Algoritmus Diferenciální Evoluce s prvky deterministického chaosu (ChaosDE) v prostředí C/C++

Differential Evolution Algorithm with Elements of Deterministic Chaos in C/C++

Anežka Kazíková

Bakalářská práce
2013

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Anežka KAZÍKOVÁ**
Osobní číslo:	**A10053**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Algoritmus Diferenciální Evoluce s prvky deterministického chaosu (ChaosDE) v prostředí C/C++**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na dané téma.**
2. **Naprogramujte algoritmus diferenciální evoluce s prvky deterministického chaosu v prostředí C/C++.**
3. **Otestujte algoritmus na sadě vybraných testovacích funkcí.**
4. **Výsledky testování přehledně graficky a tabulkově zobrazte.**
5. **Vybrané výsledky pro zvolené strategie porovnejte a zhodnoťte.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. 1. vyd. Praha: BEN - technická literatura, 2002. ISBN 80-7300-069-5.**
2. **OPLATKOVÁ, Zuzana, Pavel OŠMERA, Miloš ŠEDA, František VČELAŘ a Ivan ZELINKA. Evoluční výpočetní techniky: principy a aplikace. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.**
3. **POSPÍCHAL, Jiří, Vladimír KVASNIČKA, Peter TIŇO, František VČELAŘ a Ivan ZELINKA. Evolučné algoritmy: principy a aplikace. 1. vyd. Bratislava: Slovenská technická univerzita, 2000, 215 s. Edícia vysokoškolských učebníc. ISBN 80-227-1377-5.**
4. **PRICE, Kenneth V, Rainer M STORN, Jouni A LAMPINEN, František VČELAŘ a Ivan ZELINKA. Differentional evolution: a practical approach to global optimization. 1. vyd. Berlin: Springer, 2005, xix, 538 s. Edícia vysokoškolských učebníc. ISBN 35-402-0950-6.**
5. **FUCHS, Radovan. Deterministický chaos jako generátor náhodných čísel v prostředí C a C++. Zlín, 2012. Bakalářská práce. UTB ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Ing. Roman Šenkeřík, Ph.D.**
6. **HORÁK, Jiří. Deterministický chaos a jeho fyzikální aplikace. Vyd. 1. Praha: Academia, 2003, 437 s., viii s. barev. obr. příl. ISBN 80-200-0910-8.**
7. **ZELINKA, Ivan, Zuzana OPLATKOVÁ a Roman ŠENKEŘÍK. Aplikace umělé inteligence. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2010, 151 s. ISBN 978-80-7318-898-6.**

Vedoucí bakalářské práce: **Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **24. února 2013**

Termín odevzdání bakalářské práce: **14. června 2013**

Ve Zlíně dne 24. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Bakalářská práce se zabývá algoritmem diferenciální evoluce s prvky deterministického chaosu. Její součástí je aplikace implementovaná v prostředí C++, která na základě tohoto algoritmu používá evoluci na množství testovacích funkcí pro optimalizační úlohy. Teoretická část práce obsahuje principy diferenciální evoluce, popis testovaných účelových funkcí a základní pojmy deterministického chaosu. V praktické části je popsán objektový návrh aplikace a závislost diferenciální evoluce na různých parametrech a to na základě výsledků popsané aplikace.

Klíčová slova: diferenciální evoluce, deterministický chaos, testovací funkce, křížení, mutace

ABSTRACT

Bachelor's thesis describes the differential evolution algorithm with elements of deterministic chaos. It includes an application implemented in C++ based on this algorithm, which uses evolution on a number of test functions for optimization problems. Theoretical part of the thesis describes the principles of differential evolution, test functions and basic concepts of deterministic chaos. Practical part consists of the object-oriented design of the application and the discussion of the sensitivity of the differential evolution based on the results of the described application.

Keywords: differential evolution, deterministic chaos, test functions, crossing, mutation

Tímto bych chtěla poděkovat Ing. Romanu Šenkeříkovi, Ph.D. za ochotu, pomoc a cenné rady při tvorbě této práce. Děkuji také své rodině – manželovi a rodičům za trpělivost a podporu, kterou mne provázeli po celou dobu mého studia.

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….…….

podpis diplomanta

OBSAH

ÚVOD

Moderní informační technologie jsou prostoupeny trendem nechat se inspirovat přírodou. Využití nacházíme v robotice, fraktálové matematice, umělé inteligenci a v mnoha dalších oblastech aplikované informatiky. Optimalizace (obor, který se zabývá hledáním extrémů funkcí) nachází tuto inspiraci v evolučních algoritmech.

Evoluční algoritmy využíváme v případech, kdy je předmětem zkoumání netriviální n-dimenzionální funkce a neznáme jednoznačný postup vedoucí k jejímu řešení. Jednotlivé návrhy řešení pěstujeme v tzv. populacích, které vyvíjejí potomky pomocí mutace a křížení. Stejně jak u „přírodních vzorů“ pracuje evoluce s pojmy jako stagnace či degenerace populace.

Algoritmus diferenciální evoluce je jedním z mnoha algoritmů evolučních technik. Je určen konkrétním způsobem mutace a křížení a obecně je známo, že dosahuje dobrých výsledků. K evolučním algoritmům můžeme přidat i náhodnost v podobě generování čísel za pomoci deterministického chaosu. I ten je inspirován v přírodě.

Cílem této práce je vytvořit plně funkční program v prostředí C/C++, který bude možno použít pro optimalizační úlohy pomocí algoritmu diferenciální evoluce a generátoru pseudo-náhodných čísel, který využívá deterministický chaos. Dále vysvětlit samotný princip diferenciální evoluce a pomocí vytvořené aplikace otestovat efektivitu daného algoritmu.

I. TEORETICKÁ ČÁST

1 EVOLUČNÍ ALGORITMY

Základní princip evoluce, tak jak ho známe z Darwinovy a Mendelovy teorie, spočívá v předávání rodičovského genomu (genů a nekódující sekvence konkrétního organismu) svým potomkům a následnému uvolnění jejich životního prostoru. Setkáváme se s pojmem mutace, které podléhá předávaná informace a v jejímž důsledku se generace vyvíjí. Nevhodní jedinci pro aktuální životní prostředí pak dle Darwinovy teorie „vymírají“, zatímco ti, kteří se dokázali adaptovat, zůstávají. [1]

Algoritmy evoluční výpočetní techniky (EVT) z tohoto principu vychází. Namísto vývoje živých organismů ovšem pracujeme s výpočetní problematikou – jednotliví jedinci tedy nejsou živočichové, ale číselně vyjádřené parametry, které pomocí základních postupů evoluce (mutací a křížením) vyvíjíme a upravujeme tak, abychom v nových generacích dosáhli optimálních výsledků.

Konkrétních algoritmů vycházejících z evoluční teorie je mnoho. Jejich principy se mírně liší, ale ať už se jedná o genetické algoritmy, rojení částic, SOMA (samoorganizující se migrační algoritmus), diferenciální evoluci či jiné, základní myšlenka Darwinovy a Mendelovy teorie zůstává.

1.1 Základní pojmy evolučních algoritmů

1.1.1 Jedinec

Jedinec je numerický popis jednoho řešení daného problému. Každý jedinec má svou vlastní kvalitu, která je výsledkem ohodnocení účelové funkce. Samotného jedince si můžeme představit jako číselný vektor parametrů s jeho ohodnocením. Každý jedinec je omezen hranicemi prohledávaného prostoru. Při jeho vytváření bychom měli používat korekci parametrů, které daný prostor překročí. Účelem evolučních algoritmů je najít jedince s ideální účelovou funkcí.

Tab. 1. Znázornění jedince

CV (ohodnocení jedince)	**Parametr 1**	**Parametr 2**	**Parametr 3**
2,88277	-0,037139	-1,6088	0,541448

1.1.2 Populace

Populace je pole jedinců dané generace včetně jejich parametrů. Může být znázorněna jako matice NP x D (počet jedinců v populaci x dimenze problému) s přidaným řádkem jednotlivých ohodnocení jedinců (CV). Sloupce představují jednotlivé jedince. [2]

Tab. 2. Znázornění populace

	Jedinec 1	**Jedinec 2**	**Jedinec 3**	**Jedinec 4**	**Jedinec 5**
CV	2,88277	8,67739	12,2754	1,76051	1,03928
Parametr 1	-0,037139	2,25519	0,882058	-1,27924	0,50194
Parametr 2	-1,6088	-0,80931	-1,12773	0,315045	-0,176007
Parametr 3	0,541448	1,71363	-3,19775	-0,157448	0,869688

1.1.3 Účelová funkce

Účelová funkce hodnotí kvalitu jedince dle jeho parametrů. Označujeme ji $f(x)$ nebo také $f_{cost}(x)$ (cost znamená anglicky cena). V rámci optimalizace nahlížíme na účelovou funkci jako na geometrický problém, kdy se snažíme nalézt minimum (resp. maximum) dané funkce v N – dimenzionálním prostoru možných řešení. Počet dimenzí nám zde udává počet argumentů účelové funkce. [2]

V rámci praktické části této práce se setkáváme s několika účelovými funkcemi včetně jejich grafického zobrazení. Příkladem účelové funkce tedy může být například jakákoliv z uvedených testovacích funkcí (viz kapitola 2.6).

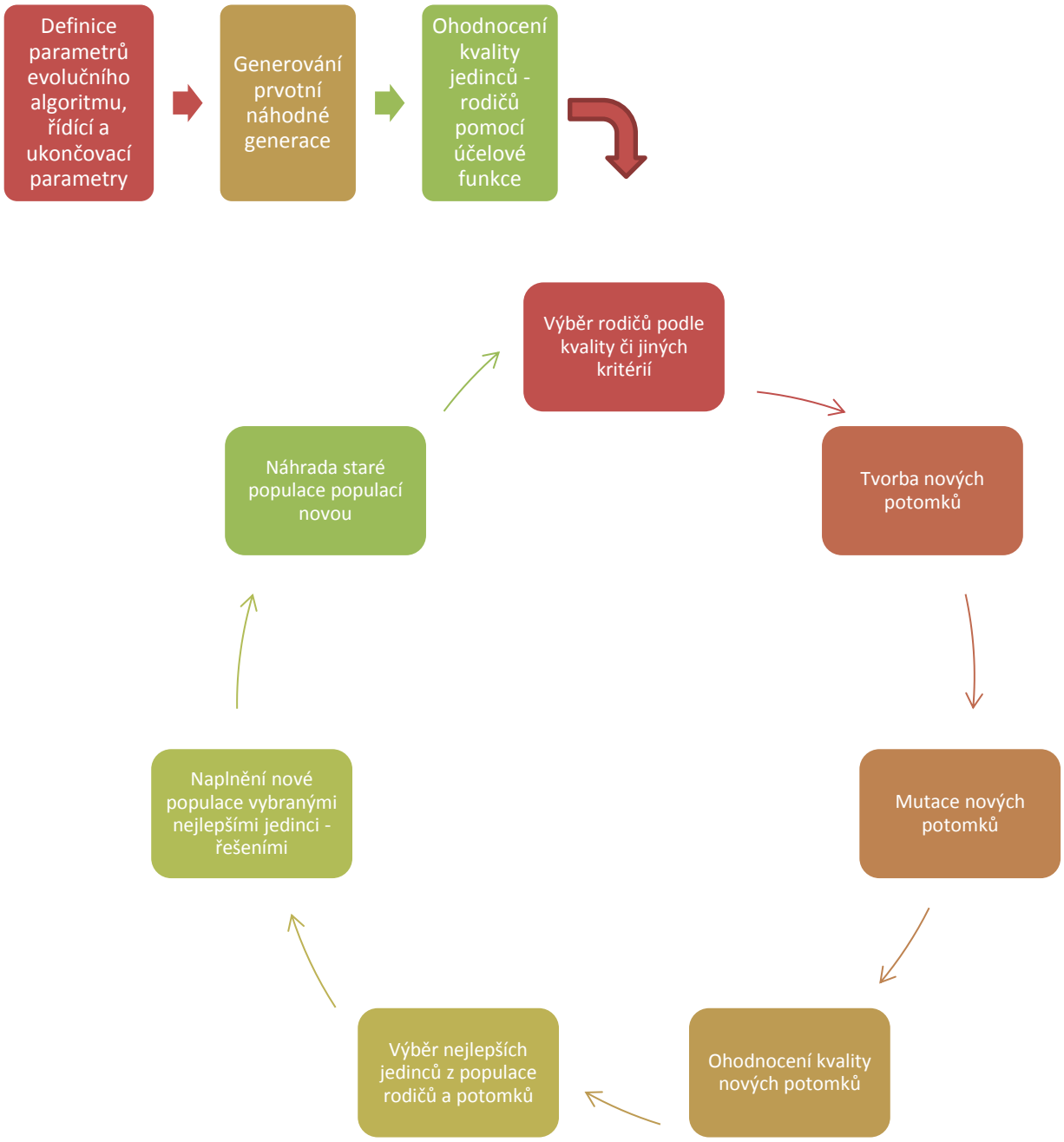
Využití účelové funkce ovšem přesahuje rámec ukázaných testovacích funkcí. V případě prediktivního řízení může účelová funkce například obsahovat veličiny akčního zásahu u, výstupu y, a žádanou hodnotu w, ke které se snažíme řídit výstupní veličinu. Vidíme tak, že aplikace účelové funkce je široká a multioborová.

1.2 Obecný cyklus evolučních algoritmů

Obecný postup EVT začíná vymezením a definicí parametrů evoluce. Ty se u každého evolučního algoritmu mění. Mezi parametry můžeme řadit například počet generací, které chceme vytvořit, stanovení účelové funkce pro řešený problém a další specializované atributy daného algoritmu. Pak vygenerujeme počáteční populaci a ohodnotíme její kvalitu pomocí účelové funkce.

Dokud nejsou splněny ukončující podmínky – například vytvoření zadaného maximálního počtu generací, opakuje se samotný evoluční cyklus:

z aktuální populace se vyberou rodiče a jejich křížením a mutací vytvoříme potomky. Kritéria výběru rodičů a jednotlivé postupy křížení a mutace jsou variabilní vzhledem k vybranému algoritmu. Každý nový jedinec se ohodnotí a vyberou se z nich ti nejlepší. Ti pak zaplní novou populaci. Stará populace je tak zapomenuta a její místo zaujímá populace nová. [1]



Obr. 1. Obecný cyklus evolučního algoritmu bez znázornění ukončení evoluce

1.3 Využití evolučních algoritmů

Na rozdíl od jiných algoritmů není u EVT zajištěno, že nalezneme to nejlepší a optimální řešení. (Také některé živočišné druhy nebyly v důsledku měnících se podmínek schopny adaptace a vymřely.) Přesto dosahujeme ve spoustě případů, kde by bylo řešení netriviální, či kde kroky vedoucí k jeho nalezení dokonce ani neznáme, velmi dobrých výsledků.

