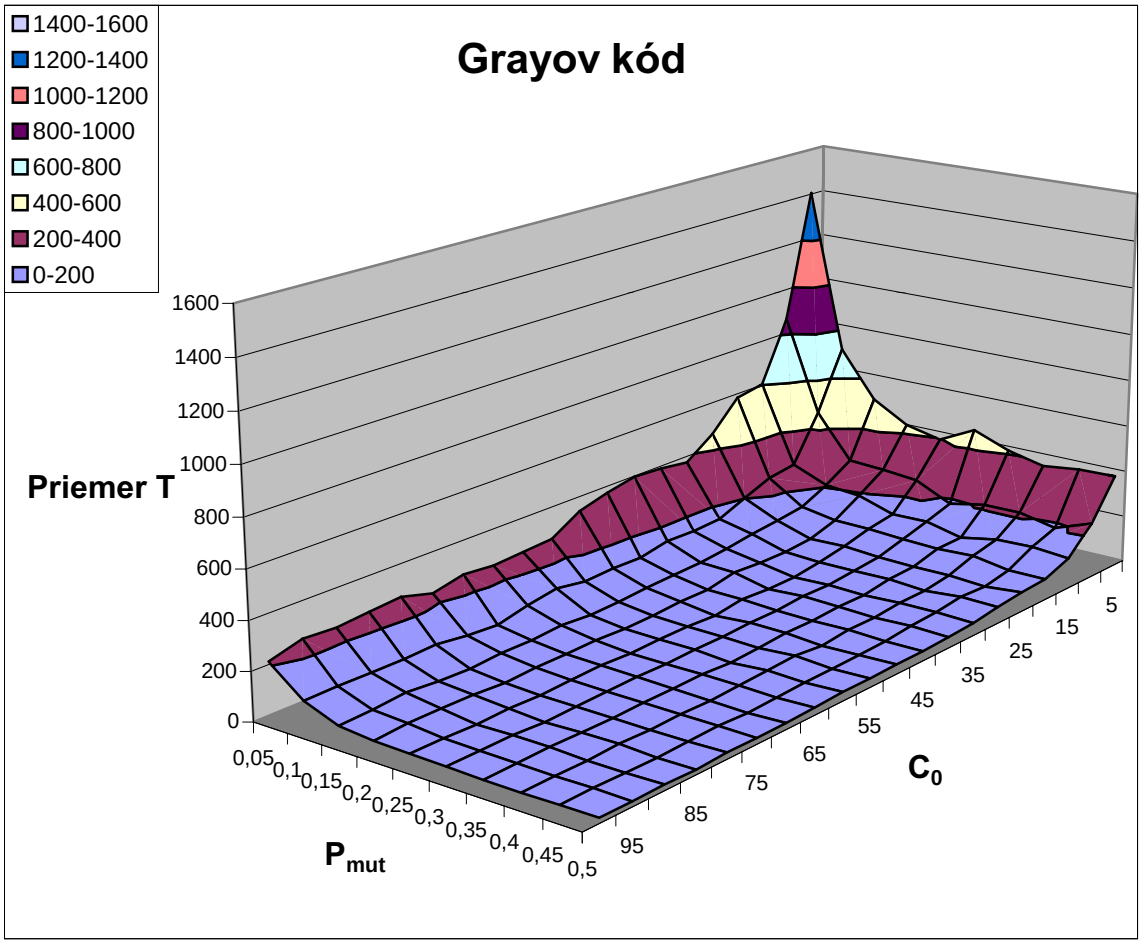


Tabuľka 9 Normálny kód – hodnoty z grafu

C_0/P_{mut}	0,05	0,1	0,15	0,2	0,25	0,3	0,35	0,4	0,45	0,5
5	1319	740	549	456	467	384	376	344	390	346
10	696	491	226	246	214	256	189	192	196	221
15	648	307	200	169	154	144	121	146	124	143
20	502	183	141	116	121	101	85	103	106	92

