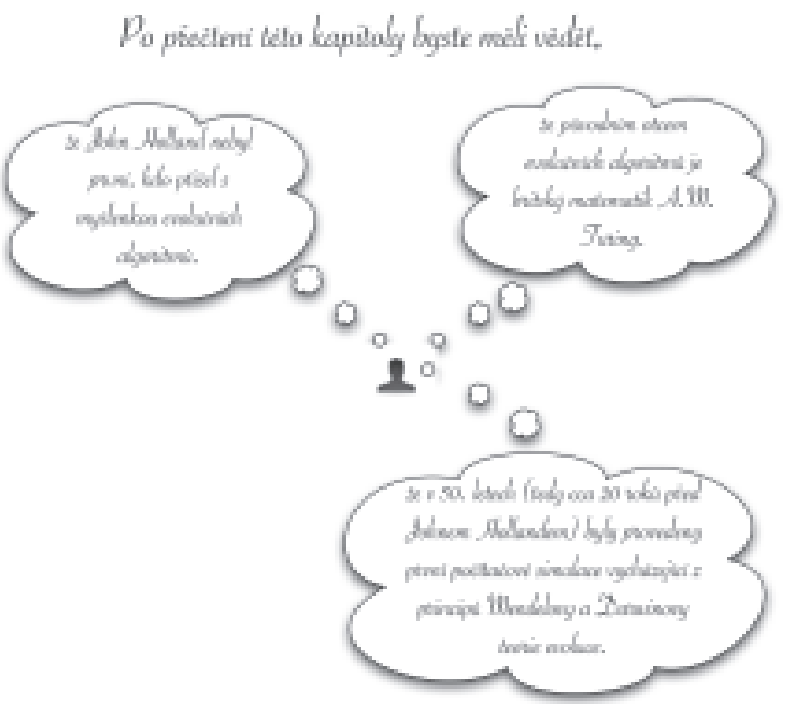
Vážení zákazníci,

dovolujeme si Vás upozornit, že na tuto ukázku knihy se vztahují autorská práva, tzv. copyright.

To znamená, že ukázka má sloužit výhradně pro osobní potřebu potenciálního kupujícího (aby čtenář viděl, jakým způsobem je titul zpracován a mohl se také podle tohoto, jako jednoho z parametrů, rozhodnout, zda titul koupí či ne).

Z toho vyplývá, že není dovoleno tuto ukázku jakýmkoliv způsobem dále šířit, veřejně či neveřejně např. umisťováním na datová média, na jiné internetové stránky (ani prostřednictvím odkazů) apod.

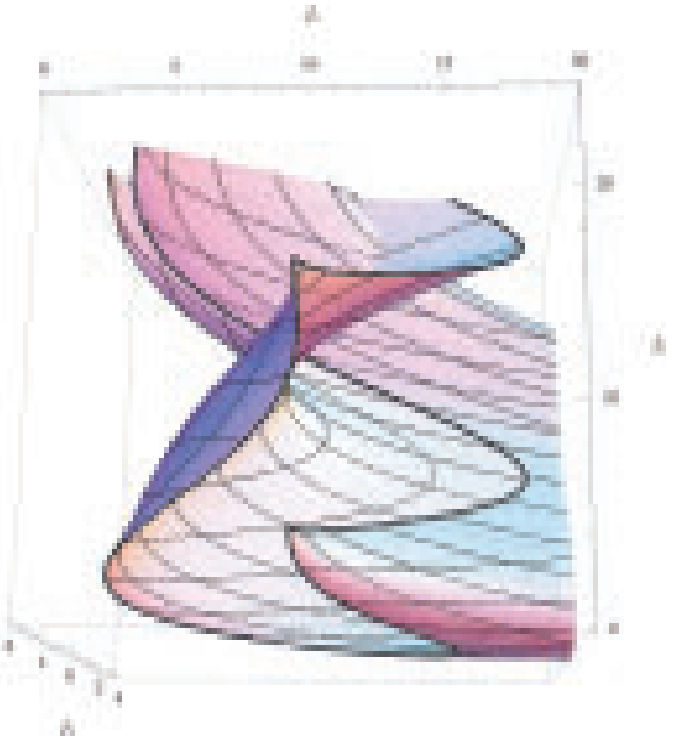
redakce nakladatelství BEN – technická literatura
redakce@ben.cz



