

Moderní adaptivní diferenciální evoluce

Bc. Jiří Hlaváček

Diplomová práce
2015



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jiří Hlaváček**
Osobní číslo: **A13482**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **kombinovaná**

Téma práce: **Moderní adaptivní diferenciální evoluce**
Téma anglicky: **Modern Adaptive Differential Evolution**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Popište odlišnosti mezi kanonickou verzí a moderními adaptivními verzemi algoritmu Diferenciální Evoluce.
3. Naprogramujte zvolenou verzi adaptivní Diferenciální Evoluce v prostředí C/C++/C#
4. Otestujte algoritmus na sadě testovacích benchmark funkcí.
5. Vytvořte jednotný systém výstupu dat kompatibilní s jinými statistickými nebo matematickými SW pro následnou analýzu dat.
6. Výsledky testování přehledně graficky a tabulkově zobrazte.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. BEN, 2002, 190 s. ISBN 80-7300-069-5.
2. ZELINKA, Ivan. Evoluční výpočetní techniky: principy a aplikace. 1. vyd. Praha: BEN – technická literatura, 2009, 534 s. ISBN 978-80-7300-218-3.
3. DE JONG, Kenneth A. Evolutionary computation: a unified approach. Cambridge: MIT Press, 2006, ix, 256 s. ISBN 02-620-4194-4.
4. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence, Academia, 1993, ISBN 80-200-0496-3.
5. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence 4., Academia, 2003, ISBN 80-200-1044-0.
6. 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.
7. PRICE, Kenneth V, Rainer M STORN a Jouni A LAMPINEN. Differential evolution: a practical approach to global optimization [online]. Berlin: Springer, 2005.
8. Handbook of Optimization: From Classical to Modern Approach. 2013. vyd. Editor Ivan Zelinka, Václav Snášel, Ajith Abraham. Berlin: Springer, 2013, xii, 1100 s. Intelligent systems reference library, 38. ISBN 978-3-642-30503-0.

Vedoucí diplomové práce:

doc. Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

6. února 2015

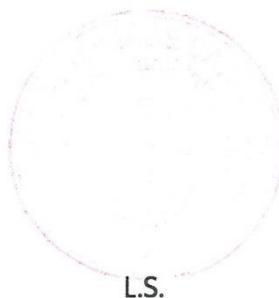
Termín odevzdání diplomové práce:

15. května 2015

Ve Zlíně dne 6. února 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



L.S.



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

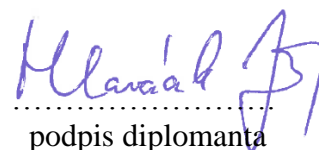
Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/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 diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/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 diplomovou/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 – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/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 diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/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 diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 15. 5. 2015


.....
podpis diplomanta

ABSTRAKT

Diplomová práce se zabývá adaptivní verzí diferenciální evoluce s použitím deterministického chaosu pro generování náhodných čísel. Teoretická část obsahuje popis problematiky diferenciální evoluce, adaptivní diferenciální evoluce a popis problematiky generování náhodných čísel jako součásti diferenciální evoluce. Praktická část obsahuje popis aplikace realizované v C# a výsledky testování výkonnosti klasické diferenciální evoluce s adaptivní verzí chaotického systému pro generování náhodných čísel a výkonnosti adaptivní diferenciální evoluce i s adaptivní verzí deterministického chaosu na sadě testovacích funkcí.

Klíčová slova: diferenciální evoluce, deterministický chaos, jDE, JADE, SADE, EPSDE, adaptivní parametry, Lozi mapa, generátor náhodných čísel

ABSTRACT

The master's thesis aims on adaptive differential evolution with deterministic chaos random number generator. The theoretical part consist from description of differential evolution field, adaptive differential evolution field and description of generating random numbers issue like part of differential evolution. The practical part provides description of application realized in C# and results of benchmark testing of classic differential evolution with adaptive version of chaotic system for generating random numbers and benchmark testing of adaptive differential evolution extended by adaptive version of deterministic chaos on set of benchmark functions.

Keywords: differential evolution, deterministic chaos, jDE, JADE, SADE, EPSDE, adaptive parameters, Lozi map, random number generator