2 DIFERENCIÁLNÍ EVOLUCE

Diferenciální evoluci vyvinuli v roce 1995 Kenneth V. Price a Dr. Rainer Storm. Algoritmus vznikl úpravou genetického žíhání pro řešení složitějších problémů. Původní binární reprezentace byla nahrazena dekadickou a logické operace vektorovými. Po přidání principu diferenciální mutace (viz níže) byly ovšem prvky genetického žíhání uznány za nadbytečné a z diferenciální evoluce (dále jen DE) byly vypuštěny.

Z ostatních evolučních algoritmů se DE schematicky podobá genetickým algoritmům a zároveň právě v porovnání s GA je velmi patrná rozdílnost obou postupů. Příkladem může být například odlišný počet rodičů, které algoritmy potřebují k vytvoření potomků. Zatímco GA používá dva rodiče, DE je potřebuje čtyři.

2.1 Parametry a terminologie

Mezi parametry DE patří:

- **CR – práh křížení**

  Interval: $CR \in [0, 1]$

  CR se využívá při vytváření nového jedince konkrétně během křížení. Určuje míru dědičnosti nově vzniklého jedince na čtvrtém rodiči nebo na zmutovaném vektoru. Pokud je CR rovno 0, bude potomek kopií čtvrtého rodiče. V opačném případě bude-li se hodnota CR rovnat 1, nový potomek bude okopírovaný zmutovaný vektor a ze čtvrtého rodiče nezíská žádnou informaci. V těchto krajních případech by se tedy z evolučního algoritmu stalo buďto pouhé náhodné hledání nebo jen kopírování starších generací bez vývoje. Doporučuje se proto, aby CR nikdy nenabývalo krajních hodnot.

  V případě, že ošetřujeme separabilní funkci, je doporučeno nastavit tento parametr na hodnoty blížící se k 0.

- **D – dimenze problému**

  D udává počet argumentů účelové funkce. Pokud prohledáváme například dvojrozměrný prostor, pak bude mít účelová funkce právě dva argumenty (pozici na ose x a na ose y).

- **NP – velikost populace**

  Interval: $NP \in [10D, 100D]$

  Udává počet jedinců v jedné generaci. Populace by nikdy neměla být nižší nežli 4, aby evoluce mohla ještě fungovat. (Pro vytvoření potomka je potřeba čtyř rodičů.)

- **F – mutační konstanta**

  Interval: $F \in [0, 2]$

  F je využíván během diferenciální mutace rozdílně v závislosti na vybrané mutační strategii. Ovlivňuje tedy podobu vzniklých zkušebních vektorů.

- **Generations – generace**

  Interval: $Generations > 0$

  Udává počet evolučních cyklů šlechtění populace.

- **Specimen – prototyp jedince**

  Udává typ čísel a interval, ze kterého se skládají jednotliví jedinci. V podstatě se jedná o omezení, v rámci kterého se jednotlivé parametry jedince mají hledat. Například: *{(Real, -10, 15), (Integer, -1, 1)}*. Díky omezení pak můžeme zajistit, aby byli jedinci generování v rámci daného intervalu. [1]

Následující tabulka (Tab. 3) ukazuje doporučené hodnoty těchto parametrů, jak jsou uvedeny ve zdroji [1].

Tab. 3. Hodnoty řídících parametrů diferenciální evoluce

Řídící parametr	**Interval**	**Doporučená**	**Poznámka**
CR	[0, 1]	0,8 – 0,9	Práh křížení
NP	[10D, 100D]	10D	Velikost populace
F	[0, 2]	0,3 – 0,9	Mutační konstanta
Generations	> 0		Počet kol evoluce populace

Parametry evolučních algoritmů určují kvalitu a průběh evoluce. V základní verzi DE jsou parametry sice voleny jako pevné, v modifikovaných verzích DE je ale lze v závislosti na kvalitě probíhající evoluce měnit. V takovémto případě pokud k vyhodnocení vhodných parametrů využijeme diferenciální evoluci samotnou, hovoříme o tzv. meta-diferenciální evoluci. Parametry této DE je trojice NP, CR, F. Samotná evoluce pak má sice efektivnější

výsledky, ovšem za cenu prodlouženého evolučního procesu. Časově je meta-diferenciální evoluce mnohem náročnější nežli obyčejná DE. [2]

2.2 Mutace

Mutace spočívá ve vytvoření tzv. šumového vektoru *v* za použití náhodně vybraných rodičů a mutační konstanty. Konkrétní způsob jeho vzniku záleží na zvolené mutační strategii.

2.2.1 Mutační strategie

Možností tvorby šumového vektoru je samozřejmě veliké množství. Aby se jednalo o diferenciální mutaci, stačí v podstatě při výpočtu šumového vektoru pouze použít rozdíl dvou náhodně vybraných vektorů rodičů. Seznam některých variant mutačních strategií je uveden v tabulce (Tab. 4).

Vzhledem k tomu, že využíváme náhodně vybrané rodiče, na jejich pořadí při výpočtu nezáleží. Je tedy jedno, zda v dané strategii použijeme $x_{r1,j}^G$, $x_{r2,j}^G$ či $x_{r3,j}^G$.

Tab. 4. Mutační strategie [2]

Best/1	$v = x_{best,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$	(1)
Rand/1	$v = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$	(2)
Rand-To-Best/1	$v = x_{i,j}^G + \lambda \cdot \left(x_{best,j}^G - x_{i,j}^G\right) + F \cdot (x_{r1,j}^G - x_{r2,j}^G)$	(3)
Best/2	$v = x_{best,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G)$	(4)
Rand/2	$v = x_{5,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G)$	(5)

Strategií s doposud nejlepšími výsledky je strategie Rand/1 (2), která využívá tří různých náhodně vybraných rodičů dané populace a mutační konstanty *F*. Během této mutace nejprve náhodně vybereme tři rodiče, rozdíl prvního a druhého vynásobíme mutační konstantou a výsledný vektor přičteme k třetímu rodiči.

Dalším možným strategickým krokem je používat jako jednoho z rodičů jedince s doposud nejlepšími výsledky. Jedná se o variace obsahující slovo „Best“.

Jednotlivé strategie se liší i v závislosti na vybraném křížení. Kompletní shrnutí najdete podrobněji popsané v kapitole 2.4.1 Strategie DE.

2.3 Křížení

Na rozdíl od GA, nastává křížení u DE až po mutaci. Křížením vytvoříme ze šumového vektoru a čtvrtého (dosud nepoužitého) rodiče nového jedince, kterému budeme říkat zkušební (nebo také trial) vektor. Jeho vznik závisí na zvolené strategii, náhodně generovaném čísle a velikosti CR.

2.3.1 Binární křížení

U binární strategie křížení (označujeme též Bin) vycházíme z náhodně vygenerovaného čísla a hodnoty prahu křížení CR. Odpovídající dvojice parametrů (první parametr šumového vektoru s prvním parametrem aktivního rodiče, druhý s druhým, atd.) vždy vybereme právě jeden a to v závislosti na náhodně generované hodnotě. Je-li náhodná hodnota nižší nežli práh křížení CR, bereme parametr ze šumového vektoru, v opačném případě pak z aktivního rodiče.

Strategie se dá zapsat vztahem:

$$y_j = \begin{cases} v_j & pokud \quad U_j \leq CR \\ x_{ij} & pokud \quad U_j > CR \end{cases} \tag{6}$$

Kde y_j zde reprezentuje vzniklý prvek trial vektoru, v_j odpovídající parametr šumového vektoru, x_{ij} parametr aktivního rodiče a $U_j \in [0,1]$ náhodně vygenerovaný prvek. [4]

Nově vzniklý jedinec není automaticky zařazen do nové populace, ale soutěží o toto místo s aktivním rodičem a to pomocí hodnotící funkce. Tím je zajištěna garance imunity proti degradaci populace. [2]

Možné jsou též drobné modifikace. V jedné z nich vybíráme pořadí prvního evaluovaného parametru náhodně a automaticky ho řadíme z příslušného parametru šumového vektoru. Tím se dá například zajistit, že i v případě nulové hodnoty CR, se alespoň jeden prvek nového jedince změní a celý vektor nebude pouhou kopií svého rodiče. Veškeré další hodnoty pak vybíráme dle algoritmu binárního křížení. Takovýto algoritmus by se zapsal vztahem (7):

$$y_j = \begin{cases} v_j & pokud \quad U_j \leq CR \quad nebo \quad j = l \\ x_{ij} & pokud \quad U_j > CR \quad a \quad j \neq l \end{cases} \tag{7}$$

Kde l je náhodně vybrané celé číslo z intervalu $\{1, 2, \dots, D\}$.[4]

2.3.2 Exponenciální křížení

U exponenciálního křížení (též Exp) nejprve náhodně vybereme startovní pozici z intervalu $\{1, 2, \dots, D\}$. Následně vycházíme z parametrů šumového vektoru do té doby, dokud nevygenerujeme náhodné číslo větší nežli CR. Jakmile ovšem náhodná hodnota přesáhne CR, kopírujeme veškeré zbylé hodnoty aktivního rodiče až do naplnění nového jedince. [4]

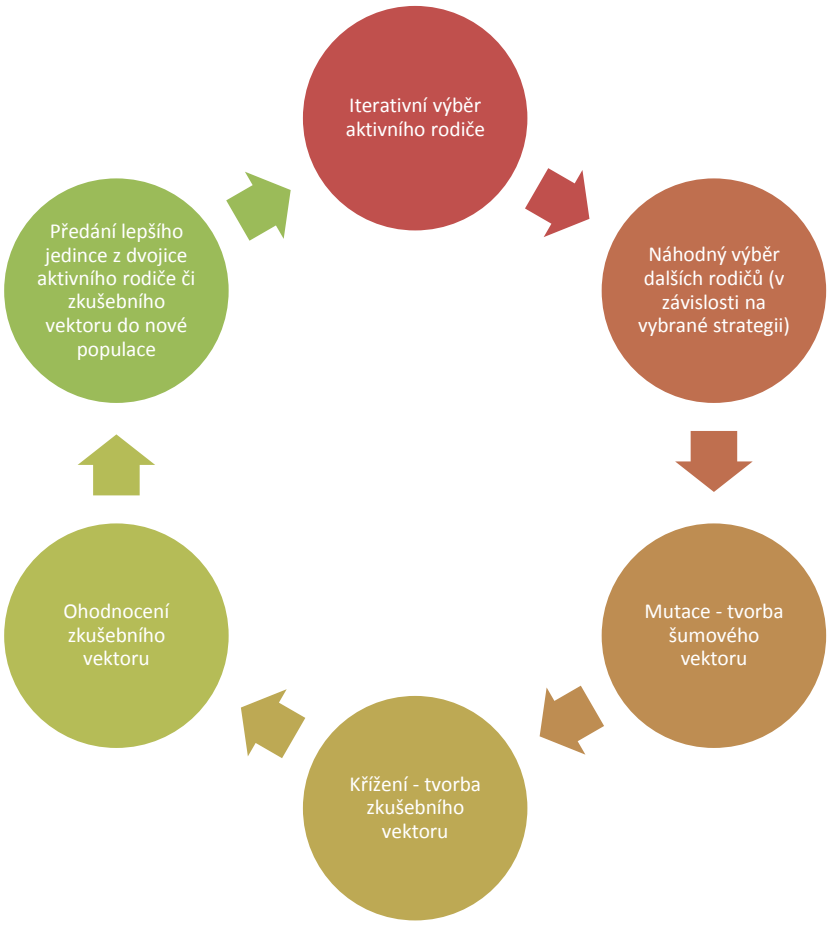
Hodnota účelové funkce nového jedince se porovná s aktivním rodičem a stejně jako u binárního křížení do příští generace postoupí ten lepší.

Rozdíl mezi exponenciálním a binárním křížením spočívá tedy především v tom, že u exponenciální strategie přiřazujeme jednotlivé parametry ze stejného zdroje vždy za sebou, zatímco u binární strategie se prvky z jednotlivých zdrojů mohou střídat. Dále snížené opakování porovnávání a generování nových náhodných prvků u exp. křížení teoreticky může za určitých podmínek snížit dobu vykonávání tvorby nového jedince. Ačkoliv by podle internetové stránky Dr. Reinera Storna samotná metoda křížení neměla mít na výsledek příliš značný vliv, Ken Price tvrdí, že binární křížení není nikdy horší nežli exponenciální. [5] Kapitola 5.1.2 obsahuje srovnání výsledků jednotlivých strategií testovaných v rámci zkoumané aplikace.

2.4 Princip činnosti DE

Průběh diferenciální evoluce je obdobný k obecnému evolučnímu algoritmu, jak je uveden v kapitole 1.2. Začíná stanovením parametrů (F – mutační konstanta [0, 2], CR – práh křížení [0, 1], NP – počet jedinců v populaci, D – rozměr jedince a Specimen – prototyp jedince). Následuje tvorba první populace a její ohodnocení.

Samotný cyklus diferenciální evoluce bez znázornění ukončovacích podmínek a vyhodnocení jednotlivých generací zobrazuje obrázek (Obr. 2). Jedna otočka cyklu vytvoří jednoho jedince do nové populace.



Obr. 2. Obecný diferenciální evoluční cyklus bez znázornění ukončení evoluce

Cyklus začíná výběrem aktivního rodiče. Toho vybíráme klasickým iteračním procesem, kdy postupně procházíme veškeré jedince v populaci. Aktivnímu rodiči pak přiřadíme další tři nestejné náhodně vybrané jedince, na které aplikujeme mutaci. Mutační (též šumový) vektor pak zkřížíme s aktivním rodičem na základě náhodně vygenerované hodnoty a velikosti CR.

Vzniklého nového jedince (nyní zkušební vektor) ohodnotíme účelovou funkcí. Pokud hodnota účelové funkce (CV – cost value) bude horší, nežli u aktivního rodiče, přechází do nové populace aktivní rodič. V opačném případě jsme úspěšně vytvořili nového jedince, který je lepší, nežli byl jeho rodič. Díky tomuto poslednímu kroku máme zajištěno, že populace pozdějších generací bude vždy lepší nebo stejná než generace předchozí.

Následuje výběr dalšího aktivního rodiče a cyklus evoluce se opakuje až do vyčerpání populace. Ve své základní verzi je DE ukončena, pokud vyšlechtíme zadaný počet generací. Programátor si ovšem může ukončující podmínku sám upravit – například ošetřit případ, kdy nastává tzv. stagnace, případně změnit za určitých podmínek řídící parametry.