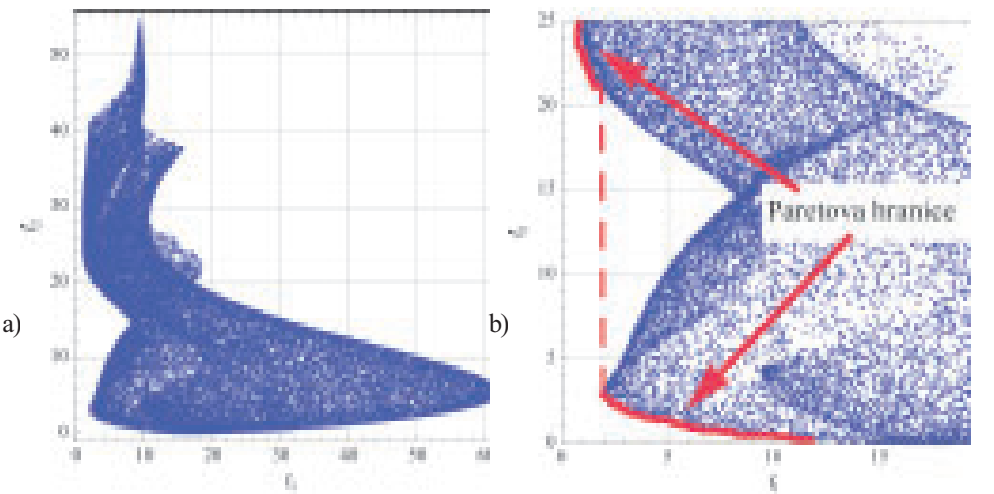
3.1 Pár zajímavých faktů

Začátek historie evolučních algoritmů se obvykle datuje do poloviny 70. let (Holland, 1975), kdy se poprvé objevily genetické algoritmy, případně do poloviny let 60., kdy byly poprvé s úspěchem použity tzv. evoluční strategie (Rechenberg, 1973), (Schwefel, 1977). Vezmou-li se však v úvahu všechna fakta, lze s jistotou tvrdit, že duchovními otci evolučních výpočetních technik jsou takové osobnosti, jako byl matematik A. M. Turing, N. A. Barricelli a jiní. Již v této době byly formulovány a definovány principy, které zcela jasně popisují principy evolučních algoritmů. To, že nebyly programátorsky realizovány, bylo zřejmě dáno nedostatkem výkonné výpočetní techniky.

Obr. 3.1 A. M. Turing



Obr. 7.17 *Detail obr. 7.15. Paretova hranice je ve skutečnosti projekcí křivky v E_3 (tučná červená čára), viz též obr. 7.18b)*



Obr. 7.18 *Standardizované zobrazení množiny možných řešení a Paretovy hranice. K vygenerování bylo použito 40 000 náhodně zvolených bodů podle (7.7)*

11.12 Evoluční strategie

Současně s vývojem genetických algoritmů vznikal v Evropě v šedesátých letech algoritmus pojmenovaný jako „Evoluční strategie“ (ES). Byl vyvinut na Technické univerzitě v Berlíně výzkumníky P. Bienertem, I. Rechenbergem a H. P. Schwefelem. První aplikace tohoto algoritmu byly zaměřeny na problémy funkčního designu z oblasti strojního inženýrství (Rechenberg, 1965), (Schwefel, 1968) a (Lichtfuss, 1965). Evoluční strategie byly vyvinuty v několika variantách – strategiích. První strategie, tzv. dvoučlenné ES (*two-membered ES*), byly vyvinuty a použity z důvodů časové náročnosti na tehdejší, dnes již pravěkou, výpočetní techniku (kalkulátory apod.). Prvním, kdo aplikoval ES na počítač, byl H. P. Schwefel, což odstartovalo vývoj dalších strategií, jako jsou vícečlenné ES (*multi-membered ES*), rekombinativní ES (*recombinative ES*), sebeadaptivní ES (*self-adaptive ES*) atp.

Evoluční strategie se od tehdejších genetických algoritmů (GA) liší v několika ohledech:

- ES používají reprezentaci jedinců v oboru reálných čísel, GA v binární,
- ES nepoužívaly operátory křížení, používaly jen operátory selekce a mutace

Situace u téměř paralelního vývoje ES a GA je analogická jako u simulovaného žíhání (SA). To bylo navrženo dvěma autory (Kirkpatrick, 1983), (Černý, 1985) nezávisle na sobě a v těsném časovém rozpětí.