Tímto bych chtěl poděkovat Doc. Ing. Romanu Šenkeříkovi, Ph.D. za ochotu, pomoc a cenné rady při tvorbě této práce. Děkuji také svojí partnerce a dětem za trpělivost a podporu po celou dobu mého studia.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 EVOLUČNÍ ALGORITMY	11
1.1 HISTORIE EVOLUČNÍCH ALGORITMŮ	11
1.2 PRINCIPY FUNKCE EVOLUČNÍCH ALGORITMŮ	12
2 DIFERENCIÁLNÍ EVOLUCE.....	15
2.1 PRINCIPY FUNKCE DIFERENCIÁLNÍ EVOLUCE	15
2.1.1 Stanovení parametrů.....	15
2.1.2 Tvorba první populace	16
2.1.3 Evoluční cyklus generace.....	16
2.1.4 Testování ukončovacích podmínek.....	19
2.1.5 Vyhodnocení generace	20
3 ADAPTIVNÍ DIFERENCIÁLNÍ EVOLUCE	21
3.1 PRINCIPY ADAPTIVNOSTI	21
3.2 JDE	21
3.3 SADE	22
3.4 JADE	23
3.5 EPSDE.....	24
4 GENEROVÁNÍ NÁHODNÝCH ČÍSEL.....	26
4.1 PSEUDONÁHODNÁ ČÍSLA	26
4.2 CHAOTICKÉ SYSTÉMY	27
II PRAKTICKÁ ČÁST	29
5 APLIKACE SELF-ADAPTIVE CHAOTIC DE	30
5.1 OBJEKTOVÝ NÁVRH APLIKACE	31
5.1.1 Rozhraní pro implementaci tříd	31
5.1.2 Implementace tříd.....	33
5.2 OVLÁDÁNÍ APLIKACE	41
6 VÝSTUPY TESTOVÁNÍ.....	44
6.1 TESTOVACÍ FUNKCE	44
6.2 ADAPTIVNÍ HODNOTY CHAOSRND	48
6.3 ADAPTIVNÍ HODNOTY PARAMETRŮ JDE.....	50
6.4 EFEKTIVITA VARIANT DE.....	52
6.4.1 První De Jongova funkce	53
6.4.2 Rastriginova funkce	55
6.4.3 Schwefelova funkce	57
6.4.4 Ackleyho funkce II.....	58
7 SHRUTÍ VÝSLEDKŮ TESTOVÁNÍ.....	60
ZÁVĚR	61
CONCLUSION	62
SEZNAM POUŽITÉ LITERATURY.....	63
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	65

SEZNAM OBRÁZKŮ	66
SEZNAM TABULEK.....	68
SEZNAM PŘÍLOH.....	69

ÚVOD

V moderních informačních technologiích je využívání a rozvíjení heuristických algoritmů jedním s dynamicky se rozvíjejících směrů. Jednou s těchto oblastí je skupina evolučních algoritmů, která se využívá především v optimalizačních úlohách. Optimalizační úlohy je třeba řešit napříč různými obory a v současnosti se ukazuje, že jeden z nejvhodnějších typů algoritmů pro řešení těchto úloh je diferenciální evoluce.

Algoritmů diferenciální evoluce je více různých variant, které využívají různých metod mutace a křížení jedinců v populacích. Jednotlivé varianty diferenciální evoluce se ukazují různě efektivní na rozdílné typy úloh. Dalším důležitým faktorem ovlivňujícím výkonnost všech variant diferenciální evoluce je nastavení jejích řídicích parametrů.

Problematiku optimálního nastavení řídicích parametrů je možné řešit pomocí adaptivní verze diferenciální evoluce, kde se nastavení řídicích parametrů přizpůsobuje, aby měl algoritmus lepší výkonnost při řešení daného úkolu.

V rámci algoritmu diferenciální evoluce se využívá generování náhodných hodnot. Používanou metodou je generátor pseudonáhodných čísel anebo je náhodnost generována využitím deterministického chaosu.

Cílem práce je vytvoření funkční aplikace, která umožní provádět diferenciální evoluci a adaptivní diferenciální evoluci s využitím adaptivních změn řídicích parametrů evoluce i deterministického chaosu. Tyto verze diferenciální jsou následně porovnány i s klasickou diferenciální evolucí využívající pseudonáhodná čísla na sadě testovacích funkcí.

I. TEORETICKÁ ČÁST

1 EVOLUČNÍ ALGORITMY

Principy evoluce, tak jak je popisuje moderní evoluční syntéza, která vychází ze spojení Darwinovy evoluční teorie a Mendelovy teorie dědičnosti, jsou stále platné. I když existuje v oblasti přírodních věd několik směrů, které do určité míry zpochybňují platnost této teorie, tak je stále brána jako součást základních principů v této vědecké oblasti. [1]

Evoluční výpočetní techniky (EVT) jsou numerické algoritmy, které z těchto teorií vycházejí. Základním paradigmatem těchto algoritmů je tedy podobně jako u Darwinovy a Mendelovy teorie křížení jednotlivců v rámci populací, mutace jejich genomu a cyklický vznik potomků do dalších generací. Rodiče a potomci, kteří nejsou vhodni pro životní prostředí (řešení optimalizační úlohy), vymírají (jsou vymazáni). Potomci se pak stávají rodiči v další generaci. [2]

Algoritmů EVT využívajících tento základní princip je více druhů (simulované žíhání, rojení částic, genetické algoritmy, mravenčí algoritmy, SOMA, diferenciální evoluce). Každý druh EVT používá mírně odlišnou strategii, ale základní principy jsou stále stejné.