V rámci každé generace se uchovává hodnota nejlepšího jedince. [1]

2.4.1 Strategie DE

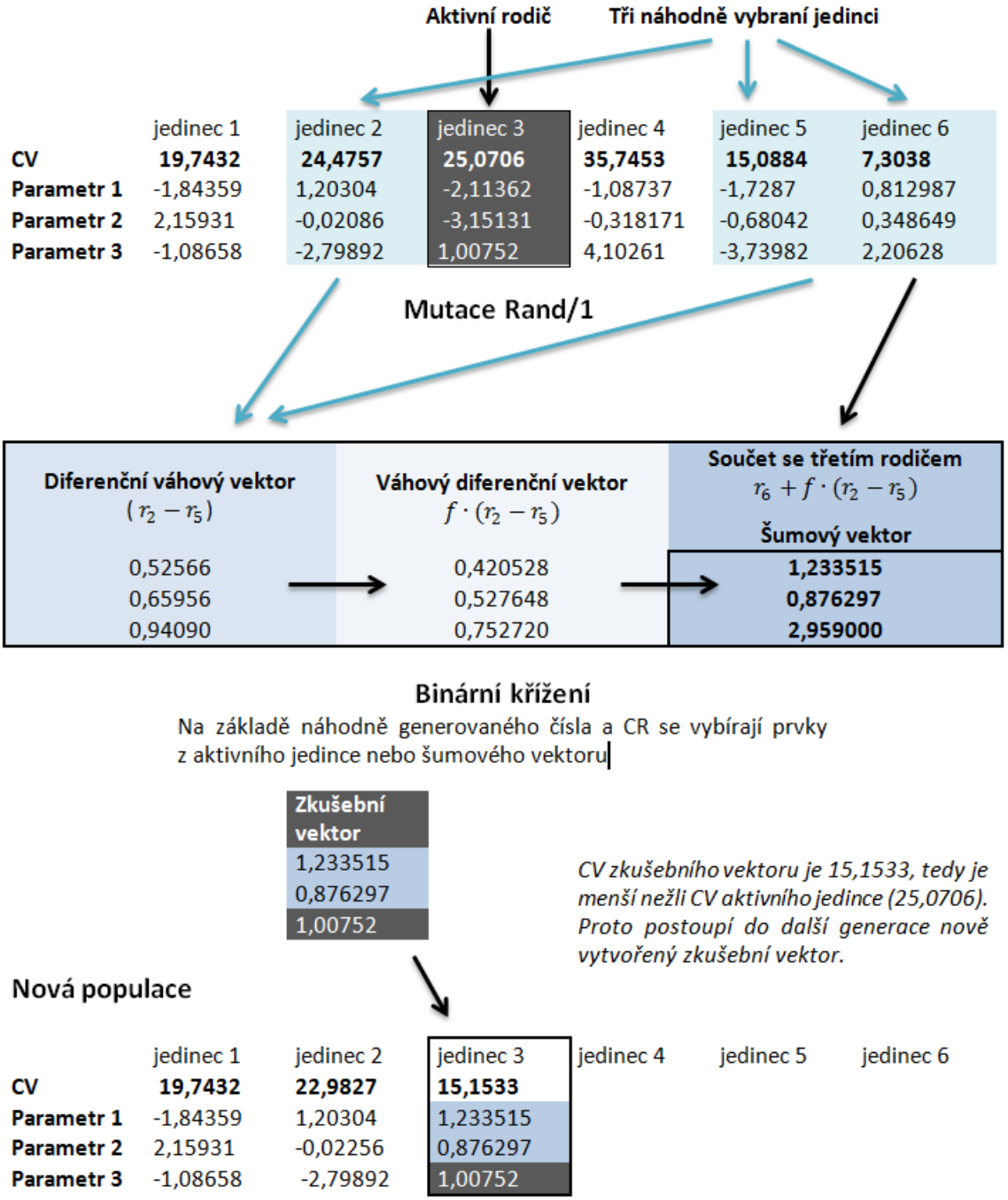
Jednotlivé variace strategií DE se skládají z části mutační a křížení. Obecná konvekce zápisu strategie je ve formě DE/x/y/z. DE určuje, že se jedná o diferenciální evoluci, *x* představuje výběr aktivního prvku, *y* počet diferencí vektorů potřebných k mutaci *x* a *z* udává strategii křížení. [3]

Mutační strategie, jak jsem ji představila v kapitole 2.2.1, je tedy popsána parametry *x* a *y*. Pro úplnost doplňuji i hodnoty, kterých mohou jednotlivé položky nabývat.

Tab. 5. Strategie diferenciální evoluce

DE	**x**	**y**	**z**
	Best	1	Exp
	Rand	2	Bin
	Rand-to-Best	…	

V závislosti na vybrané strategii se tedy mění i jednotlivé kroky evolučního cyklu. Strategie DE/Rand/1/Bin je zobrazena na obrázku (Obr. 3).



Obr. 3. Princip DE/Rand/1/Bin

2.5 Stagnace

Evoluční algoritmy mohou nabývat suboptimálních výsledků (tj. výsledků, při kterých nebyl nalezen globální extrém, ale třeba jen lokální) v případě, že je splněna jedna z následujících podmínek:

- Celá populace se nachází v lokálním extrému dané účelové funkce.
- Populace ztratila diverzibilitu.
- Optimalizační proces probíhá pomalu, nebo vůbec neprobíhá.

Takovému případu říkáme předčasná *konvergence k suboptimálnímu řešení* a lze ho ovlivnit následujícími faktory:

- Nastavením řídících parametrů
- Velikostí populace
- Počtem generací
- Definicí účelové funkce
- Definicí omezení

Stagnací se nazývá jev, při kterém se sice vývoj hodnot účelové funkce zastavil, aniž by ještě byl nalezen globální extrém, ale žádná z výše uvedených podmínek splněna nebyla – tedy populace je dále různorodá, nenachází se v žádném z lokálních extrémů funkce ani nebylo zastaveno vytváření nových potomků. Optimalizační proces tedy neprobíhá bez zřejmých důvodů. [1]

Jednou z možných příčin stagnace je, že veškeré možné kombinace vzniklých jedinců mají horší ohodnocení, nežli jejich rodiče. V takovém případě do další populace postupují opět pouze původní rodiče a nemají šanci se vyvíjet. Riziko stagnace se tedy snižuje se zvyšujícím se počtem řešení (jedinců), které jsme schopni v rámci jedné generace vytvořit. Počet možných řešení evoluce se dá vypočítat. Například pro případ strategie DE/Rand/1/Bin je počet řešení dán rovnicí:

$$n_{trial} = \begin{cases} NP^3 - 3 \cdot NP^2 + 2 \cdot NP & jestli\ CR = 1 \\ D \cdot NP \cdot (NP^3 - 3 \cdot NP^2 + 2 \cdot NP) & jestli\ CR = 0 \\ 2^D \cdot NP \cdot (NP^3 - 3 \cdot NP^2 + 2 \cdot NP) & jinak \end{cases} \quad (8)$$

Další možností boje proti stagnaci je změna řídících parametrů evoluce, tj. kromě již zmíněného počtu jedinců v populace (NP) i mutační faktor (F) a práh křížení (CR). Všechny tyto parametry mají na vznik nových jedinců a jejich kvalitu vliv a jejich změna může vývoj nové generace a tím i optimalizační proces opět obnovit.

Příkladem problematicky zvolených parametrů jsou například nastavení F či CR na hodnotu 1,0. V takových případech pozorujeme zvýšené riziko generování duplikátních řešení jedinců – bývá tedy praktičtější nastavit hodnotu daných parametrů raději na 0,99, nežli na číslo bez desetinného rozvoje. [2]

2.6 Testovací funkce a vyhodnocení účelové funkce

Testovací funkce jsou využívány za účelem demonstrace funkčnosti evolučních algoritmů. U většiny uvedených funkcí známe její průběh i polohu jejích extrémů. Pomocí testovacích funkcí je také možné porovnávat efektivitu jednotlivých evolučních algoritmů. [1]

Samotný výpočet hodnoty účelové funkce pak spočívá v prostém dosazení parametrů jedince do původní účelové funkce.

2.6.1 Galerie testovaných účelových funkcí

Zadání jednotlivých testovacích funkcí se v závislosti na vybraných zdrojích často mění. Veškeré uvedené informace byly otestovány programem popsaným v praktické části a předpisy funkcí i daná maxima by měly tedy být konzistentní. Grafy jednotlivých funkcí byly vytvořeny ve Wolfram Alpha.

Ve výčtu testovacích funkcí uvádím všechny známé globální extrémy funkcí ve formátu $f(x_1, x_2) = y$. Tento zápis vyjadřuje, že extrém dané funkce máme hledat na pozici (x_1, x_2) a že hodnota účelové funkce v tomto bodě nabývá hodnoty y.

První de Jongova funkce (1st De Jong)

Účelová funkce

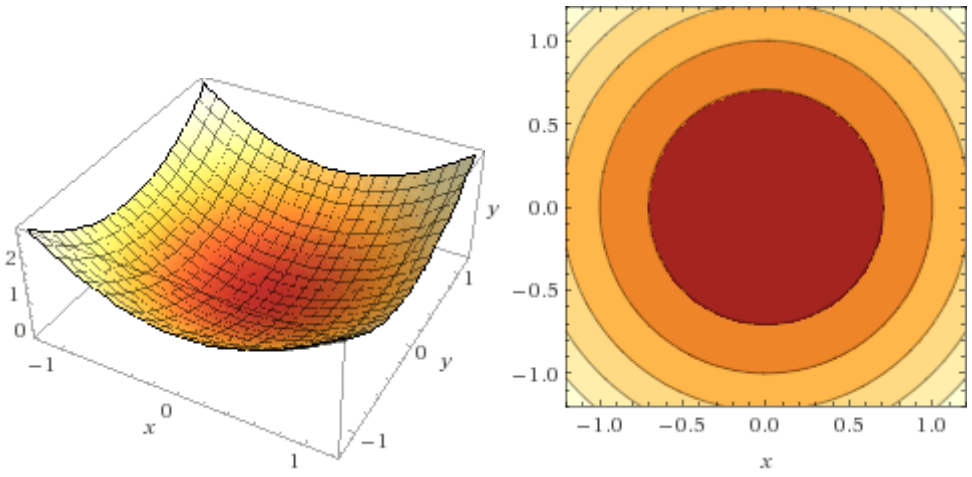
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D} x_i^2 \tag{9}$$

Globální minimum

V $\boldsymbol{E_2}$: $f(0,0) = 0$

V $\boldsymbol{E_n}$: $f(0,0,\dots,0) = 0$

[1]



Obr. 4. První de Jongova funkce

Druhá de Jongova funkce (Rosenbrocks's saddle)

Funkce je též známá pod názvem Rosenbrockovo sedlo nebo Banánová funkce.

Účelová funkce

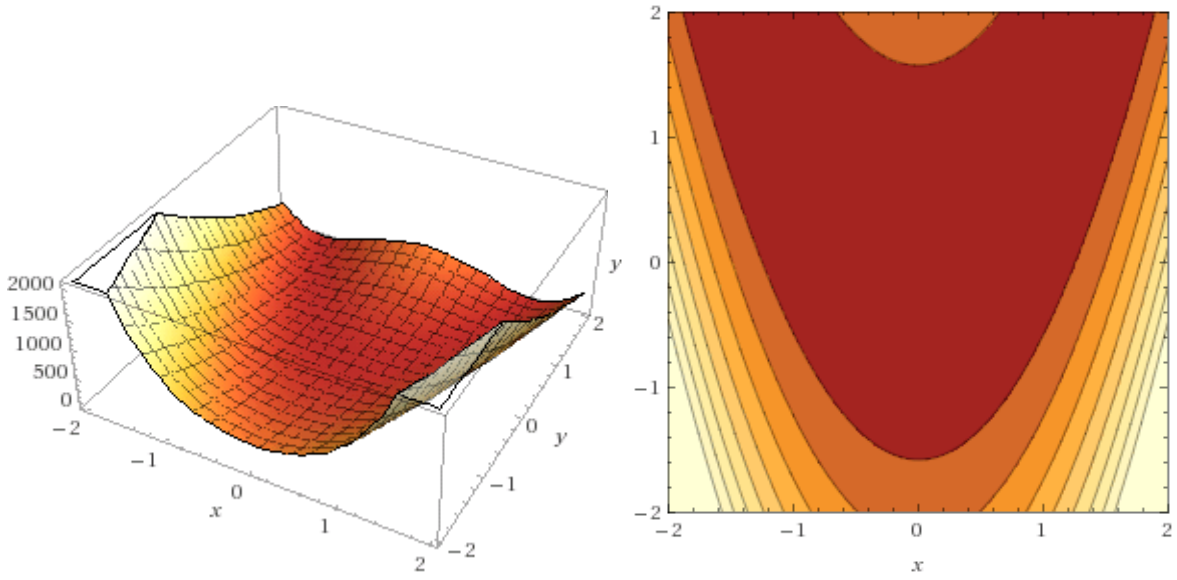
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D-1} 100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \qquad (10)$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(1,1) = 0$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(1,1,\dots,1) = 0$$

[1]



Obr. 5. Druhá de Jongova funkce

Třetí de Jongova funkce (3rd De Jong)

Účelová funkce

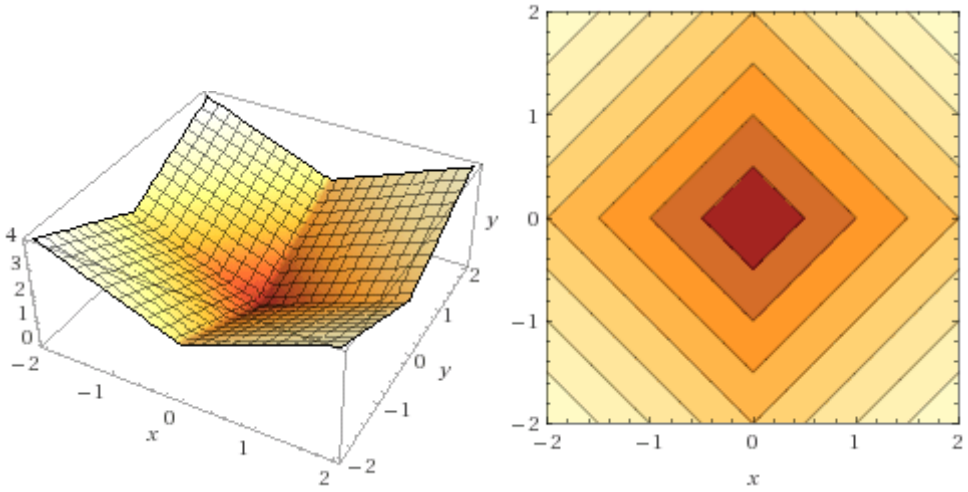
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D} |x_i| \tag{11}$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = 0$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = 0$$