U značení ES se používá syntaxe „+“ a „ ,“, např. $(\mu + \lambda)$ – ES nebo (μ, λ) – ES. Parametry μ a λ reprezentují označení pro populaci rodičů, resp. potomků, a symboly „+“ a „ ,“ strategii výběru řešení do nové populace. Symbol + znamená, že se do nové populace vybírají nejlepší řešení z populace rodičů a potomků, symbol „ ,“ znamená, že se do nové populace vybírají nejlepší řešení pouze z populace potomků.

11.12.1 Dvoučlenné ES: (1 + 1) –ES

Jedná se o nejjednodušší verzi ES, která pracuje s jedním jedincem-rodičem, z něhož pomocí tzv. Gaussova mutačního operátoru vytváří nového potomka. Pseudokód této strategie je:

(1+1) – ES (x, s, f_{cost}, *iterace, FV*)

vstup:

x: prvopočáteční náhodně vygenerovaný rodič;

σ: směrodatná odchylka pro Gaussovo normální rozdělení

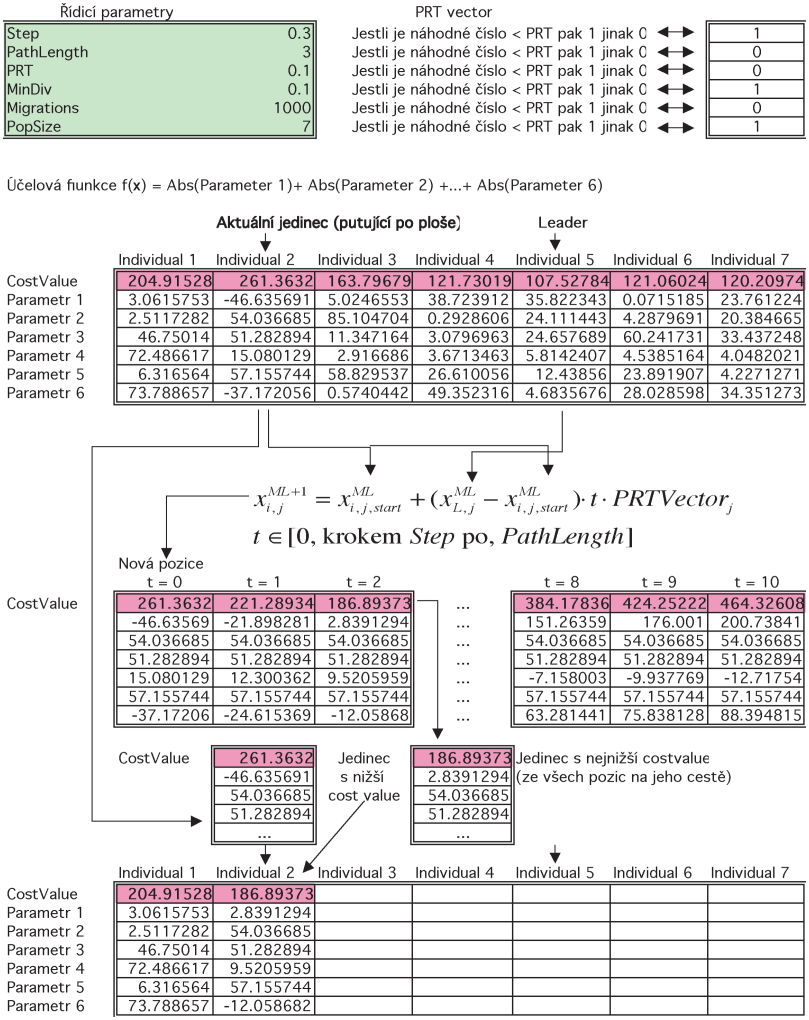
f_{cost}: účelová funkce vracející vhodnost aktuálního řešení

iterace: max. počet iterací ES

FV: vhodnost (*fitness value*), při jejímž dosažení se ES zastaví

pomocné proměnné: y – nový jedinec;