25	470	198	127	107	102	82	85	78	80	75
30	481	192	118	94	92	64	55	76	65	64
35	368	152	107	77	73	51	58	66	62	55
40	373	195	94	66	64	50	57	61	53	52
45	352	138	65	77	51	47	45	49	45	45
50	304	137	70	54	56	49	43	39	39	43
55	282	133	78	57	47	42	40	42	38	39
60	232	125	71	55	41	33	35	30	34	36
65	270	109	64	53	34	36	39	29	37	33
70	300	93	69	48	30	41	33	34	31	29
75	275	95	68	35	31	34	35	26	29	29
80	285	90	51	38	25	28	27	22	31	24
85	279	100	57	35	25	30	23	23	24	24
90	273	90	57	34	35	29	26	24	26	21
95	226	83	64	38	31	29	32	25	21	21
100	190	90	52	31	23	26	22	18	22	21

Graf 2



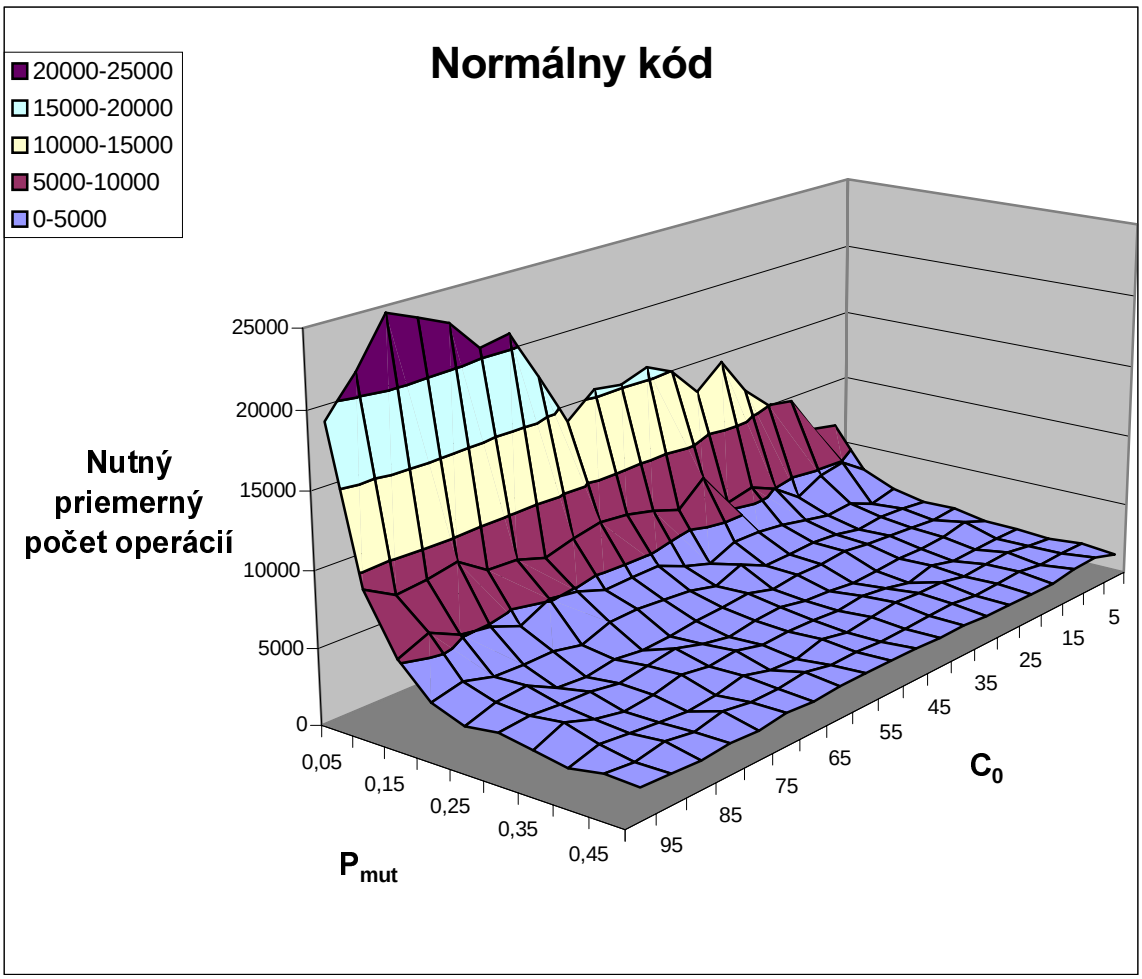
Tabuľka 10 Grayov kód – hodnoty z grafu

C_0/P_{mut}	**0,05**	**0,1**	**0,15**	**0,2**	**0,25**	**0,3**	**0,35**	**0,4**	**0,45**	**0,5**
5	1404	717	516	430	396	472	413	378	396	401
10	864	463	275	266	253	181	212	213	181	237
15	603	265	205	193	143	147	111	140	145	132