[1]



Obr. 6. Třetí de Jongova funkce

Čtvrtá de Jongova funkce (4th de Jong)

Účelová funkce

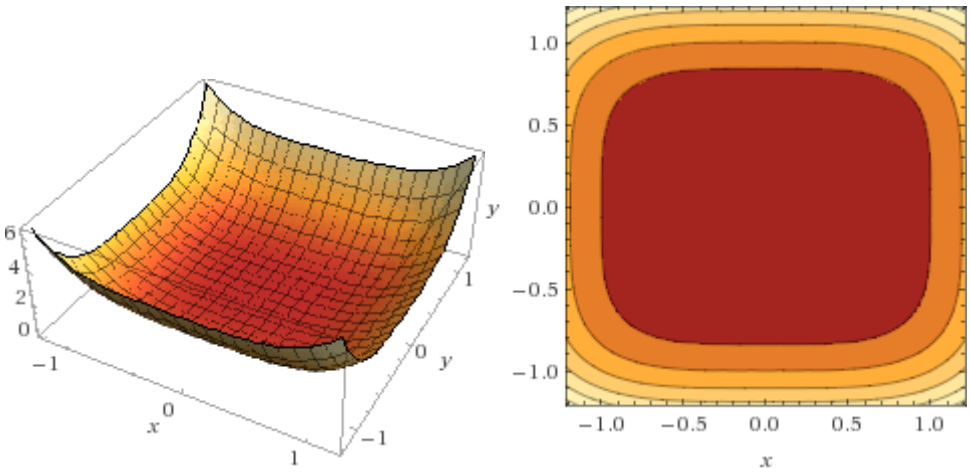
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D} i\, x_i^4 \tag{12}$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = 0$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = 0$$

[1]



Obr. 7. Čtvrtá de Jongova funkce

Rastriginova funkce (Rastrigin's function)

Účelová funkce

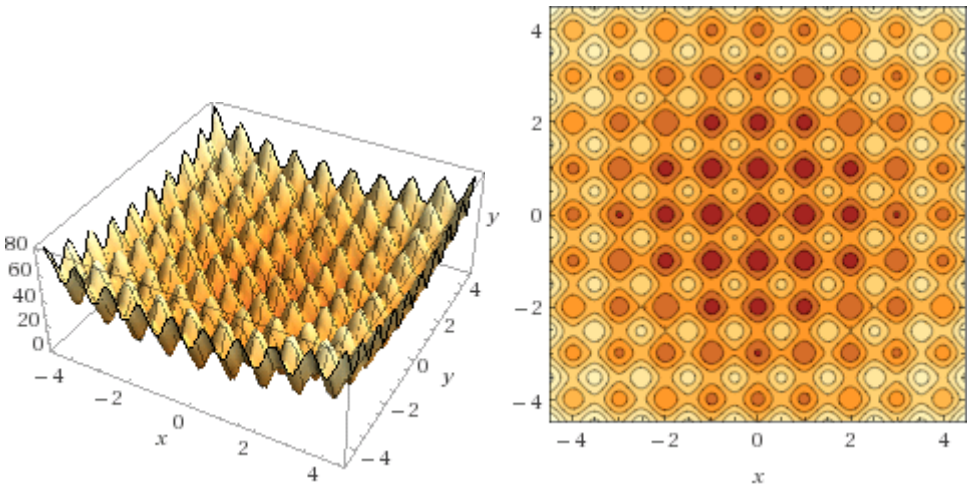
$$f(x_1, x_2, \dots, x_D) = 10D + \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i)) \tag{13}$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = 0$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = 0$$

[6]



Obr. 8. Rastriginova funkce

Schwefelova funkce (Schwefel's function)

Účelová funkce

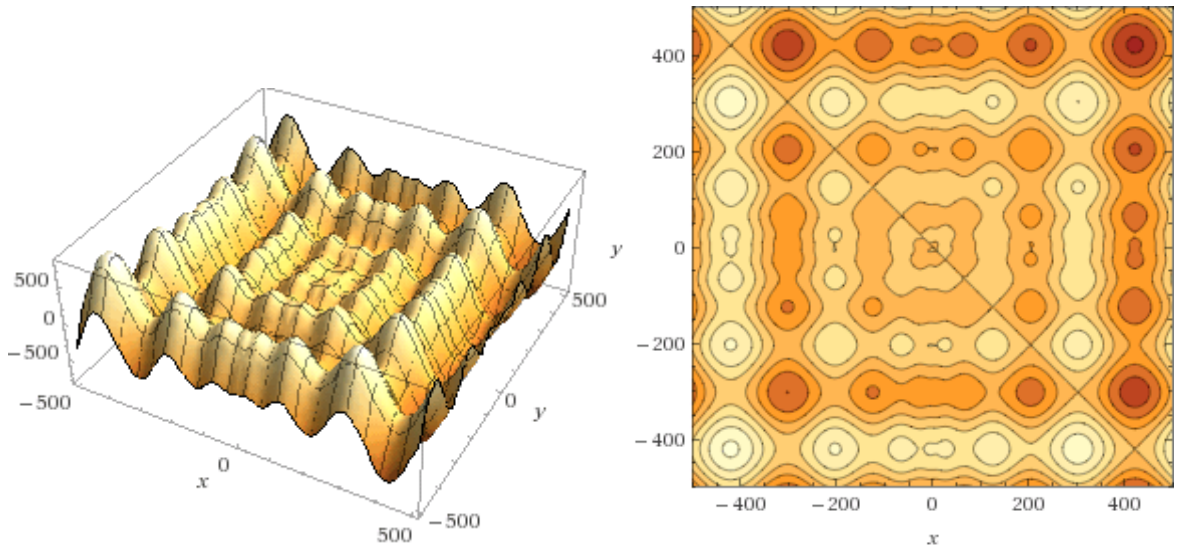
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D} -x_i sin(\sqrt{|x_i|}) \tag{14}$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}\text{: } f(420{,}969; 420{,}969) = -837{,}966$$

$$\mathbf{V}\ \boldsymbol{E_n}\text{: } f(420{,}969; \dots; 420{,}969) = -418{,}983 \cdot n$$

[1]



Obr. 9. Schwefelova funkce

Griewangkova funkce (Griewangk's function

Účelová funkce

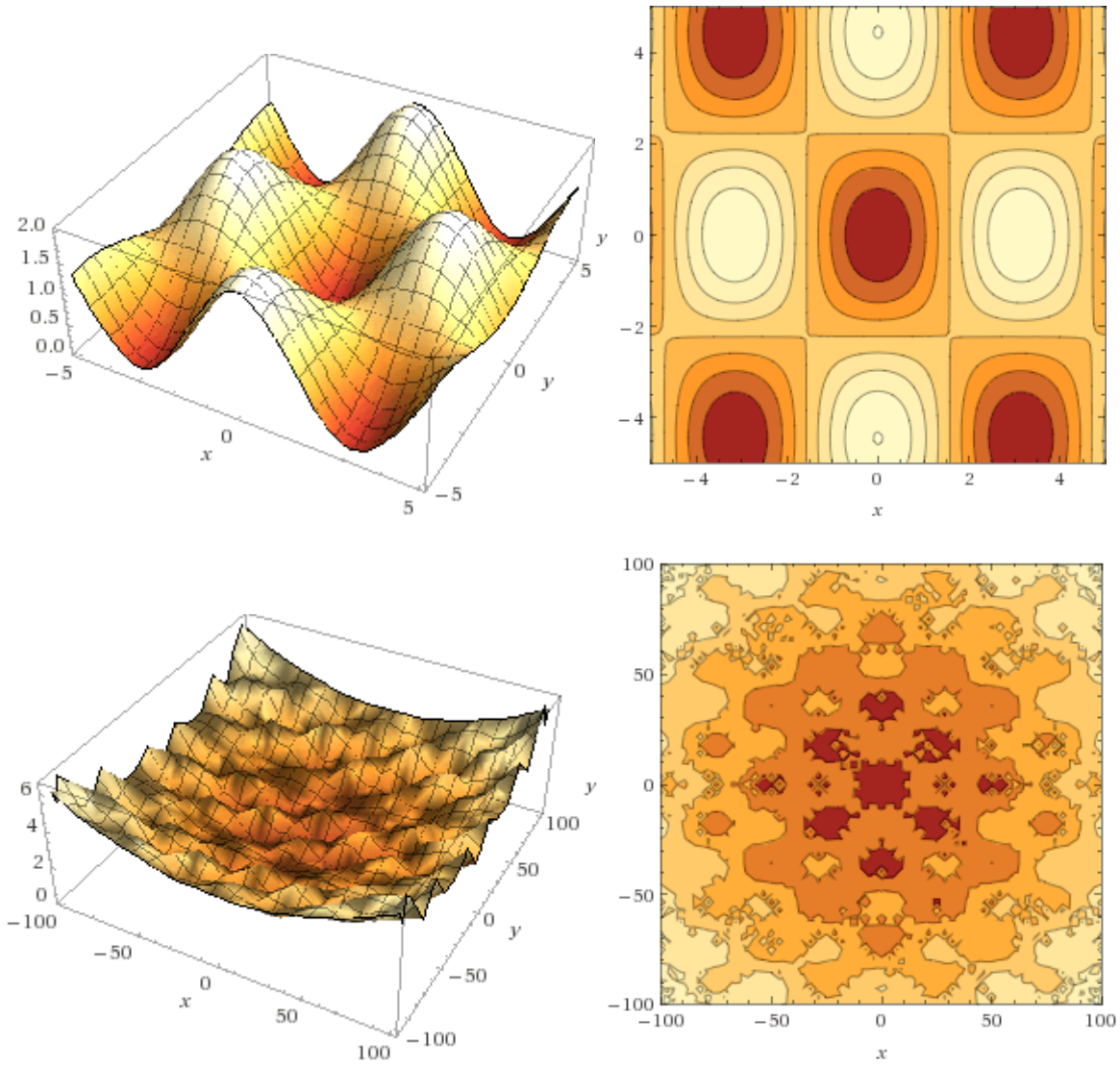
$$f(x_1, x_2, \dots, x_D) = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) \qquad (15)$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = 0$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = 0$$

[1]



Obr. 10. Griewangkova funkce v intervalu [-5; 5] (nahoře) a [-600;600] (dole)

Sinová obálková sinusoidální funkce (Sine envelope wave function)

Účelová funkce

$$f(x_1, x_2, \dots, x_D) = -\sum_{i=1}^{D-1} 0{,}5 + \frac{\sin\left(\sqrt{x_i^2 + x_{i+1}^2} - 0{,}5\right)^2}{\left(1 + 0{,}001(x_i^2 + x_{i+1}^2)\right)^2} \qquad (16)$$

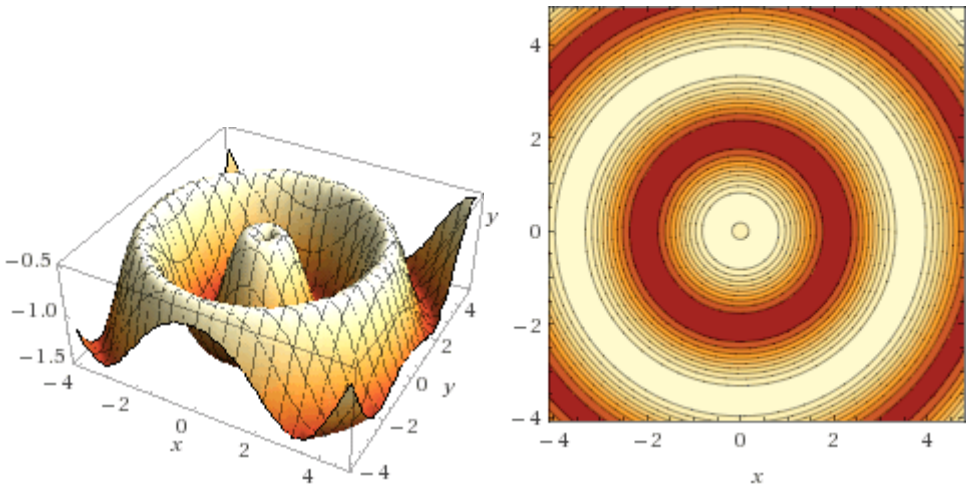
Globální minimum

V $\boldsymbol{E_2}$: Globální extrém najdeme tam, kde je poloměr kružnice

$r = \sqrt{x_i^2 + x_{i+1}^2} = \pm 2{,}06668$ a funkční hodnota $y = -1{,}491$

V $\boldsymbol{E_n}$: kružnice s funkční hodnotou $y = -1{,}491 \cdot (n-1)$

[6]



Obr. 11. Sinová obálková sinusoidální funkce

Roztažená sinusoidální V funkce (Stretched V sine wave function)

Účelová funkce

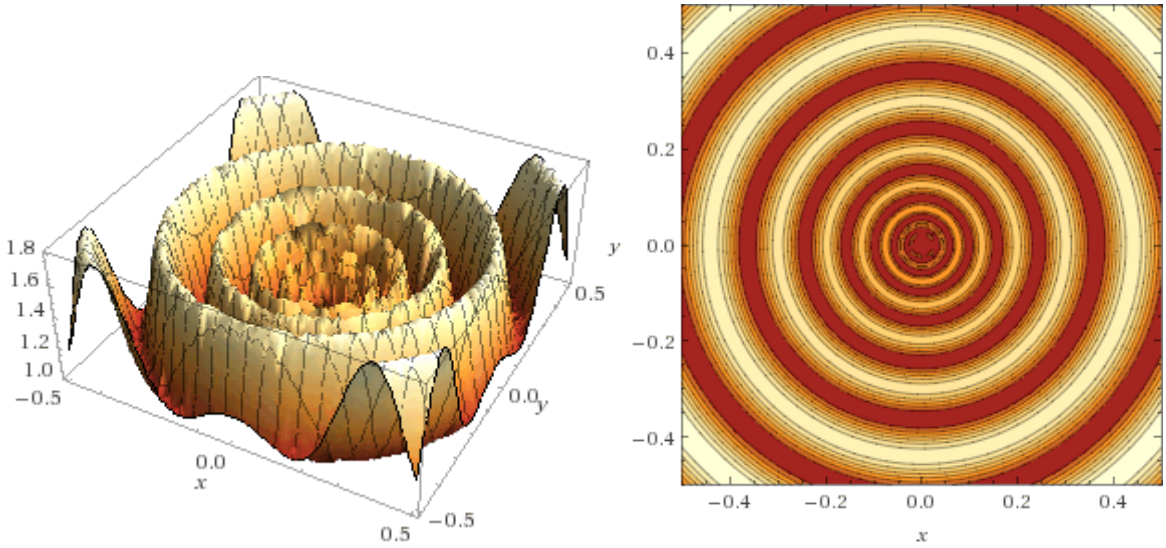
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D-1} \left(\sqrt[4]{x_i^2 + x_{i+1}^2} \sin\left(50 \sqrt[10]{x_i^2 + x_{i+1}^2} \right)^2 + 1 \right) \quad (17)$$