1.1 Historie evolučních algoritmů

První úspěšné použití evoluční strategie bylo realizováno v polovině 60. let a genetické algoritmy se začínají objevovat v polovině 70. let. Nicméně ačkoliv nebyly programově realizovány, byly základní principy popisující evoluční algoritmy formulovány již dříve matematikem A. M. Turingem, matematikem N. A. Barricellim a dalšími. [2]

Od prvních simulací po současný výzkum v této oblasti prošly evoluční algoritmy dynamickým rozvojem. Postupně vědci testováním a různými úpravami algoritmů vyvinuli rozdílné typy algoritmů postavených na základech EVT. Na základech Genetických algoritmů představených Johnem Hollandem vznikla celá třída algoritmů, kde každá verze algoritmu využívá různé strategie křížení, mutace a ocenění jedinců v populaci. Paralelně s genetickými algoritmy se vyvíjely algoritmy označené jako Evoluční strategie. Evoluční strategie byly také vyvíjeny v různých verzích, které používaly rozdílné strategie pro vytváření nových jedinců do následujících generací (potomků).

Dalšími algoritmy, které postupným vývojem vznikly na základě inspirace v sociálním chování živočišných společenstev, jsou Rojení částic, Optimalizace mravenčí kolonií nebo Samoorganizující se migrační algoritmus (SOMA).

Velmi úspěšným algoritmem, který byl vyvinut v původní verzi na základě genetického žihání, je Diferenciální evoluce. Od první verze prošla Diferenciální evoluce postupným vývojem a v současnosti se ukazuje velice efektivním nástrojem v optimalizačních úlohách.

Evoluční výpočetní techniky se v současnosti ukazují jako silný nástroj v oblasti optimalizace a mnoho vědců na celém světě se zabývá dalším vývojem jednotlivých algoritmů, které je možné využít při vývoji různých technologií napříč nejen technickými odvětvími.

1.2 Principy funkce evolučních algoritmů

Evoluční algoritmy spadají do kategorie heuristických postupů, kdy řešení nemusí vždy být nalezeno nebo nemusí být zcela přesné. EVT jsou však používány na řešení úloh, které mají charakter n -dimenzionálních funkcí, kde nalezení přesného řešení může být časově velice náročné nebo postup řešení není vůbec známý. Díky použití EVT je možné dosáhnout velice dobrých výsledků i na takových úlohách.

Obecný cyklus EVT pro jednotlivé generace probíhá v následujících krocích (Obr. 1):

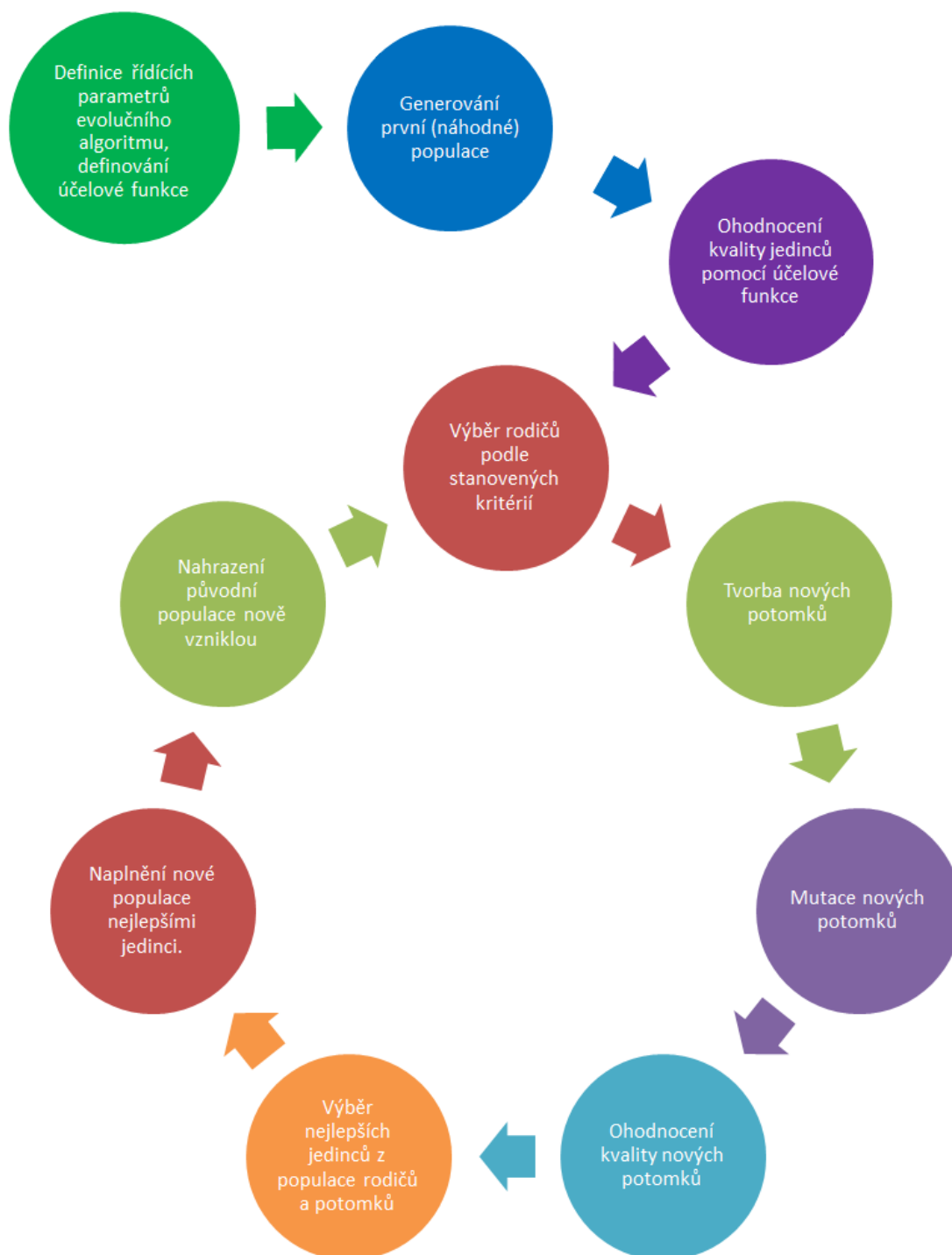
1. Nastavení řídicích parametrů daného evolučního algoritmu – tyto parametry ovlivňují efektivitu algoritmu a určují, za jakých podmínek algoritmus končí. Definování účelové funkce, která slouží k ohodnocení tzv. vhodnosti jedince pro řešení daného problému – hledá se maximum nebo minimum definované účelové funkce.
2. Vygenerování první populace jedinců. Jedinci jsou vektory hodnot, jejichž počet je dán počtem argumentů účelové funkce. Počet jedinců v populaci je jedním z řídicích parametrů evolučního algoritmu. Implementace populace záleží na programátorském přístupu. Může se například jednat o dvojrozměrnou matici nebo o jednorozměrné pole obsahující jedince jako objekty třídy definující jejich strukturu. Hodnoty reprezentující jedince jsou generovány náhodně tak, aby pokryly prostor možných řešení.
3. Ohodnocení všech jedinců populace pomocí vyhodnocení účelové funkce. Vhodnost může být přímo hodnota účelové funkce nebo upravená (normalizovaná) hodnota účelové funkce.