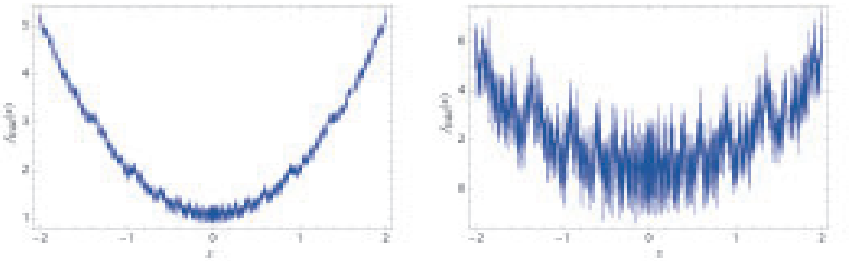
```
for cyklus < iterace do
      begin
            y = x + N(0, σ) „tvorba nového potomka“
            if f_cost (y) < f_cost (x)
                  then begin
                  x = y;
                  end
            if f_cost (x) < FV
```



Obr. 11.37 Princip SOMA: jedno migrační kolo je ukončeno po přemístění všech jedinců. Po migračním kole jsou jedinci přemístěni – viz populaci dole

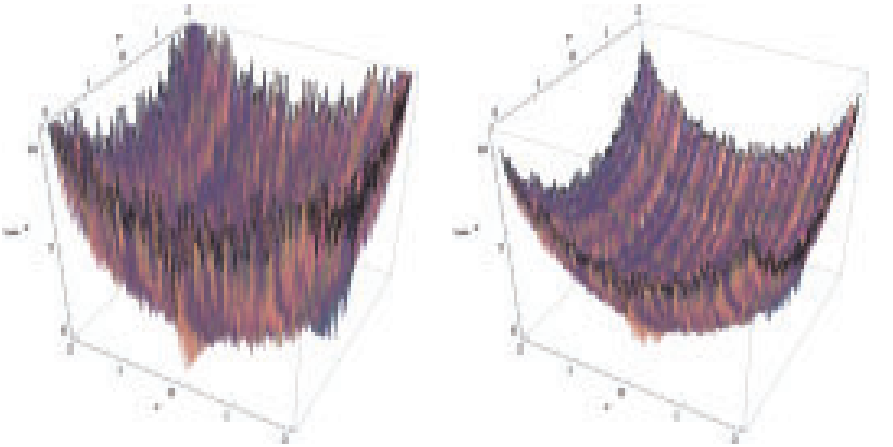
Na obr. 11.37 je spíše konkrétní příklad SOMA. Obecnější popis – pseudokód SOMA lze zapsat:

```
SOMA AllToOne
vstup:
      x: prvopočáteční náhodně vygenerovaná populace rodičů;
      Řídicí a ukončovací parametry algoritmu - viz tab. 11.3
      f_cost: účelová funkce vracející vhodnost aktuálního řešení
      Specimen: vzorový jedinec
      for i < Migrace do
      begin
            selekce nejlepšího jedince - Leadera
```



Obr. 13.26 Fraktální funkce (13.6) superponovaná na funkci z obr. 13.3 s různým stupněm zesílení fraktální složky (viz (13.7))

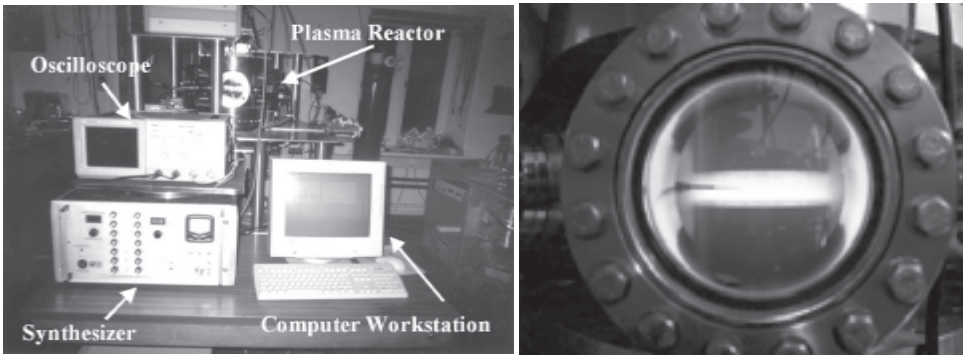
Totéž lze provést i pro vícedimenzionální případy testovacích funkcí, viz obr. 13.27.



Obr. 13.27 Fraktální funkce (13.6) superponovaná na funkci z obr. 13.3 v E_2 s různým stupněm zesílení fraktální složky

Je jasné, že při použití testovacích funkcí tohoto typu nelze předem spočítat pozici a hodnotu globálního extrému, ale naopak je možné vzájemně porovnávat výkony jednotlivých algoritmů.