Globální minimum

$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = 0$

$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = 0$

[1]



Obr. 12. Roztažená sinusoidální V funkce

Ackleyho funkce I (Ackley's function I)

Účelová funkce

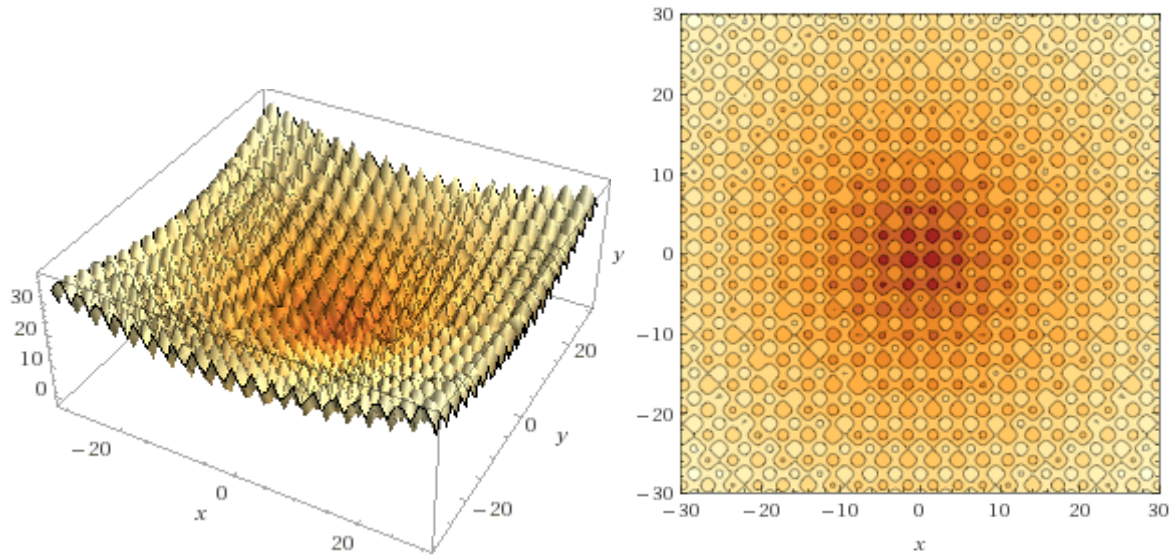
$$f(x_1, x_2, \dots, x_D) = \sum_{i=1}^{D-1} \left(\frac{\sqrt{x_i^2 + x_{i+1}^2}}{e^{\frac{1}{5}}} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right) \tag{18}$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(\pm 1{,}50236;\ -0{,}754865) = -4{,}4901$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(1{,}51563;\ -1{,}1151;\ -1{,}10972; \dots;\ -1{,}10972; -0{,}747245) = -7{,}54276 - 2{,}91867 \cdot (\mathrm{n} - 3)$$

[6]



Obr. 13. Ackleyho funkce I

Ackley's function II – Ackleyho funkce II

Účelová funkce

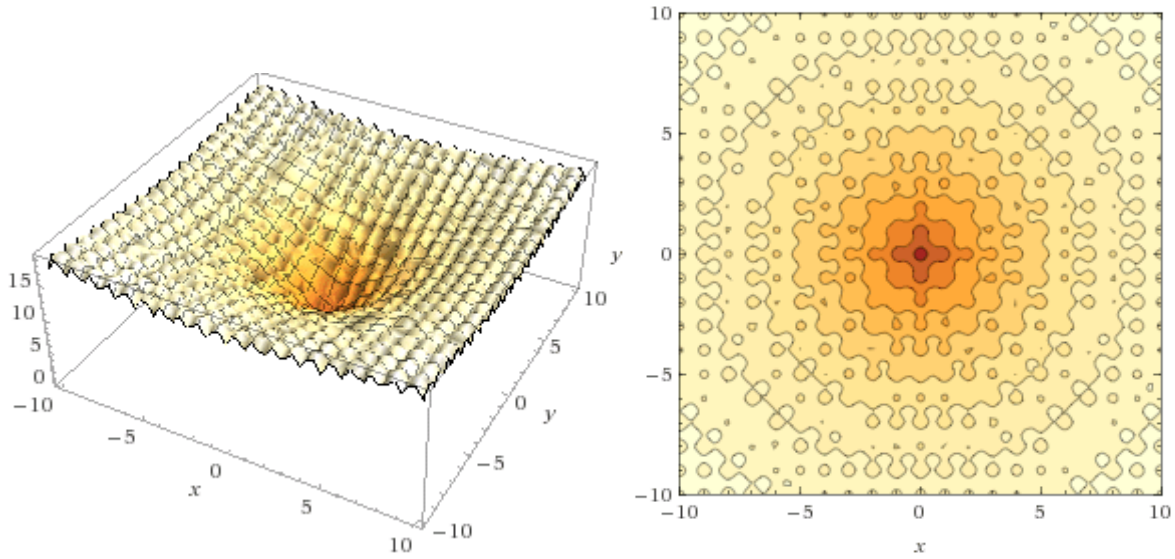
$$f(x_1, \dots, x_D) = \sum_{i=1}^{D-1} \left(20 + e - e^{\frac{\cos(2\pi x_i) + \cos(s\pi x_{i+1})}{2}} - 20 \cdot (e^{-0{,}2 \cdot \sqrt{\frac{x_i^2 + x_{i+1}^2}{2}}}) \right) \qquad (19)$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = 0$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = 0$$

[1]



Obr. 14. Ackleyho funkce II

Michalewiczova funkce (Michalewicz's function)

Účelová funkce

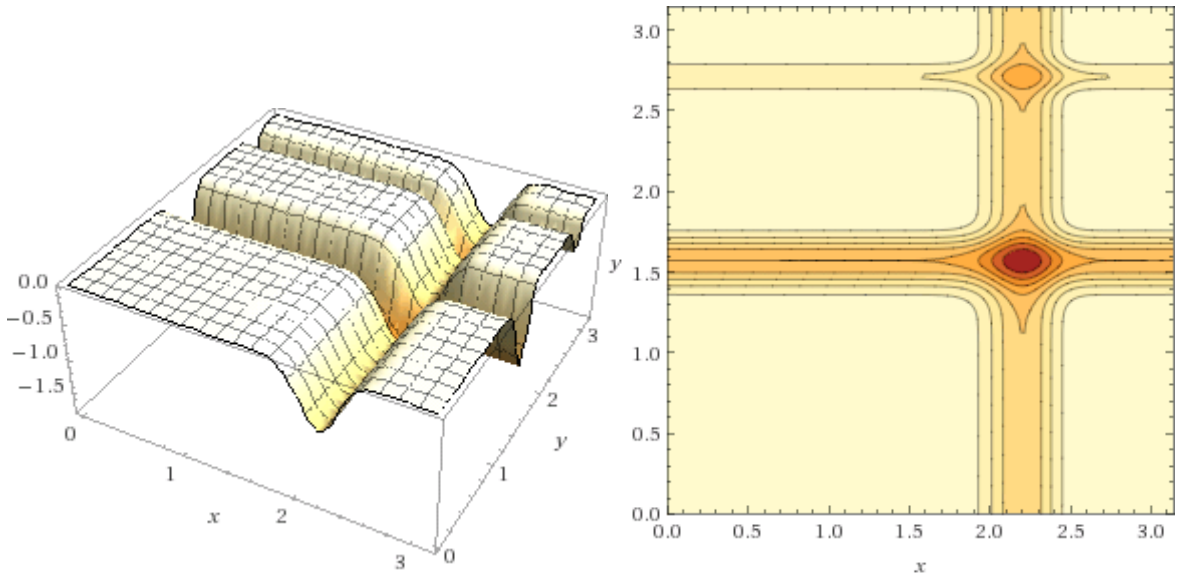
$$f(x_1, x_2, \dots, x_D) = -\sum_{i=1}^{D} \sin(x_i) \sin^{20}\left(\frac{ix_i^2}{\pi}\right) \tag{20}$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(2{,}20291;\ 1{,}57096) = -1{,}8013$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(2{,}2091, 1{,}57104, \dots, 1{,}57104) = 1{,}00098 \cdot (n-2)$$

[7]



Obr. 15. Michalewiczova funkce

Mastersova funkce (Master's cosine wave function)

Účelová funkce

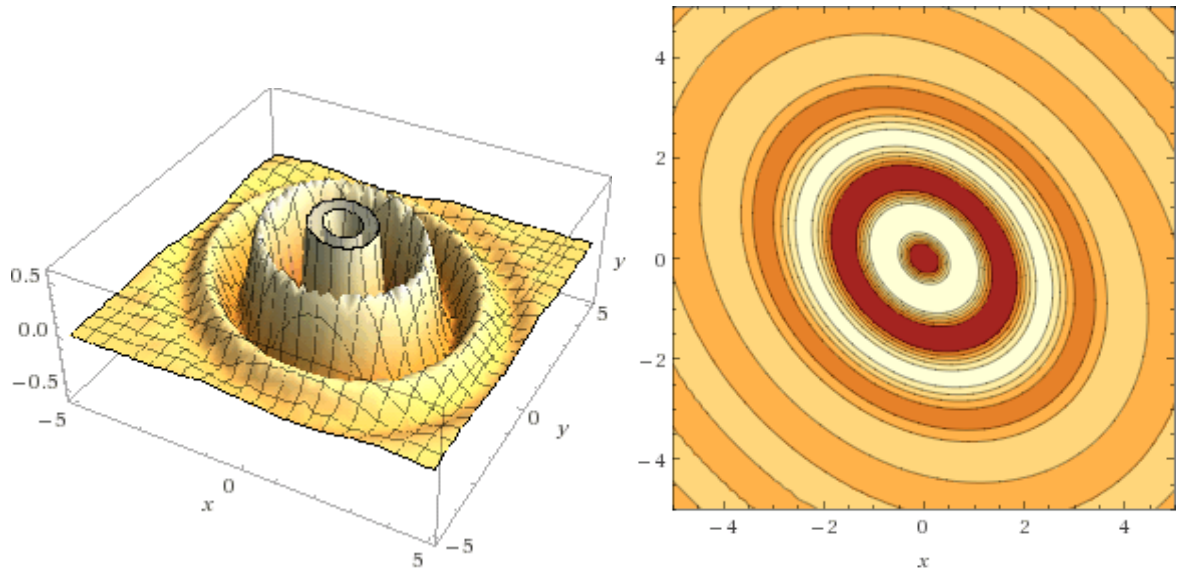
$$f(x_1, \dots, x_D) = -\sum_{i=1}^{D-1}\left(e^{\frac{-(x_i^2+x_{i+1}^2+0{,}5x_ix_{i+1})}{8}}\cos(4\sqrt{x_i^2+x_{i+1}^2+0{,}5x_ix_{i+1}})\right) \qquad (21)$$

Globální minimum

$$\mathbf{V}\ \boldsymbol{E_2}: f(0,0) = -1$$

$$\mathbf{V}\ \boldsymbol{E_n}: f(0,0,\dots,0) = -1 \cdot n$$

[6]



Obr. 16. Masterova funkce

3 DETERMINISTICKÝ CHAOS

3.1 Základní pojmy deterministického chaosu

Zatímco chaos vnímáme jako projev absolutní, čisté náhody, chování systémů ve skutečné přírodě je charakterizováno termínem *deterministický chaos*. Mezi jeho typické znaky patří:

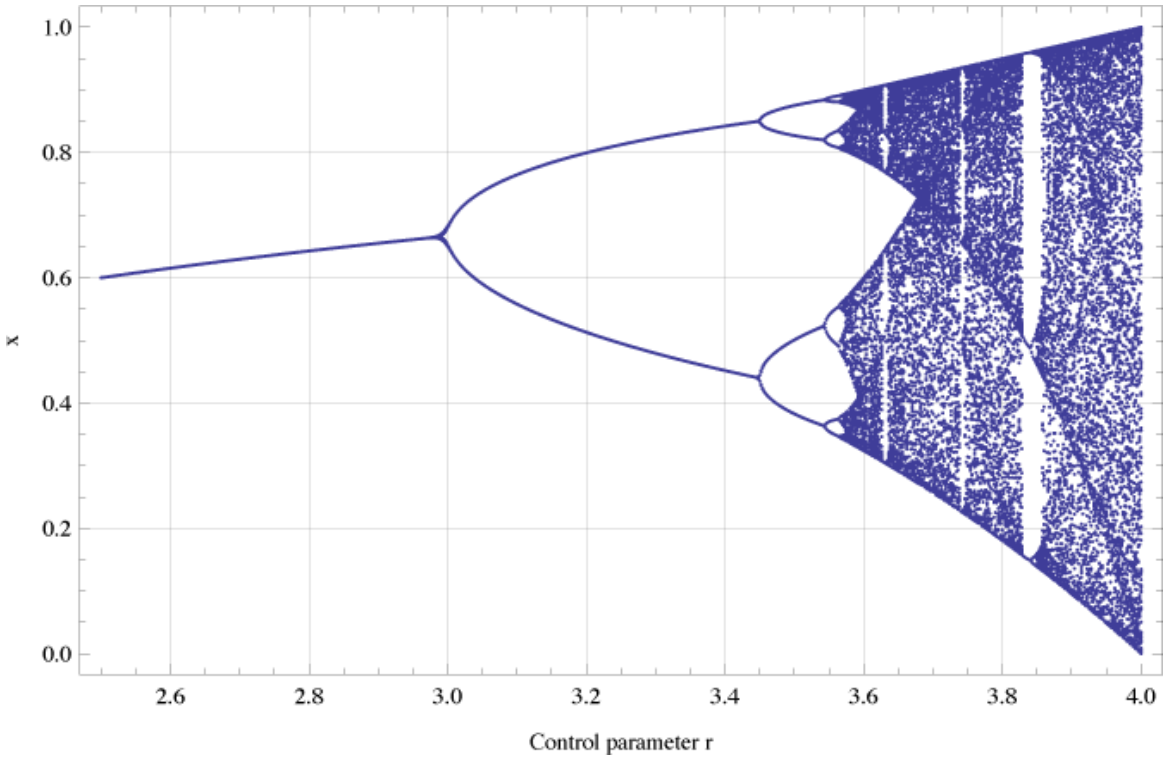
- velká citlivost i na extrémně malé změny počátečních podmínek nebo jiných parametrů
- dynamický systém je globálně stabilní, ale lokálně nestabilní a nepředvídatelný
- chyby v měření počátečních podmínek exponenciálně narůstají
- chaotické systémy jsou vždy nelineární a to i přesto, že je lze matematicky popsat

[9]

3.1.1 Bifurkace

Víme, že v systémech deterministického chaosu se dva nekonečně blízké stavy exponenciálně rozcházejí. Tento jev se nazývá *bifurkací*. Jedná se o náhlou změnu stability, která chaosu předchází. V případě, kdy v soustavě dochází ke vzniku na sebe navazujících bifurkací, tedy říkáme, že nastává chaos. Z tohoto důvodu je studium podmínek vzniku bifurkace důležitou součástí analýz nelineárních dynamických soustav. [12]

Grafické znázornění bifurkace se nazývá bifurkačním diagramem. Znázorňuje závislost limitních stavů systému na jednom z řídících parametrů. Typický bifurkační diagram je na obrázku (Obr. 17).