20	582	248	151	126	124	115	109	116	91	90
25	459	201	135	89	94	83	78	79	69	83
30	373	153	121	78	71	71	72	61	75	70
35	389	145	103	78	53	59	53	59	60	54
40	397	148	91	85	51	60	54	48	53	47
45	371	123	85	59	57	49	45	48	41	41
50	338	125	79	58	38	48	46	34	36	40
55	264	118	69	49	43	41	36	46	34	37

60	255	137	63	46	42	41	28	39	32	37
65	246	116	72	49	36	35	35	32	27	31
70	258	85	61	46	41	36	32	26	21	26
75	229	93	61	44	35	32	30	29	27	25
80	264	109	51	36	29	28	26	23	23	27
85	254	102	54	44	32	22	26	28	19	20
90	242	101	64	44	25	29	28	20	21	23
95	249	103	52	43	29	26	24	21	23	21
100	212	99	45	32	27	24	20	18	21	21

Ďalšie dva grafy ukazujú, koľko priemerne operácií (počítaní funkcie f) algoritmus potreboval na to, aby sa dostal do globálneho minima. Znova "priemerne" znamená, že hodnota bola vypočítaná zo 100 pokusov. Celé skúmanie som robil pri k=11, takže zmysel použiť tento algoritmus má, keď tieto hodnoty budú pod 2048.