4. Výběr jedinců v populaci, ze kterých se stanou rodiče, pro vytvoření nových jedinců do další populace. Výběr jedinců záleží na strategii konkrétního algoritmu – podle vhodnosti, částečně náhodně nebo podle dalších kritérií.
5. Tvorba potomků křížením rodičů. Strategie křížení se různí podle použitého evolučního algoritmu. Může se jednat o prohazování některých hodnot u jedinců nebo o operace prováděné mezi jednotlivými hodnotami tvořící rodiče.
6. Je provedena mutace nově vytvořených potomků. Dojde k pozměnění hodnot jedinců podle nastavených řídicích parametrů evolučního algoritmu.
7. Dojde k ohodnocení všech potomků stejným způsobem, jakým byli ohodnoceni rodiče v kroku 3.
8. Jsou vybráni nejlepší jedinci z nově vzniklých potomků i rodičů.
9. Vybraní nejlepší jedinci vytvoří novou populaci, ze které se stane populace pro výběr rodičů do dalšího cyklu evolučního algoritmu.
10. Původní populace je nahrazena nově vzniklou populací.

Dále se v cyklech opakují kroky 4 až 10 do doby než dojde k ukončení algoritmu na základě podmínek stanovených řídicími parametry evolučního algoritmu. Obvykle se jedná o předem stanovený počet evolučních cyklů nebo o detekci minimální změny nejlepších jedinců v populaci po stanovený počet evolučních cyklů.

Jednotlivé evoluční algoritmy se mohou od tohoto základního schématu lišit. Některé méně (Rojení částic neobsahuje křížení a mutaci) a některé více (Optimalizace mravenčí kolonií používá zcela odlišnou strategii), všechny jsou však označovány jako Evoluční výpočetní techniky.

Evoluční výpočetní techniky jsou populární, protože jsou schopny řešit velice obtížně řešitelné úkoly a v některých případech (např. konstrukční návrhy) jsou schopny nahradit člověka. [2]



Obr. 1 Obecný cyklus evolučního algoritmu

2 DIFERENCIÁLNÍ EVOLUCE

Diferenciální evoluce (DE) je z pohledu historického vývoje EVT poměrně novým algoritmem. Byl vyvinut v roce 1995 K. V. Pricem a R. Stromem. Principy DE jsou podobné genetickým algoritům. DE v původním návrhu stavěla na principu genetického žíhání s tím, že došlo k záměně binární reprezentace za dekadickou a logických operací za vektorové. Dále byla doplněna tzv. diferenciální mutace (přičtení rozdílu dvou náhodně vybraných jedinců ke třetímu) a následně byla zkombinována s metodou výběru z genetického žíhání. Následně byl princip žíhání z DE vypuštěn.