Obr. 17. Bifurkační diagram

3.2 Modely deterministického chaosu

Modely deterministického chaosu nebo také deterministické chaotické systémy jsou systémy, u nichž k určení jejich chování a celkové posloupnosti jejich stavů potřebujeme znát veškeré následující informace:

- Časově závislou rovnici
- Hodnoty všech řídících parametrů
- Počáteční podmínky

[10]

Jako model deterministického chaosu nám tedy poslouží soustava diferenciálních rovnic, u nichž známe všechny výše uvedené body a které se budou značit chaotickým chováním. V bakalářské práci [11], na kterou v praktické části navazuji, jsou uvedeny chaotické systémy pro účely generování pseudo-náhodných čísel, které jsem v programu použila a uvádím je v tabulce (Tab. 6).

Tab. 6. Použité chaotické systémy [11]

Název systému	Matematický zápis	Parametry	
Lozi	$X_{n+1} = 1 + a\|X_n\| + bY_n$ $Y_{n+1} = X_n$	$a = 1{,}7$ $b = 0{,}5$	(22)
Sinai	$X_{n+1} = X_n + Y_n + \delta cos2\pi Y_n (mod\ 1)$ $Y_{n+1} = X_n + 2Y_n (mod\ 1)$	$\delta = 0{,}1$	(23)
Disipative	$X_{n+1} = X_n + Y_{n+1} (mod\ 2\pi)$ $Y_{n+1} = bY_n + ksinX_n (mod\ 2\pi)$	$b = 0{,}1$ $k = 8{,}8$	(24)
Ikeda	$X_{n+1} = \gamma + \mu(X_n cos\omega - Y_n sin\omega)$ $Y_{n+1} = \mu(X_n sin\omega + Y_n cos\omega)$ $\omega = \beta - \frac{\alpha}{1 + X_n^2 + Y_n^2}$	$\alpha = 6$ $\beta = 0{,}4$ $\gamma = 1$ $\mu = 0{,}9$	(25)
Arnold cat	$X_{n+1} = X_n + Y_n (mod\ 1)$ $Y_{n+1} = X_n + kY_n (mod\ 1)$	$k = 2$	(26)
Hénon map	$X_{n+1} = 1 - aX_n^2 + bY_n$ $Y_{n+1} = X_n$	$a = 1{,}4$ $b = 0{,}3$	(27)
Burgers map	$X_{n+1} = aX_n - Y_n^2$ $Y_{n+1} = bY_n + X_n Y_n$	$a = 0{,}75$ $b = 1{,}75$	(28)
Holmes cubic map	$X_{n+1} = Y_n$ $Y_{n+1} = -bX_n + dY_n - Y_n^3$	$b = 0{,}2$ $d = 2{,}77$	(29)
Chirikov	$X_{n+1} = X_n + Y_{n+1} (mod\ 2\pi)$ $Y_{n+1} = Y_n + ksinX_n (mod\ 2\pi)$	$k = 1$	(30)

3.3 Diferenciální evoluce s prvky deterministického chaosu

Funkčnost mnoha algoritmů – evoluční algoritmy nevyjímaje je založena na generování náhodných čísel. Zabudovaný generátor čísel v programovacím jazyce C/C++ se nachází v knihovně stdlib.h: konkrétně se jedná o funkci inicializující posloupnost čísel *srand()* a funkci *rand()*, která vrací číslo v rozmezí od 0 do RAND_MAX (její výsledek tedy musí být ještě nějak upraven). O vygenerované hodnotě pak ale nemluvíme jako o náhodné, ale pseudo-náhodné, jelikož inicializovaná posloupnost, ze které vycházíme, je čistě deterministická – tzn. že při stejných parametrech nám bude vždy vygenerována stejná posloupnost čísel. [8]

Mluvíme-li o diferenciální evoluci s prvky deterministického chaosu v C/C++, máme na mysli evoluci, kde místo generování čísel pomocí funkce *rand()* používáme generátor založený na chaotických systémech. Ty se projevují nepravidelným a zdánlivě nepředvídatelným chováním a mají s generátory pseudo-náhodných čísel leccos společného. [8]

II. PRAKTICKÁ ČÁST

4 OBJEKTOVÉ ZPRACOVÁNÍ DIFERENCIÁLNÍ EVOLUCE S PRVKY DETERMINISTICKÉHO CHAOSU

4.1 Třída Chaos

Třída Chaos využívaná pro generování pseudo-náhodných čísel vychází z bakalářské práce Radovana Fuchse Deterministický chaos jako generátor náhodných čísel v prostředí C a C++ [11], na kterou navazuji.

Základní proměnné

double polex[MAX_DELKA]	Pole, do kterého generujeme výsledná chaotická desetinná čísla
double b0, double k0, …	Parametry jednotlivých chaotických modelů
string system	Název vybraného chaotického systému
double maximum1	Dosavadní vygenerované maximum (využíváme pro správné rozdělení hodnot do intervalu (0, 1))

Základní metody

Chaos()	Konstruktor nastavuje základní parametry
void Lozi(), void Sinai, void Disipative(), void ArnoldCat(), void HolmesCubic(), void Henon(), void Ikeda(), void Chirikov	Funkce pro modelování chaotických systémů
void max()	Projde pole vygenerovaných čísel a najde maximum
void vypis()	Uloží výsledek do souboru
void vyber()	Nastaví výběr chaotického systému

4.2 Třída DE

Třída DE je základní třídou implementace diferenciálního chaosu.

Základní proměnné

double * population	Populace je určena polem desetinných čísel
double * newPop	Nově vznikající populace
double * noise	Šumový vektor (výsledek mutace)
double * trial	Zkušební vektor (výsledek křížení)
double * best	Nejlepší jedinec generace

Základní parametry diferenciální evoluce

double crossover	Práh křížení
int popSize	Velikost populace
double factor	Mutační konstanta
int generations	Počet evolučních cyklů
int dimension	Počet argumentů účelové funkce

Testovací pomocné proměnné

int testFunction	Vybraná testovací funkce
double minimum, double maximum	Meze testovací funkce
int stagnation	Počet stagnujících generací
int maxStagnation	Maximální počet stagnujících generací

Základní metody

DE	Konstruktor. Nastavuje základní parametry jako CR, NP, F, D, počet generací a další
void init	Nastavuje chaotický generátor čísel, meze účelové funkce a naplní úvodní populaci náhodnými daty pomocí metody void setPopulation

double chaosRand	Generuje pseudo-náhodné číslo za pomoci instance třídy Chaos
void best1, void rand1, void randToBest1, void best2, void rand2	Jednotlivé mutační strategie
void selectParents	Náhodný výběr rodičů během mutace
void binCross, void expCross	Binární křížení, exponenciální křížení
void evolve	Evoluční metoda. Závisí na vybraných mutačních a křížících strategiích.
double costFunction	Účelová funkce
void getPopulation	Vypíše nejlepšího jedince populace.

4.2.1 Vlastní účelová funkce

Účelová funkce *double costFunction()* vrací hodnotu funkce jedince. Matematická operace vykonaná s jednotlivými parametry jedince závisí na vybrané testovací funkci. Program je také možné rozšířit o svou vlastní novou účelovou funkci. V takovém případě se žádaná funkce zapíše do *costFunction()* jako nová podmínka ve tvaru (Obr. 18)

```
if (testFunction==A){

    for (int i=1; i<=dimension; i++){
        double x1=testPopulation[index+i*popSiz];          //parametr i

        costValue+=//zapis nove funkce patri sem
    }
}
```

Obr. 18. Implementace nové sumační účelové funkce s i-tým parametrem

Kde A je číslo reprezentující novou účelovou funkci v konfiguračním souboru. Pokud by nová funkce nepočítala pouze s jedním parametrem ale i s parametrem následujícím, použila by se podmínka uvedená v obrázku (Obr. 19).

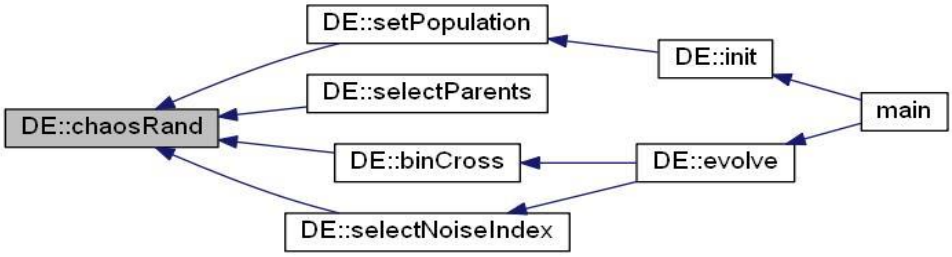
```
if (testFunction==B){
    for (int i=1; i<dimension; i++){
        double x1=testPopulation[index+i*popSiz];          //parametr i
        double x2=testPopulation[index+(i+1)*popSiz];      //parametr i+1

        costValue+=//zapis nove funkce patri sem
    }
}
```

Obr. 19. Implementace nové sumační účelové funkce se dvěma navazujícími parametry

4.2.2 Pseudo-náhodný generátor čísel s pomocí deterministického chaosu

Chaotický generátor čísel v metodě *chaosRand()* využívá třída diferenciální evoluce hned v několika metodách. Náhodně generované hodnoty potřebujeme při nastavování úvodní populace *setPopulation()*, při výběru rodičů pro mutaci *selectParents()*, binárním křížení *binCross()* a exponenciálním křížení, přesněji řečeno při generování náhodné hodnoty porovnávané s CR, o což se stará metoda *selectNoiseIndex()*. Bereme-li to z funkce main, používáme chaotické systémy jak při inicializaci nové populace, tak i v každém evolučním cyklu (Obr. 20).

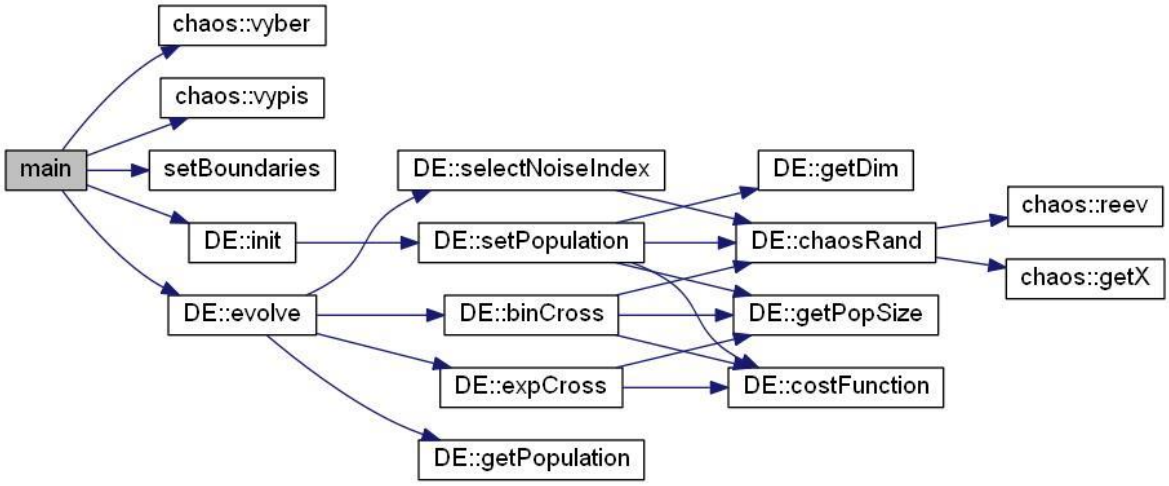


Obr. 20. Využití chaotického generování čísel

4.3 Implementace programu ve funkci main

Při implementaci funkce *main()* nejprve založíme objekt třídy Chaos, podle vybraného modelu vygenerujeme chaos a určíme účelovou testovací funkci s jejími určenými omezeními.

Pak založíme instanci třídy DE a nastavíme její základní parametry a úvodní generaci. Následuje zavolání metody *evolve()*, která spustí samotnou evoluci. Na konci programu zavoláme destruktor třídy DE. Obr. 21 zobrazuje graf volání jednotlivých metod.



Obr. 21. Call diagram funkce main

5 POPIS OVLÁDÁNÍ PROGRAMU

Základní nastavení a parametry evoluce se nastaví v souboru ***config.txt*** (Obr. 22), který najdete v hlavním adresáři, kde se nachází také soubor ***main.cpp***. Veškeré parametry nastavitelné v konfiguračním souboru jsou zadány číselně. V případě desetinných parametrů (CR a F) oddělujeme desetinný rozvoj tečkou. Kvůli parametrům, u kterých není zřejmý číselný význam, je přímo v konfiguračním souboru uvedena převodní tabulka (Obr. 23).

```
Konfigurační soubor

PARAMETRY PRO TESTOVÁNÍ
~~~~~~~~~~~~~~~~~~~~~~~~~~
Testovaci funkce = 1
Mutacni strategie = 1
Strategie krizeni = 1
Minimum = -100
Maximum = 100

PARAMETRY DIFERENCIÁLNÍ EVOLUCE
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
CR = 0.8
NP = 15
F = 0.8
generace = 10
D = 2

PARAMETRY PRO CHAOTICKY GENERATOR CISEL
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Chaoticky system = 9
```

Obr. 22. Ukázka konfiguračního souboru config.txt

Pokud tedy například chcete použít algoritmus diferenciální evoluce na Rastriginovu testovací funkci při použití strategie Rand/1/Bin v intervalu [-10; 10], zadáte do konfiguračního souboru parametry:

- Testovaci funkce = 4
- Mutacni strategie = 1
- Strategie krizeni = 1
- Minimum = -10
- Maximum = 10

Program po spuštění vypíše kontrolu, zda se podařilo konfigurační soubor spustit a jaké z něj načetl parametry. V případě, že se soubor *config.txt* nepodaří spustit, bude počítat program evoluci s výchozími hodnotami, které se dají nastavit přímo v kódu funkce *main()*.