Graf 3

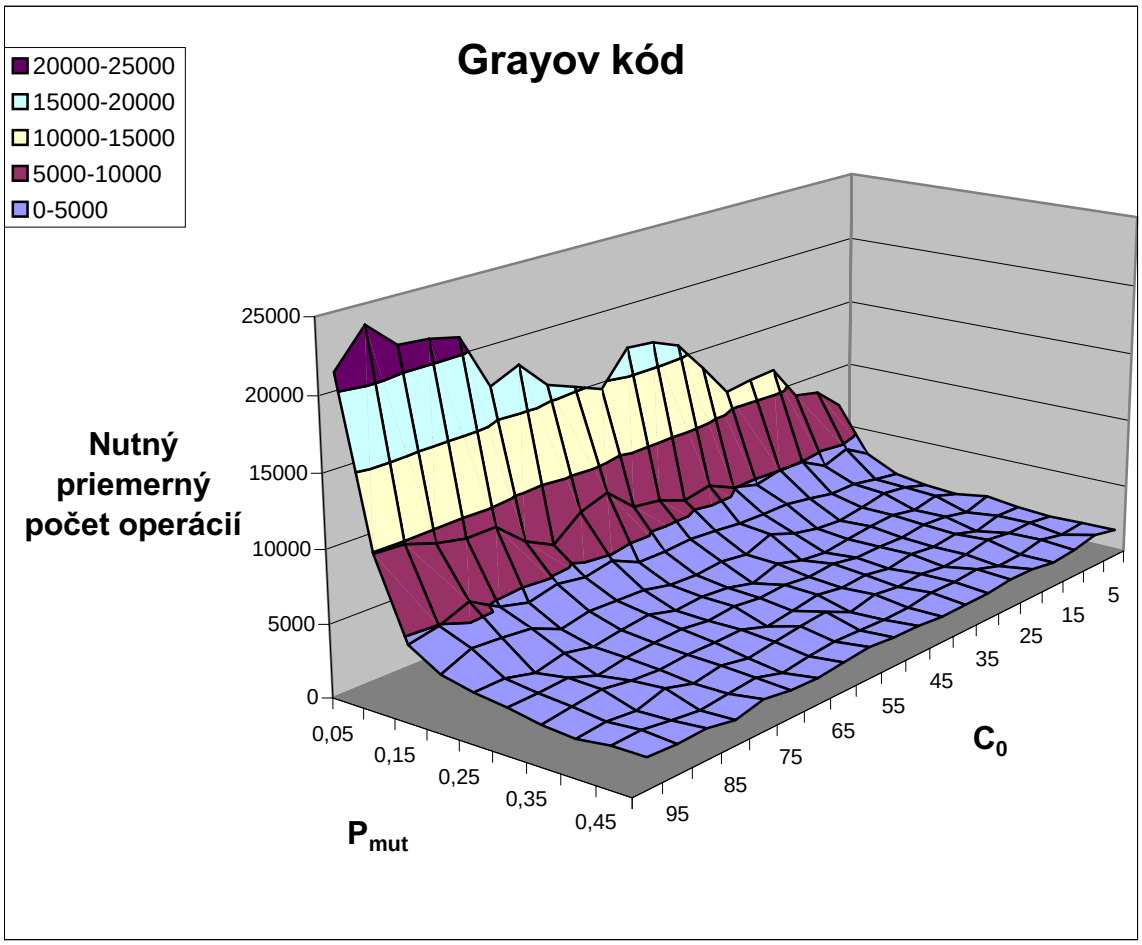


Tabuľka 11

C_0/P_{mut}	**0,05**	**0,1**	**0,15**	**0,2**	**0,25**	**0,3**	**0,35**	**0,4**	**0,45**	**0,5**
5	6595	3700	2745	2280	2335	1920	1880	1720	1950	1730
10	6960	4910	2260	2460	2140	2560	1890	1920	1960	2210
15	9720	4605	3000	2535	2310	2160	1815	2190	1860	2145
20	10040	3660	2820	2320	2420	2020	1700	2060	2120	1840
25	11750	4950	3175	2675	2550	2050	2125	1950	2000	1875
30	14430	5760	3540	2820	2760	1920	1650	2280	1950	1920
35	12880	5320	3745	2695	2555	1785	2030	2310	2170	1925
40	14920	7800	3760	2640	2560	2000	2280	2440	2120	2080
45	15840	6210	2925	3465	2295	2115	2025	2205	2025	2025
50	15200	6850	3500	2700	2800	2450	2150	1950	1950	2150
55	15510	7315	4290	3135	2585	2310	2200	2310	2090	2145
60	13920	7500	4260	3300	2460	1980	2100	1800	2040	2160

65	17550	7085	4160	3445	2210	2340	2535	1885	2405	2145
70	21000	6510	4830	3360	2100	2870	2310	2380	2170	2030
75	20625	7125	5100	2625	2325	2550	2625	1950	2175	2175
80	22800	7200	4080	3040	2000	2240	2160	1760	2480	1920
85	23715	8500	4845	2975	2125	2550	1955	1955	2040	2040
90	24570	8100	5130	3060	3150	2610	2340	2160	2340	1890
95	21470	7885	6080	3610	2945	2755	3040	2375	1995	1995
100	19000	9000	5200	3100	2300	2600	2200	1800	2200	2100