Členitost výsledné funkce lze ovlivňovat parametrem D. Vliv D na výsledný tvar funkce je jasně vidět na obr. 13.28, kde jsou čtyři verze téže funkce pro různé fraktální dimenze: $D = 1{,}1$; 1,4; 1,6 a 1,9.



Obr. 14.70 Hardvare laboratoře: a) harmonický syntetizátor a PC s OS Unix, b) komora reaktoru s Langmuirovou sondou (tenká čára uprostřed obrázku). Foto© první autor

14.10.5 Nastavení experimentu

Všechny experimenty popsané v tomto článku byly provedeny ve výzkumné laboratoři univerzity v Oxfordu a zařízení výše popsané je fotograficky dokladováno na obr. 14.70. Součástí zařízení byl i digitální osciloskop, který byl použit k měření aktuálních kompenzačních signálů a k ukládání těchto dat na paměťový nosič. Řídicí software běžel na klasickém počítači PC pod operačním systémem Unix. Všechny tři algoritmy byly napsány v jazyku C++ a integrovány do existujícího software, který je v této laboratoři používán k ovládání Langmuirovi sondy. Jako plazmový systém byl použit standardní laboratorní plazmový reaktor.

Tab. 14.17 Parametry plazmy použité v experimentu

Parametry plazmatu*	
Plyn	Argon
Výkon	50 W
Tlak	100 mTorr
Průtok	9,5 sccm

** jednotky ponecháváme raději originální, tak jak byly použity v dřívějších experimentech*

Tab. 14.18 Nejlepší nastavení algoritmů použitá v experimentech

SA	
T_0	25000
decr – dekrement teploty	0,8
n_T iterací za teplotu	50
T_f	4000
Počet částic	3

DE	
CR	0,5
F	0,8
NP	50
Generace	250

SOMA	
PopSize	50
MinDiv	–1
Migrations	14
PathLength	2
Step	0,11
PRT	0,1

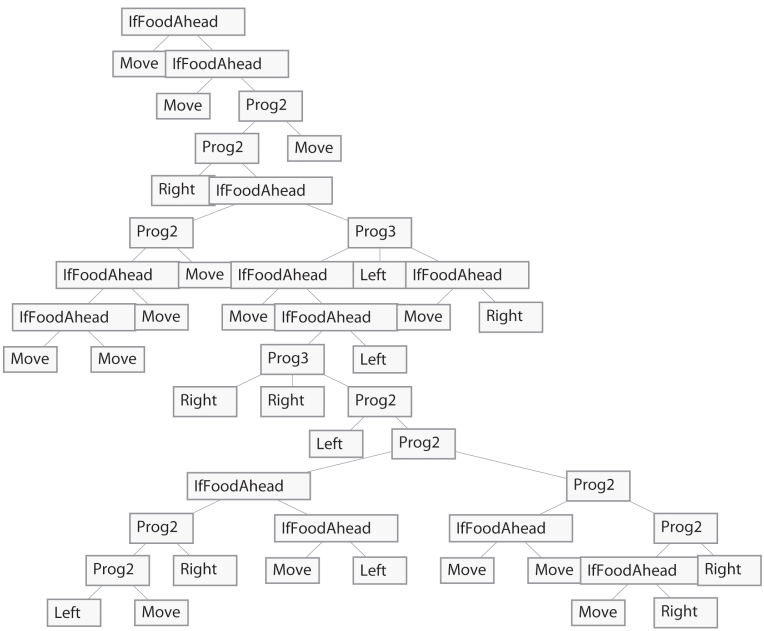
V programu (14.85) lze nalézt i redundandní části. Například ve 2. řádku je zbytečná část – **Prog2[Left, Right]**. Mravenec se otočí vlevo a hned vpravo. Jinými slovy, je to zbytečná operace. Pokud by čtenáře zajímala lingvistická interpretace programu (14.85), pak lze tato pravidla popsat následovně:

Mravenec se podívá před sebe. Jestliže se zde nachází jídlo, přesune se na toto pole a jídlo sebere. Poté se vrací na začátek pravidel a opět se rozhoduje, jestli je na poli před ním jídlo. Pokud ne, mravenec se otočí doprava. Dalším krokem je opět otočení doprava, protože pokračujeme v sekvenci kroků, které přikazuje Prog3. Pak následuje již zmíněná zbytečná operace, kdy se mravenec otočí vlevo a hned se vrátí do původní pozice. A pokračuje testováním přítomnosti jídla před sebou, v kladném případě se přesune a jídlo sebere, v opačném případě se otočí vlevo a teprve pak se přesune. A sada pravidel se začne aplikovat znovu od začátku. Jak je vidět, tento princip neposkytuje přesný postup přesunu po jednotlivých polích ve smyslu souřadnic pohybu.