```
/*****************************************************************************
 *                                                                           *
 *   PŘEVODNÍ TABULKA PARAMETRICKÝCH HODNOT                                  *
 *                                                                           *
 * - V horním nastavení vyplňte číslo odpovídající dané strategii či funkci  *
 *                                                                           *
 * 1. Testovací funkce                                                       *
 * 1stDeJong    0                                                            *
 * 2ndDeJong    1                                                            *
 * 3rdDeJong    2                                                            *
 * 4thDeJong    3                                                            *
 * Rastrigin    4                                                            *
 * Schwefel     5                                                            *
 * Griewangk    6                                                            *
 * SinEnvelope  7                                                            *
 * StretchVSin  8                                                            *
 * Ackley       9                                                            *
 * ndAckley     10                                                           *
 * Eggplate     11                                                           *
 * Rana         12                                                           *
 * Patological  13                                                           *
 * Michalewicz  14                                                           *
 * Masters      15                                                           *
 *                                                                           *
 *                                                                           *
 * 2. Mutační strategie                                                      *
 * Best1                0                                                    *
 * Rand1                1                                                    *
 * RandToBest1          2                                                    *
 * Best2                3                                                    *
 * Rand2                4                                                    *
 *                                                                           *
 *                                                                           *
 * 3. Strategie křížení                                                      *
 * Exp                  0                                                    *
 * Bin                  1                                                    *
 *                                                                           *
 * 4. Chaotický system                                                       *
 * Lozi         1                                                            *
 * Disipative   2                                                            *
 * Sinai        3                                                            *
 * Ikeda        4                                                            *
 * ArnoldCat    5                                                            *
 * Henon        6                                                            *
 * Burgers      7                                                            *
 * HolmesCubic  8                                                            *
 * Chirikov     9                                                            *
 *                                                                           *
 * 5. Omezení intervalu hledání                                              *
 * defaultní omezení intervalu hledání          x                            *
 *                                                                           *
 *****************************************************************************/
```

Obr. 23. Parametrická převodní tabulka v konfiguračním souboru

Následuje spuštění chaotického generátoru čísel podle vybraného systému. Výstupní hodnoty generátoru jsou jednak uloženy v poli desetinných čísel, ale i do výstupního souboru *chaos.txt*, takže si uživatel může zkontrolovat, zda se jedná skutečně o chaotický systém.

Posledním krokem programu je samotná diferenciální evoluce. Program vypisuje vždy nejlepšího jedince generace, dokud nedojde k maximální zadané generaci, či nezačne evoluce stagnovat po předem určený počet generací. Výsledek programu je v tomto tvaru:

$$Generace\ x\ !\ CV\ |\ 1.parametr\ |\ \ldots|D-t\acute{y}\ parametr,$$

x… pořadí generace

CV… „cost value“ – hodnota účelové funkce (31)

Obrázek (Obr. 24) ukazuje možný výstup programu, pokud dojde ke stagnaci.

```
Generace 813 !0 | -8.55937e-011 | 5.92437e-010 |
Generace 814 !0 | -8.55937e-011 | 5.92437e-010 |
Generace 815 !0 | -8.55937e-011 | 5.92437e-010 |
Generace 816 !0 | -8.55937e-011 | 5.92437e-010 |
Generace 817 !0 | -8.55937e-011 | 5.92437e-010 |
Generace 818 !0 | -8.55937e-011 | 5.92437e-010 |
Generace 819 !0 | -8.55937e-011 | 5.92437e-010 |

Evoluce stagnuje, bude proto ukoncena a to po 666 nezlepsujicich se generacich.