Graf 4



Tabuľka 12

C_0/P_{mut}	**0,05**	**0,1**	**0,15**	**0,2**	**0,25**	**0,3**	**0,35**	**0,4**	**0,45**	**0,5**
5	7020	3585	2500	2150	1980	2360	2065	1890	1980	2005
10	8640	4630	2750	2660	2530	1810	2120	2130	1810	2370
15	9045	3975	3075	2895	2145	2205	1665	2100	2175	1980
20	11640	4960	3020	2520	2480	2300	2180	2320	1820	1800
25	11475	5025	3375	2225	2350	2075	1950	1975	1725	2075
30	11190	4590	3630	2340	2130	2130	2160	1830	2250	2100
35	13615	5075	3605	2730	1855	2065	1855	2065	2100	1890
40	15880	5920	3640	3400	2040	2400	2160	1920	2120	1880
45	16695	5535	3825	2655	2565	2205	2025	2160	1845	1845
50	16900	6250	3950	2900	1900	2400	2300	1700	1800	2000
55	14520	6490	3795	2695	2365	2255	1980	2530	1870	2035
60	15300	8220	3780	2760	2520	2460	1680	2340	1920	2220
65	16055	7540	4680	3185	2340	2275	2275	2080	1755	2015
70	18060	5950	4270	3220	2870	2520	2240	1820	1470	1820
75	17175	6975	4575	3300	2625	2400	2250	2175	2025	1875
80	21120	8720	4080	2880	2320	2240	2080	1840	1840	2160
85	21590	8670	4590	3740	2720	1870	2210	2380	1615	1700
90	21780	9090	5760	3960	2250	2610	2520	1800	1890	2070

95	23655	9785	4940	4085	2755	2470	2280	1995	2185	1995
100	21200	9900	4500	3200	2700	2400	2000	1800	2100	2100

Čo vyplýva z tabuliek a grafov

Ako už vyplývalo z teórie, je lepšie na spoľahlivosť mať okolie s čo najväčšou mohutnosťou. Na druhej strane zvyšuje sa tým výpočtová náročnosť, čo v praxi znamená, že to nesmieme preháňať. Preto si myslím, že 50 úplne postačuje, stačilo by však aj menšie číslo. Čo sa týka pravdepodobnosti mutácie, ukázalo sa, že je výhodné ju nastaviť na vyššie čislo okolo 0,35, pri Grayovom kódovaní dokonca 0,45. Algoritmus vtedy konvergoval rýchlejšie. Kódovanie bolo pri testovaní lepšie Grayovo, no ako sa ukázalo z tabuliek, celkovo je o niečo málo víťaz normálne kódovanie, pretože malo stabilnejšie výsledky pod 2048. Minimálny počet operácií z celkového hľadiska dosiahlo Grayovo kódovanie, takže ťažko rozhodnúť, ktoré je lepšie. Určite proti Grayovmu kódovaniu hraje vyššia časová náročnosť pri dekódovaní binárneho reťazca, nakoľko efektivita nie je o moc lepšia.
Najlepšie výsledky z hľadiska efektivity som vyznačil modrou farbou, prikláňam sa však k červeným. Teda podľa mňa je najlepšie nastaviť parametre tak, ako v záverečnom odporúčaní.

Pozn.: Podľa výsledkov algoritmus nepracuje veľmi efektívne, pri malej dimenzii vektora α. Počet operácií je v najlepšom veľmi blízky počtu operáciám vyčerpávajúceho prehľadávania. Algoritmus začína byť efektívny až pri väčších dimenziách vektora α.

Záverečné odporúčanie
Použiť normálne kódovanie
Mohutnosť okolia nastaviť na 20
Dimenzia binárneho vektora 11
Pravdepodobnosť mutácie 0,35
Počet krokov okolo 100

alebo

Použiť Grayovo kódovanie
Mohutnosť okolia nastaviť na 50 - 70
Dimenzia binárneho vektora 11
Pravdepodobnosť mutácie 0,45
Počet krokov 50-30