DE na rozdíl o genetických algoritmů, kterým se nejvíc podobá, využívá k tvorbě potomků čtyři rodiče, zatímco genetické algoritmy jen dva. Dalším rozdílem oproti genetickým algoritmům je to, že prvně dochází k mutaci až poté ke křížení (viz níže).

2.1 Principy funkce diferenciální evoluce

Základní princip DE dodržuje z velké části obecné principy evolučních algoritmů popsaných v kapitole 1.2. Jednotlivé kroky před zahájením evolučního cyklu, během něj i po ukončení cyklu jsou specifické v nastavení řídicích parametrů, výběru strategie mutace a křížení. Ukončovací podmínky by měly být nastaveny tak, aby byly schopny detekovat stagnaci, ke které může u DE dojít. Následující podkapitoly popisují jednotlivé kroky DE.

2.1.1 Stanovení parametrů

Řídicí parametry určují, jak efektivní bude průběh celé evoluce. Při nevhodném nastavení parametrů může evoluce hledat řešení pomaleji nebo ho nemusí nalézt vůbec. Jedná se o následující parametry (doporučené hodnoty parametrů jsou převzaty z [3]):

- **F** – mutační konstanta. Jedná se o hodnotu, která je z intervalu $[0, 2]$. Doporučená hodnota je v rozmezí $0,3 - 0,9$. Při výpočtu mutace je využívána jako koeficient podle zvolené mutační strategie.
- **CR** – práh křížení. Hodnota tohoto parametru je z intervalu $[0, 1]$. Doporučená hodnota je v rozmezí $0,8 - 0,9$. V případě, že hodnota tohoto parametru bude nastavená na 0, tak nebude docházet k žádným mutacím jedinců, kteří jsou vybírání do nových generací. Bude-li hodnota nastavena na 1, tak budou noví jedinci mít náhodné hodnoty a bude docházet pouze k náhodnému prohledávání prostoru možných řešení.

- **D** – dimenze. Je dána počtem argumentů účelové funkce, což znamená, že je dána funkčním popisem řešeného problému a je možné ji ovlivnit pouze vytvořením jiné účelové funkce popisující danou problematiku.
- **NP** – počet jedinců v populaci. Nejmenší počet jedinců je teoreticky 4, reálně se však počet jedinců stanovuje v rozmezí [10D, 100D]. Doporučená hodnota je 10D a při vysoce multimodálních funkcích 100D. V případě, že je populace malá, bude horší výběr z jedinců, u velké populace pak bude časově náročné tvoření populací nových.
- **Specimen** – prototyp jedince. Jedná se o definici typů (včetně omezení), jakých mohou nabývat hodnoty, které tvoří jedince. Generování jedinci musí mít jednotlivé hodnoty odpovídající těmto omezením. Např.: {(Real, -10, 15), (Integer, -1, 1)}
- **Generations** – počet evolučních cyklů, během nich se jednotlivé generace vyvíjí. Musí být rozhodně větší jak nula. Tento parametr určuje, kdy nejpozději je DE ukončena. Bude-li počet generací stanoven příliš malý, může DE skončit dříve než nalezne řešení optimalizační úlohy. Když však bude počet generací velký může DE trvat dlouhou dobu. Je třeba tedy uvážit i výkon počítače, na kterém bude DE spuštěna.

2.1.2 Tvorba první populace

První populace, která slouží pro zahájení evolučního cyklu, je vytvořena podle nastavených řídicích parametrů popsanych v kapitole 2.1.1 tak, jednotlivé hodnoty všech jedinců jsou generovány náhodně. Pro generování náhodných hodnot je možné využít generátor pseudo-náhodných čísel nebo generátor fungující na principu deterministického chaosu. Důležité je, aby generované hodnoty splňovaly omezení daná prototypem jedince (Specimen). Jednotlivé hodnoty jedinců první populace a jejich ohodnocení účelové funkce. Ohodnocení účelové funkce (Cost value) je uloženo jako součást jedince nezávisle na tom, jakým způsobem jsou jedinci programově realizováni (vektory, objekty, atd.).

2.1.3 Evoluční cyklus generace

Evoluční cyklus probíhající na jedné generaci spočívá v tom, že se postupně vybírají všichni jedinci populace jako tzv. aktivní jedinec. S každým vybraným aktivním jedincem je prováděna mutace a křížení, až do vyčerpání všech jedinců populace. Aktivní jedinci buď v populaci zůstanou do další generace, nebo jsou nahrazeni tzv. zkušebním vektorem.

Po dokončení tohoto cyklu je dokončena jedna generace a začíná další cyklus, až do chvíle, než jsou splněny ukončovací podmínky. Postup tvorby zkušebního jedince a rozhodnutí o jedinci, který bude zařazen do další generace je následující:

1. **Mutace** – probíhá na základě operací se čtyřmi rodiči. Jedním z rodičů je aktivní jedinec evolučního cyklu a k němu jsou náhodně zvoleni tři další nestejní jedinci z aktuální populace. Dva náhodně vybraní jedinci jsou od sebe odečtení (rozdíl vektorů reprezentujících hodnoty parametrů jedince). Tímto rozdílem vznikne tzv. diferenční vektor. Hodnoty diferenčního vektoru jsou následně vynásobeny mutační konstantou F a tím vznikne tzv. váhový diferenční vektor. Hodnoty váhového diferenčního vektoru jsou následně vektorově přičteny ke třetímu náhodně vybranému vektoru a výsledkem je tzv. šumový vektor. Vektorové operace mutace se mohou lišit na základě vybrané mutační strategie (Tab. 1.).