Process returned 0 (0x0)   execution time : 1.047 s
Press any key to continue.
```

Obr. 24. Výstup programu se stagnací

5.1 Výsledky testování algoritmu DE

Výstup programu většinou není z důvodu náhodnosti statický (při opakovaném spuštění programu dostáváme pokaždé jiný výsledek), proto jsou uvedené výsledky pouze orientační. Kvalita absolutního výsledku evoluce závisí na zvolených parametrech ale i na vybraném chaotickém systému. V rámci programu jsem otestovala uvedené účelové funkce a výsledky porovnala se známými výsledky uvedených testovacích funkcí v dvojrozměrném prostoru. Tab. 7 ukazuje průměrnou úspěšnost několika běhů programu při vyhledávání extrémů s parametry: {mutační strategie = 1; strategie křížení = 1; CR = 0.5; NP = 15; F = 0.8; generace = 2000; D = 2}.

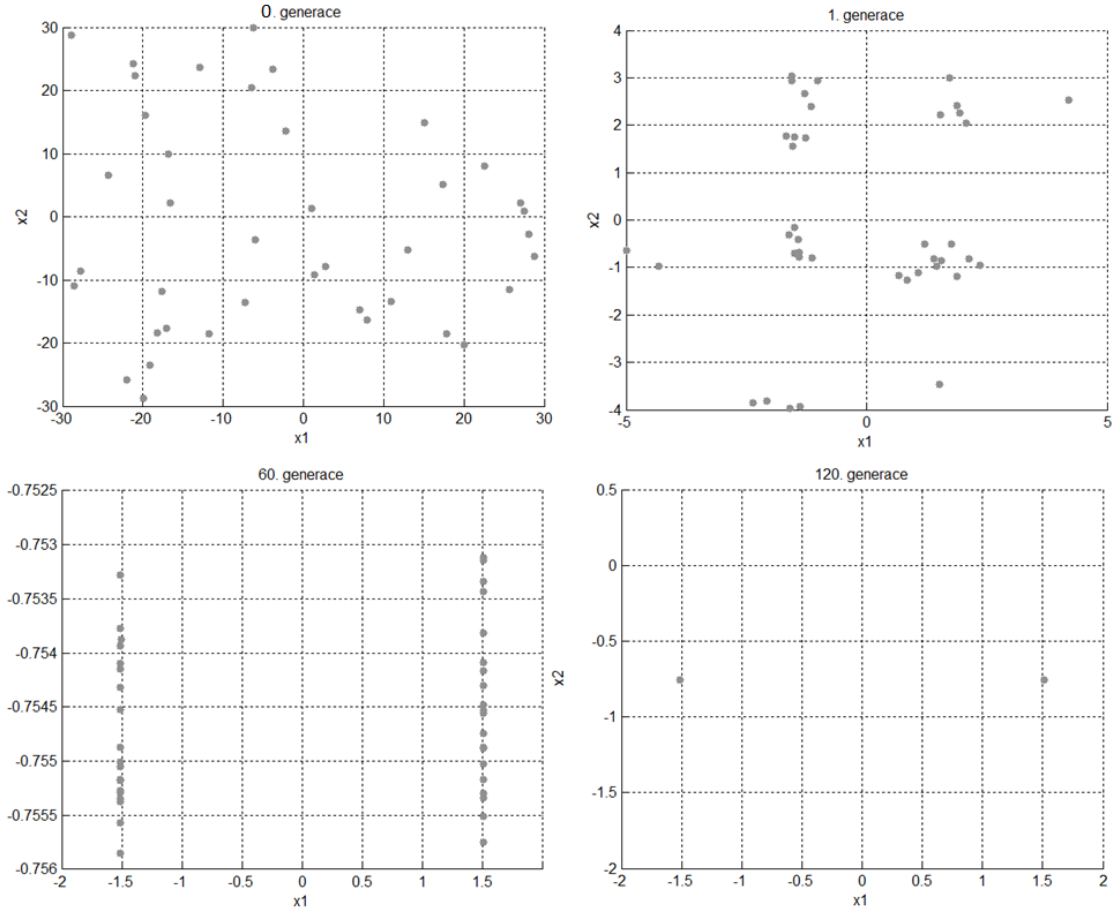
Procentuální odchylku jsem počítala jako aritmetický průměr absolutních hodnot rozdílu CV a odpovídajících parametrů. Výsledná odchylka uvedená v tabulce (Tab. 7) tak může nabývat nenulové hodnoty i přesto, že vypočítaná CV se od známé hodnoty extrému neliší.

$$\frac{|CV_z - CV_{DE}| + |par1_z - par1_{DE}| + |par2_z - par2_{DE}|}{3} \tag{32}$$

Tab. 7. Nalezené extrémy testovacích funkcí

Testovací funkce	**Známý extrém funkce**	**Vyhledaný extrém funkce**	**Procentuální odchylka**
1st De Jong	0	1,2856E-263	$1{,}3502 \cdot 10^{-132}$ %
2nd De Jong	0	0	0 %
3rd De Jong	0	-1,7781E-140	$1{,}1854 \cdot 10^{-140}$ %
4th De Jong	0	0	$6{,}8149 \cdot 10^{-82}$ %
Rastrigin's function	0	0	$3{,}07071 \cdot 10^{-10}$ %
Schwefel's function	-837,966	-837,966	0 %
Griewangk's function	0	0	$5{,}03857 \cdot 10^{-09}$ %
Sine envelope sine wave function	-1,491	-1,4915	0,00048432 %
Stretched V sine wave function	0	-6,27119E-64	$2{,}0904 \cdot 10^{-64}$ %
Ackley's function I	-4,4901	-4,5901	0,035753333 %
Ackley's function II	0	0	$4{,}03701 \cdot 10^{-20}$ %
Michalewicz's function	-1,8013	-1,8013	$5{,}33333 \cdot 10^{-05}$ %
Masters's cosine wave function	-1	-1	$8{,}98149 \cdot 10^{-10}$ %

Z tabulky vyplývá, že úspěšnost diferenciální evoluce je při daných parametrech velmi vysoká a odchylky od nalezeného výsledku extrémně nízké.



Obr. 25. Vývoj populace ve vybraných populacích první Ackleyho testovací funkce

5.1.1 Závislost na parametrech

Pro testování závislosti průběhu DE na parametrech byla vybrána druhá de Jongova funkce, neboť má zpravidla nulovou odchylku při hledání extrému. Jednotlivý výběr parametrů je hodnocen nejprve na základě toho, zda vůbec algoritmus daný extrém nalezne (Tab. 8) a pak rychlostí nalezení správného výsledku (Tab. 9). Ukazatelem rychlosti je *první generace, ve které je nalezen známý globální extrém*. Protože každý běh programu je jiný, je při vyhodnocování obou tabulek počítán průměr z deseti různých evolucí. Proměnné parametry jsou CR a F, ostatní parametry jsou *{NP = 15; generace = 5000; D = 2}*. V průběhu programu je aplikována detekce stagnace.

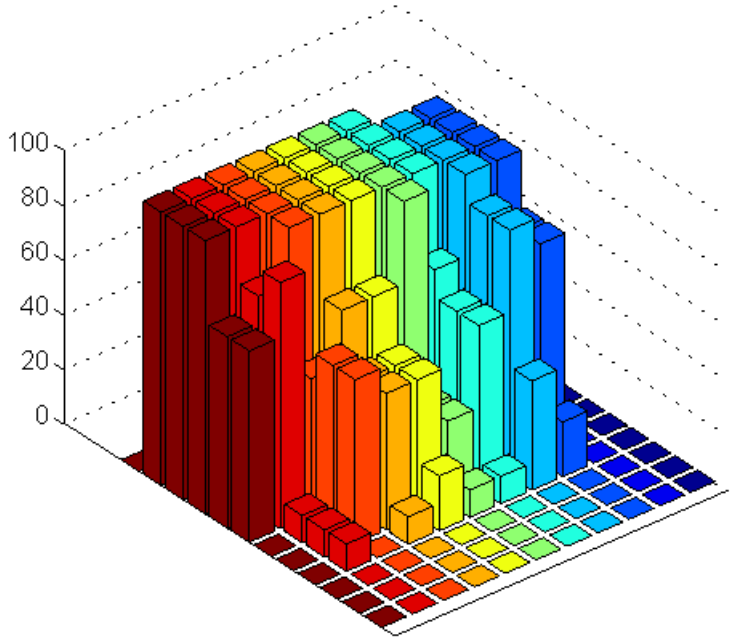
Tab. 8. Úspěšnost nalezení známého extrému

	CR										
F	**0**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
0	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
0,1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
0,2	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
0,3	0%	0%	20%	40%	10%	10%	20%	10%	0%	10%	0%
0,4	0%	0%	80%	90%	60%		50%	50%	60%	10%	0%
0,5	0%	0%	100%	100%	70%	100%	70%	70%	50%	90%	70%
0,6	0%	0%	100%	100%	100%	100%	100%	100%	100%	80%	70%
0,7	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%
0,8	0%	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%
0,9	0%	0%	80%	70%	100%	100%	100%	100%	100%	100%	100%
1	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%

Obě tabulky (Tab. 8 i Tab. 9) jsou navzájem provázané. V praxi se bude uživatel chtít pohybovat v černě označené části tabulky, neboť tam je pravděpodobnost nalezení správného extrému nejvyšší a zároveň na místech, kde jsou hodnoty rychlosti nejnižší. Z tohoto pohledu je pro danou funkci faktor 0,7 až 0,8 a práh křížení 0,8 až 1. Hodnotě 1 je ale lepší se vyhnout, aby se při evoluci zohledňovaly i předchozí generace a nejednalo se o evoluci čistě náhodnou.

Tab. 9. Průměrná první generace nalezení známého extrému v závislosti na parametrech CR a F

	CR										
F	**0**	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1**
0	-	-	-	-	-	-	-	-	-	-	-
0,1	-	-	-	-	-	-	-	-	-	-	-
0,2	-	-	-	-	-	-	-	-	-	-	-
0,3	-	-	2087	1020	515	391	301	247	-	120	-
0,4	-	-	2361	1075	725	442	345	230	180	133	-
0,5	-	-	**2584**	**1237**	750	**525**	364	259	194	151	100
0,6	-	-	**2874**	**1396**	**872**	**626**	**432**	**315**	**222**	191	132
0,7	-	-	**2930**	**1544**	**910**	**664**	**468**	**344**	**255**	**199**	**157**
0,8	-	-	**3436**	**1641**	**1050**	**771**	**524**	**386**	**194**	**203**	**179**
0,9	-	-	3547	1938	**1163**	**840**	**617**	**428**	**336**	**232**	**192**
1	-	-	2087	1020	515	391	301	247	-	120	-



Obr. 26. Úspěšnost DE v závislosti na parametrech

5.1.2 Srovnání strategií

Jednotlivé strategie srovnávám na základě pořadí první úspěšné generace a průměrné doby výpočtu evoluce. Data jsou průměrem ze třiceti běhů diferenciální evoluce při těchto parametrech: {Testovaci funkce=1, CR = 0.8, NP = 15, F = 0.8, generace = 1000, D = 3, Chaoticky system=2}. Tabulka (Tab. 10) ukazuje celkové srovnání všech uvedených strategií v pořadí podle průměrného počtu generací, tabulka (Tab. 11) podle doby výpočtu a tabulka (Tab. 12) srovnání strategií křížení.

Tab. 10. Pořadí strategií podle průměrného počtu generací

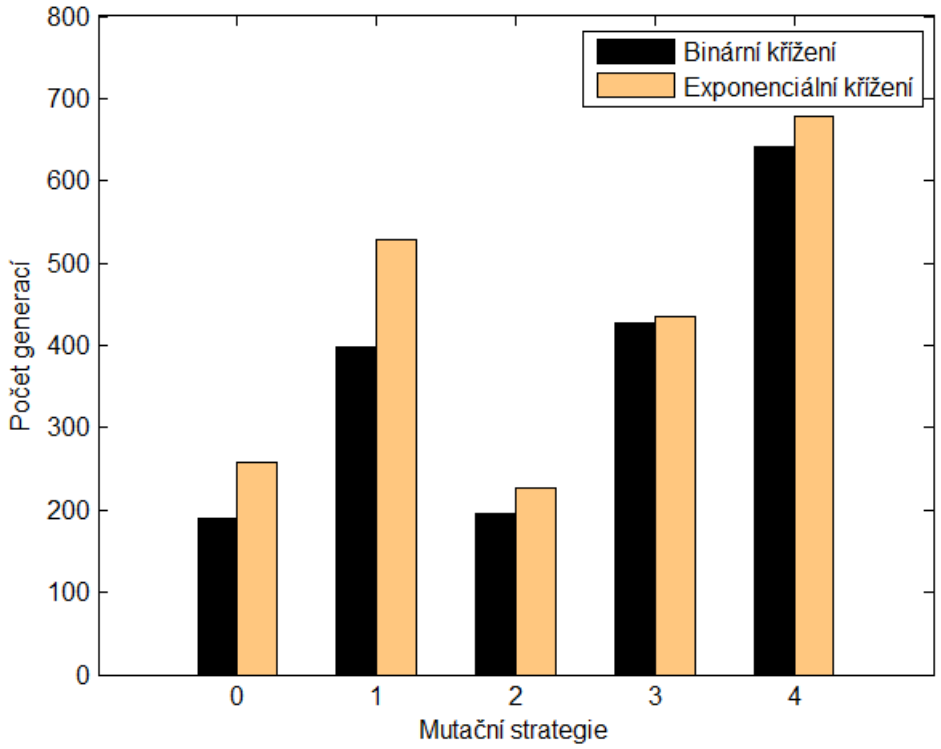
Použitá strategie	**Průměrný počet generací**	**Průměrná doba výpočtu**
Best/1/Bin	190	0,614 s
RandToBest/1/Bin	195	0,589 s
RandToBest/1/Exp	226	0,582 s
Best/1/Exp	257	0,643 s
Rand/1/Bin	399	0,758 s
Best/2/Bin	428	0,744 s
Best/2/Exp	435	0,925 s
Rand/1/Exp	528	0,792 s
Rand/2/Bin	642	0,863 s
Rand/2/Exp	678	0,984 s

Tab. 11. Pořadí strategií podle průměrné doby výpočtu

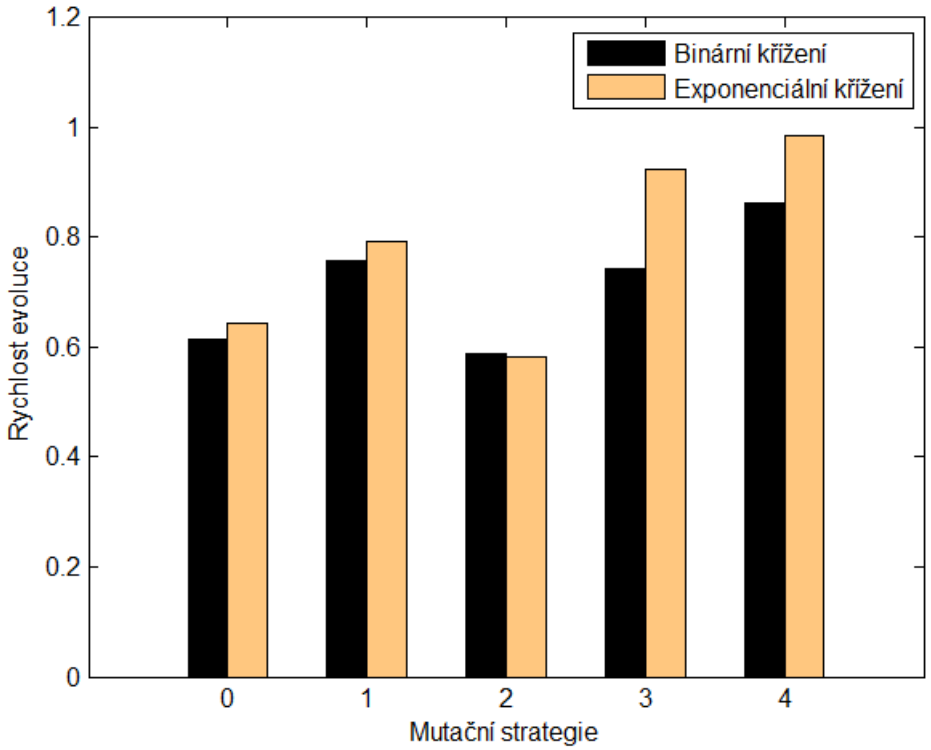
Použitá strategie	**Průměrný počet generací**	**Průměrná doba výpočtu**
RandToBest/1/Exp	226	0,582
RandToBest/1/Bin	195	0,589
Best/1/Bin	190	0,614
Best/1/Exp	257	0,643
Best/2/Bin	428	0,744
Rand/1/Bin	399	0,758
Rand/1/Exp	528	0,792
Rand/2/Bin	642	0,863
Best/2/Exp	435	0,925
Rand/2/Exp	678	0,984

Tab. 12. Srovnání exponenciálního a binárního křížení

	Binární křížení		**Exponenciální křížení**	
Mutační strategie	**Počet generací**	**Doba výpočtu**	**Počet generací**	**Doba výpočtu**
Best/1	190	0,614 s	257	0,643 s
Rand/1	399	0,758 s	528	0,792 s
RandToBest/1	195	0,589 s	226	0,582 s
Best/2	428	0,744 s	435	0,925 s
Rand/2	642	0,863 s	678	0,984 s



Obr. 27. Srovnání strategií křížení: počet generací



Obr. 28. Srovnání strategií křížení: rychlost evoluce

Z výsledných grafů vidíme, že až na jeden případ vychází binární strategie křížení lépe nežli exponenciální. Dosahuje lepších výsledků jak v pořadí první generace, která nachází extrém, tak i v době vykonávání. Jedinou výjimkou je strategie RandToBest/1/Exp, která dosáhla nejrychlejšího vyhledání extrémů ze všech testovaných strategií.

Nutno také podotknout, že rozdíly mezi výsledky jednotlivých strategií jsou minimální a v rámci běžného použití je uživatel mnohdy ani nemusí zaznamenat.

5.2 Porovnání výsledků diferenciální evoluce s klasickým pseudo-náhodným a chaotickým generátorem čísel

Při porovnávání evolucí s pseudo-náhodným generováním čísel pomocí funkce *rand()* a generovaným disipativním systémem jsem porovnávala tyto faktory:

- čas běhu programu před jeho ukončením (při zjištění stagnace se program sám po určité množství generací ukončí)
- první úspěšnou generaci, která našla známý extrém

Uplynulý čas, než se program sám ukončí, byl opět měřen aritmetickým průměrem běhu několika evolucí za stejných parametrických podmínek. Při velikosti populace 15 a 1000 generací se program evoluce s funkcí *rand()* ukončil po 0,66 sekundách, zatímco evoluce s *chaosRand()* po 0,88 sekundách. Evoluce s funkcí *rand()* je tedy mírně rychlejší než evoluce využívající *chaosRand().* Obě výsledné rychlosti jsou ale adekvátní optimalizačním potřebám.

Pořadí první úspěšné generace, která dosáhla známého extrému, ukazuje opačný výsledek. V případě deterministického chaosu s disipativním systémem se jedná průměrně o 93. generaci, zatímco v druhém případě o generaci 126.

Uvedené výsledky jednotlivých srovnání ovšem neplatí obecně. Efektivita evoluce se bude lišit v závislosti jak na daném chaotickém modelu, tak i na vybrané testovací funkci.

ZÁVĚR

Cílem této práce bylo ukázat, že diferenciální evoluce je technika, která ačkoliv je založena na jednoduchých matematických operacích, dokáže vyřešit i extrémně složité problémy velmi efektivně.

Práce představuje princip diferenciální evoluce, chaotických deterministických systémů a testovací účelové funkce, které jsou zobrazeny v klasických, ale i vrstevnicových grafech. V praktické části pak byl popsán použitý objektový návrh řešení diferenciální evoluce s prvky deterministického chaosu a za pomoci vytvořené aplikace byla ověřena funkčnost algoritmu a představeny jeho výsledky. Ty jsou zkoumány na základě závislosti na vybraných parametrech, strategiích a výběru typu generátoru čísel.

Výsledná aplikace vyhledává extrém daných známých testovacích funkcí popsaných v teoretické části. Použitý algoritmus diferenciální evoluce je na testování účelové funkce nezávislý, takže v případě, kdy by uživatel potřeboval optimalizovat nějakou novou zatím nedefinovanou funkci, do programu je možné novou testovací funkci snadno doplnit.

U diferenciální evoluce sice není plně zaručena jistota nalezení optimálního řešení, ale při vhodném zadání parametrů dosahuje vynikajících výsledků v krátkém čase.

CONCLUSION

The purpose of this thesis is to show that differential evolution although it is a technique based on simple mathematical operations, is able to solve even extremely difficult problems quite effectively.

The principle of differential evolution, chaotic deterministic systems and test functions were introduced and shown on classical and contour plots. The practical part of this thesis included object-oriented design of chaotic differential evolution implementation and the functionality verification of this algorithm checked by the outcomes of the described application. Part of the thesis consisted of a research of the evolution outcome sensitivity on selected parameters and the impact of the choice of a number generator.

The application searches extremes of known test functions described in the theoretical part. Because of the fact that used differential evolution algorithm function is independent of test functions, it is possible to easily add new test functions if needed.

Differential evolution is not guaranteed to find the optimal solution, but when appropriate parameters are set, it reaches remarkable results in short time.

SEZNAM POUŽITÉ LITERATURY

[1] ZELINKA, Ivan a František VČELAŘ. *Evoluční výpočetní techniky: principy a aplikace*. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.

[2] ZELINKA, Ivan. *Umělá inteligence v problémech globální optimalizace*. 1. vyd. Praha: BEN - technická literatura, 2002. ISBN 80-7300-069-5.

[3] ONWUBOLU, Godfrey C. a B. V. BABU. *New Optimization Techniques in Engineering***.** Berlin: Springer, 2004, xxii, ISBN 35-402-0167-X.

[4] INTERNATIONAL MULTICONFERENCE ON COMPUTER SCIENCE AND INFORMATION TECHNOLOGY, October 20-22 a B BABU. *2008 International Multiconference on Computer Science and Information Technology (Imcsit).* S. l.: IEEE, 2009, xxii, 712 s. ISBN 978-836-0810-149.

[5] Differential Evolution: for Continuous Function Optimization. STORN, Rainer. *Differential Evolution Homepage* [online]. 2005 [cit. 2013-03-29]. Dostupné z: http://www1.icsi.berkeley.edu/~storn/code.html

[6] BOTEV, Zdravko. Non-linear Continuous Multi-Extremal Optimization. *The Cross-Entropy Toolbox* [online]. 2004-12-17 [cit. 2013-03-30]. Dostupné z: http://www.maths.uq.edu.au/CEToolBox/node3.html

[7] Test Functions: for Unconstrained Global Optimization. HEDAR, Abdel-Rahman. *Global Optimization Test Problems* [online]. 2005 [cit. 2013-04-02]. Dostupné z: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

[8] GENTLE, James E. *Random number generation and Monte Carlo methods*. New York: Springer, 2003. 2nd ed. ISBN 0-387-0017-6. Dostupné z: http://www3.inatel.br/docentes/dayan/TP501/Random%20Number%20Generation%20and%20Monte%20Carlo%20Methods%202nd%20Ed%20-%20Gentle.pdf

[9] KRATOCHVÍL, Ctirad, Pavel HERIBAN. *Dynamické systémy a chaos.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky, 2010, 80 s. ISBN 978-80-214-4056-2. Dostupné z: http://www.umt-old.fme.vutbr.cz/images/stories/opory/2009_dynamicke_systemy_a_chaos.pdf

[10] HILBORN, Robert C. *Chaos and nonlinear dynamics: an introduction for scientists and engineers*. 2nd ed. Oxford: University Press, 2000, xxi, 650 s. ISBN 01-985-0723-2.

[11] FUCHS, Radovan. *Deterministický chaos jako generátor náhodných čísel v prostředí C a C++*. Zlín, 2012. Bakalářská práce. UTB ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Ing. Roman Šenkeřík, Ph.D.

[12] KRATOCHVÍL, Ctirad. *Bifurkace a chaos v technických soustavách a jejich modelování* [online]. Vyd. 1. Brno: ÚT AVČR v Praze, pobočka Brno, 2008, 108 s. [cit. 2013-05-02]. ISBN 978-80-87012-20-8. Dostupné z: http://www.umt-old.fme.vutbr.cz/_studium_/monografie/2008_Bifurkace_a_chaos.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EVT	Evoluční výpočetní techniky.
DE	Diferenciální evoluce.
ChaosDE	Diferenciální evoluce s prvky deterministického chaosu.
CR	Práh křížení.
F	Mutační konstanta.
NP	Velikost populace.
CV	„Cost Value“ – hodnota účelové funkce.
GA	Genetické algoritmy.
Bin	Binární křížení
Exp	Exponenciální křížení

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PI CD s textem práce a zdrojovými kódy