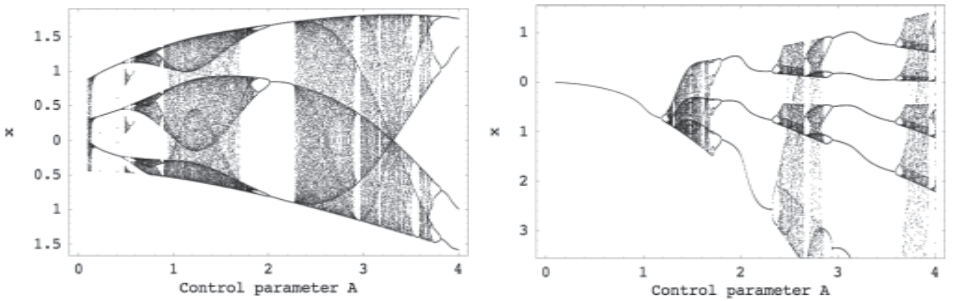
14.14.2 Řešení

Při našich simulacích s AP jsme nalezli sekvence příkazů kratší než zmiňovaných 400 kroků. Jak bude později z celkových výsledků patrné, nejúspěšnější cestu nalezl algoritmus DE, a to za **367** kroků. Pravidla jsou zobrazena ve vztahu (14.86), příp. stromovou reprezentací obr. 14.114.

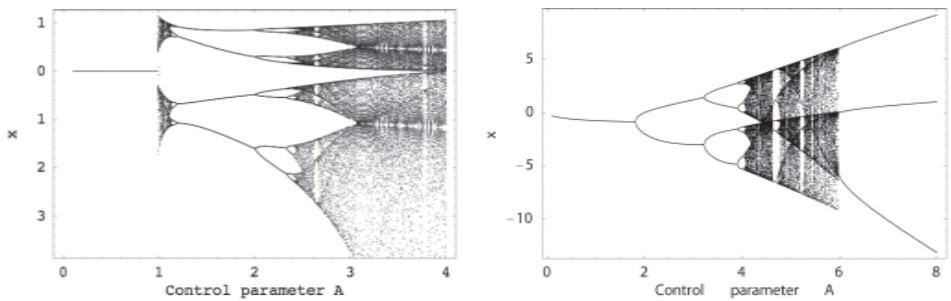
IfFoodAhead[Move, IfFoodAhead[Move, Prog2[Prog2[Right, IfFoodAhead[Prog2[IfFoodAhead[IfFoodAhead[Move, Move], Move], Move], Prog3[IfFoodAhead[Move, IfFoodAhead[Prog3[Right, Right, Prog2[Left, Prog2[IfFoodAhead[Prog2[Prog2[Left, Move], Right], IfFoodAhead[Move, Left]], Prog2[IfFoodAhead[Move, Move], Prog2[IfFoodAhead[Move, Right], Right]]]]], Left]], Left,IfFoodAhead[Move, Right]]]], Move]]] **(14.86)**



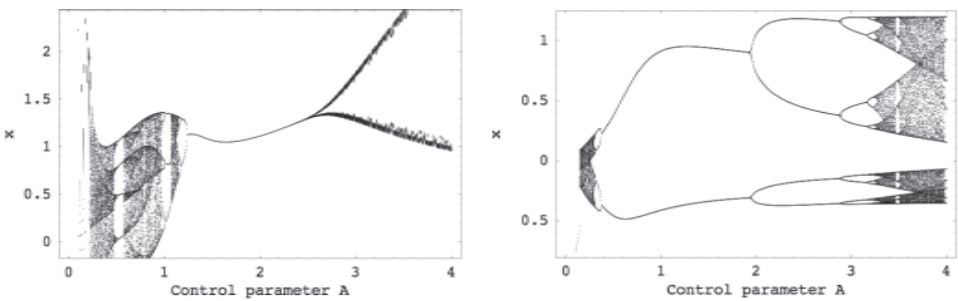
Obr. 14.114 Stromová reprezentace programu (14.86)