Tab. 1 Mutační strategie [4]

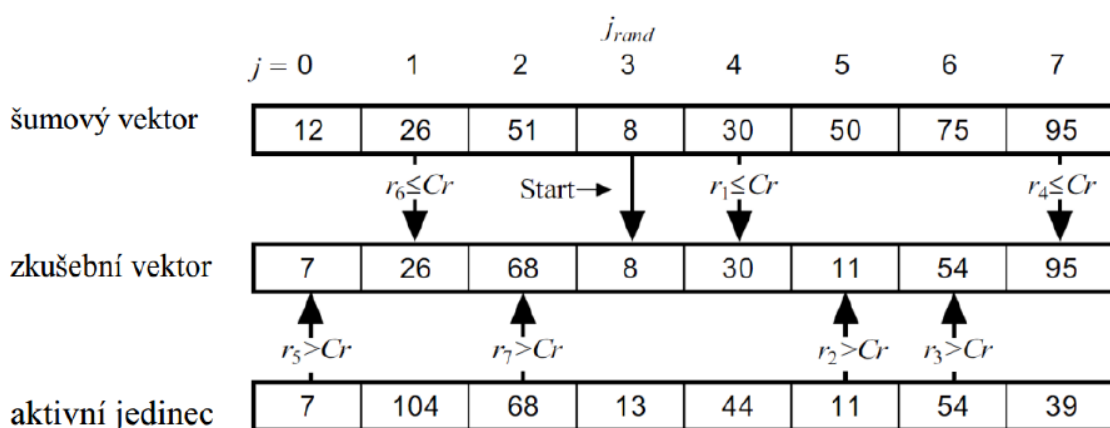
best/1	$v = x_{best,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$	(1)
rand/1	$v = x_{r1,j}^G + F \cdot (x_{r2,j}^G - x_{r3,j}^G)$	(2)
rand-to-best/1	$v = x_{i,j}^G + \lambda \cdot (x_{best,j}^G - x_{i,j}^G) + F \cdot (x_{r1,j}^G - x_{r2,j}^G)$	(3)
best/2	$v = x_{best,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G)$	(4)
rand/2	$v = x_{r5,j}^G + F \cdot (x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G)$	(5)

2. **Křížení** – probíhá mezi mutací vzniklým šumovým vektorem v a aktivním jedincem x_i . Výsledkem křížení je tzv. zkušební vektor y . Řídicím parametrem křížení je práh křížení CR . V rámci obou vybraných jedinců probíhají dále popsané postupy vždy na odpovídajících parametrech (stejný index, odpovídající vlastnost objektu, apod.). Prvně je náhodně vybrána jedna odpovídající dvojice parametrů z jedinců a do cílového vektoru je přesunuta hodnota z šumového vektoru. Tím je zabezpečena podmínka křížení DE, že minimálně jeden z parametrů musí být z šumového vektoru [2]. Další postup je závislý na zvolené strategii křížení:

- **binomické křížení** – postupně jsou vybrány ostatní odpovídající dvojice parametrů z obou jedinců a pro každou dvojici je vygenerováno náhodné číslo

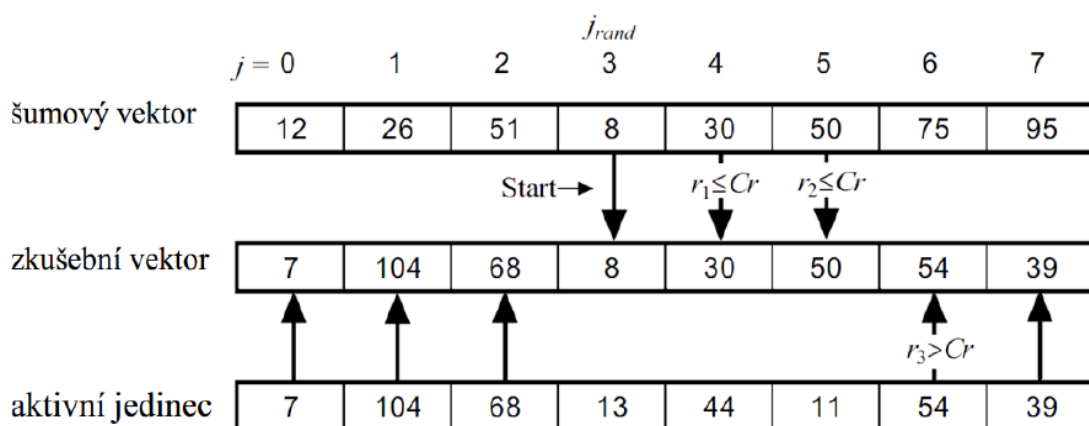
v platném rozmezí pro CR ($[0, 1]$). Pokud je vygenerované číslo menší nebo rovno nastavenému CR, tak je do cílového jedince přesunuta hodnota z šumového vektoru. V případě, že je vygenerované číslo větší než CR, tak je do cílového jedince přesunuta odpovídající hodnota z aktivního jedince. Tuto strategii je možné vyjádřit vztahem (6) nebo ji také znázorňuje schéma (Obr. 2).

$$y_j = \begin{cases} v_j \Leftrightarrow U_j \leq CR \vee j = l \\ x_{ij} \Leftrightarrow U_j > CR \wedge j \neq l \end{cases}, l \in \{1, 2, 3, \dots, D\} \quad [4] \quad (6)$$



Obr. 2 Binomické křížení [5]

- **exponenciální křížení** - postupně jsou vybrány ostatní odpovídající dvojice parametrů z obou jedinců a pro každou dvojici je vygenerováno náhodné číslo v platném rozmezí pro CR ($[0, 1]$) stejně jako u binomického křížení. Pokud je vygenerované číslo menší nebo rovno nastavenému CR, tak je do cílového jedince přesunuta hodnota z šumového vektoru. V případě, že je vygenerované číslo větší než CR, tak je do cílového jedince přesunuta odpovídající hodnota z aktivního jedince opět stejně jako u předchozí varianty. Pokud je však u některého parametru přesunuta hodnota z aktivního jedince, tak pak už jsou všechny zbylé parametry také přesunuty z aktivního jedince. Tuto strategii znázorňuje schéma na následujícím obrázku (Obr. 3).



Obr. 3 Exponenciální křížení [5]

3. Výběr jedince do nové populace – Je provedeno vyhodnocení účelové funkce pro vzniklý zkušební vektor. V případě, že je vhodnost zkušebního vektoru lepší než aktivního jedince, je vybrán do nové populace zkušební vektor, a v případě, že má lepší vhodnost aktivní jedinec, je do nové populace zařazen tento jedinec.

Kombinací všech variant mutací a křížení je tedy možné vybrat z deseti variant strategií DE.

2.1.4 Testování ukončovacích podmínek

V základní verzi DE je jedinou ukončovací podmínkou provedení předem stanoveného počtu generačních cyklů. Možných podmínek, za jakých je vhodné DE ukončit nebo pozměnit, může být více. Jednou z takových podmínek může být stav, kdy během probíhající DE již téměř nedochází ke změnám hodnot parametrů jedinců, což může být způsobeno některou z následujících možností:

- DE našla lokální extrém účelové funkce.
- Populace ztratila diverzibilitu.
- Optimalizační proces již téměř neprobíhá.

U DE může také nastat situace, že nenastane ani jedna z předcházejících možností, a přesto optimalizace neprobíhá. Tato situace se nazývá stagnace. [6]

V těchto případech může dojít k ukončení DE nebo ke změně řídicích parametrů a pokračování v běhu evolučních cyklů.

2.1.5 Vyhodnocení generace

V průběhu ukončování evolučních cyklů jednotlivých generací může dojít k statistickému zpracování klíčových hodnot generace (např. hodnota účelové funkce nejlepšího jedince). Tyto hodnoty mohou být použity například pro vizualizaci optimalizačního procesu v grafu nebo pro export do tabulek, které mohou být dále statisticky vyhodnocovány v rámci prozkoumávání efektivity jednotlivých verzí DE.

3 ADAPTIVNÍ DIFERENCIÁLNÍ EVOLUCE

Z popisu funkčnosti vyplývá, že efektivita DE je velmi závislá na nastavení řídicích parametrů DE. Při testování DE na sadě testovacích funkcí je již z předešlých testování známé, jaké nastavení řídicích parametrů vede k efektivnímu průběhu optimalizačního procesu. Pro řešení složitých neotestovaných optimalizačních úloh však může být obtížné správně nastavit řídicí parametry DE. K řešení tohoto úkolu mohou vést adaptivní verze DE.

3.1 Principy adaptivnosti

Princip funkčnosti adaptivních DE spočívá v tom, že do nových populací se přenáší takové nastavení řídicích parametrů, které vede k řešení probíhající optimalizační úlohy. Adaptivní strategie využívají také mutaci řídicích parametrů, která umožňuje v průběhu DE měnit nastavení řídicích parametrů CR a F. Existuje několik otestovaných a publikovaných verzí adaptivní DE, kde každá používá jinou strategii adaptace řídicích parametrů.