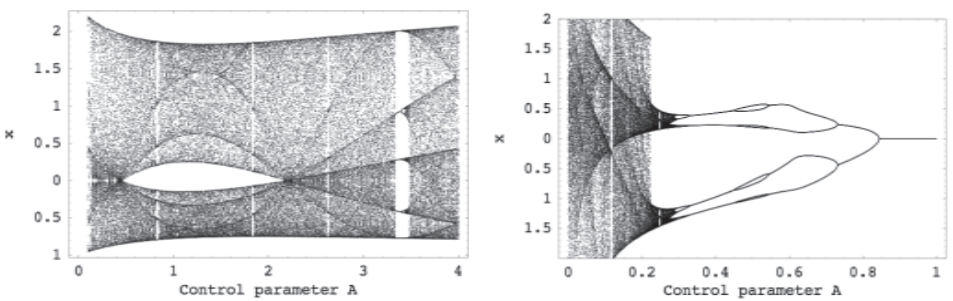
Obr. 14.142 Bifurkační diagramy podle (14.110) a (14.111)



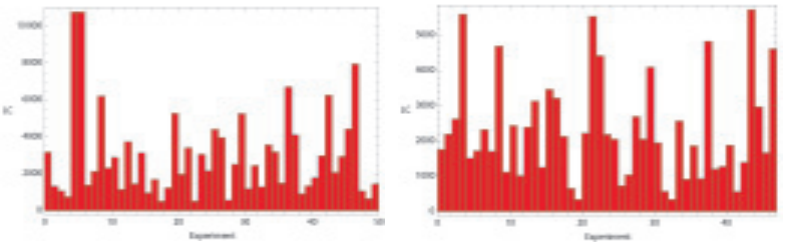
Obr. 14.143 Bifurkační diagramy podle (14.112) a (14.113)



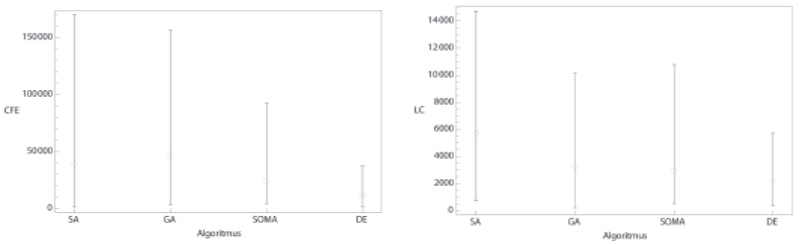
Obr. 14.144 Bifurkační diagramy podle (14.114) a (14.115)



Obr. 14.145 Bifurkační diagramy podle (14.116) a (14.117)



Obr. 14.149 Délka programů SOMA (vlevo) a DE (vpravo)



Obr. 14.150 Výkon algoritmů problému sudé-3-parity: délka programů (Leaf Count – LC) generovaných algoritmy SA, GA, SOMA a DE (vpravo) a počet ohodnocení účelové funkce u algoritmů SA ,GA, SOMA and DE (vlevo)

$$\begin{aligned}&((C \,\overline{\vee}\, A) \vee B \vee (\neg C \wedge B \wedge A) \vee (\neg B \wedge C \wedge A) \vee (\neg A \wedge C \wedge B) \vee \\ &\quad ((((\neg C \wedge B \wedge A) \vee (\neg B \wedge C \wedge A) \vee (\neg A \wedge C \wedge B)) \,\overline{\vee}\, A) \,\overline{\wedge}\, (C \vee B))) \wedge \\ &((C \vee B) \,\overline{\vee}\, (A \vee C)) \vee ((C \vee A) \wedge ((\neg C \wedge B \wedge A) \vee (\neg B \wedge C \wedge A) \vee (\neg A \wedge C \wedge B))))\end{aligned} \qquad \textbf{(14.119)}$$

Vztah (14.119) lze rovněž zobrazit klasicky jako

((Nor[C, A]) || B || (! C && B && A) || (! B && C && A) || (! A && C && B) || (Nand[Nor[(! C && B && A) || (! B && C && A) || (! A && C && B), A], C || B])) && ((Nor[C || B, A || C]) || ((C || A) && ((! C && B && A) || (! B && C && A) || (! A && C && B))))

nebo formou stromu, viz obr. 14.151.

Obr. 14.151 Strom výrazu (14.119)