Tento způsob práce s řídicími parametry vyžaduje, aby se hodnoty těchto parametrů staly součástí každého jedince, a tím je poté umožněno, že do nových populací jsou s lepšími jedinci přesouvány také hodnoty řídicích parametrů, které je umožnily vytvořit. Na základě náhodného procesu jsou pak obvykle mutovány i hodnoty těchto řídicích parametrů. Strategie přenosu a mutace řídicích parametrů záleží na verzi používané adaptivní DE. V současné době vědci pracují na testování různých verzí těchto adaptivních DE. V následujících kapitolách jsou nastíněny principy funkce adaptivních verzí DE, které jsou považované za nejmodernější a nově vyvíjené varianty jsou s nimi porovnávány. [3]

3.2 jDE

Tuto verzi adaptivní DE představil J. Brest a kol. Jedná se o velice jednoduchou a efektivní variantu adaptivní DE. Základem je využití varianty DE/rand/1/bin s tím, že řídicí parametry F a CR jsou uloženy odděleně pro každého jedince populace spolu s jeho hodnotami parametrů (F a CR se stávají součástí vektoru jedince). Díky tomu dochází k přenosu řídicích parametrů do nové populace spolu s jedincem, pokud je v rámci evolučního cyklu úspěšný.

Hodnoty řídicích parametrů CR a F jsou při tvorbě první populace nastaveny náhodně, stejně jako ostatní hodnoty parametrů jedince. Náhodně vygenerované hodnoty musí spadat do platných intervalů hodnot pro tyto parametry.

Pro F je nastavena hodnota pravděpodobnosti $\tau_1 = 0.1$ a pro CR hodnota pravděpodobnosti $\tau_2 = 0.1$ [7] a s touto pravděpodobností dochází k mutaci řídicích parametrů u jednotlivých členů aktuální populace. Pro obě hodnoty pravděpodobnosti jsou vygenerována náhodná čísla z intervalu $[0, 1]$ a v případě, že je náhodná hodnota menší, než příslušná hodnota pravděpodobnosti, je odpovídající řídicí parametr mutován. Mutace spočívá ve vygenerování nové náhodné hodnoty daného řídicího parametru a to v intervalu $[0, 1]$ pro CR a $[0.1, 0.9]$ pro F. [7] Mutace řídicích parametrů probíhá před zahájením evolučního cyklu, takže tyto nové (mutované) řídicí parametry se již podílí na tvorbě následující generace jedinců.

3.3 SaDE

Adaptivní modifikace SaDE staví princip funkčnosti na myšlence, že v průběhu optimalizačního procesu mohou být efektivnější různé varianty DE pro generování zkušebního vektoru. Volba varianty DE pro tvorbu zkušebního vektoru je realizována na základě pravděpodobnosti, která se v průběhu SaDE mění na základě její předchozí úspěšnosti.

Pro výběr zkušebního vektoru u SaDE je vybírána na základě pravděpodobnosti p varianta (strategie) DE/rand/1/bin, DE/rand/2/bin, DE/rand-to-best/2/bin nebo DE/current-to-rand/1. Výchozí nastavení pravděpodobnosti p je pro všechny strategie stejné (0.25), což znamená, že mají stejnou šanci, že budou vybrány.

Pro SaDE je nastaven počet generací, tzv. učící perioda. Po uběhnutí učící periody dojde k přepočítání pravděpodobností p pro každou strategii na základě výpočtu úspěšnosti v právě dokončené učící periodě. Úspěšnost strategie S je vypočítána podle vztahu (8) a pravděpodobnosti p pro strategie podle vztahu (7). [3]

$$p_k = \frac{S_k}{\sum_{j=1}^4 S_j} \quad (7)$$

$$S_k = \frac{succ_k}{succ_k + fail_k} + \varphi \quad (8)$$

Hodnoty $succ_k$ (úspěchy) a $fail_k$ (neúspěchy) jsou vypočítány jako suma všech úspěšných (neúspěšných) použití příslušných strategií při tvorbě zkušebních vektorů během všech generací v průběhu právě dokončené učící periody. Logicky součet $succ_k + fail_k$ dává počet všech použití dané strategie v rámci učící periody. Hodnota φ nastavená na hodnotu 0.1 je

použitá pro případ, že by v průběhu učící periody nebyla daná strategie úspěšná ani jednou. Hodnoty pravděpodobností p jsou dále přepočítávány při každém dalším evolučním cyklu, přičemž je do učící periody přidána aktuální populace a je z ní odebrána populace nejstarší. Počet generací v učící periodě je konstantní – je dán jako vstupní parametr SaDE.

Mutační konstanta F je náhodně generována při tvorbě každého zkušebního vektoru. Náhodné generování F má v rámci SaDE charakter normálního rozdělení se střední hodnotou 0.5 a rozptylem 0.3.

Práh křížení CR je také generován jako náhodná hodnota z normálního rozložení $N(CR_{m_k}, 0.1)$, kde je $CR_{m_k} = 0.5$ pro všechny strategie během první učící periody. Adaptace CR spočívá v tom, že jsou uloženy všechny hodnoty CR, které se podílely na úspěšné tvorbě zkušebních vektorů v rámci učící periody, a použité strategie, do paměti CR_{mem_k} a pro další tvorbu nové populace je generována hodnota CR pro každou strategii opět z normálního rozložení $N(CR_{m_k}, 0.1)$. Hodnota CR_{m_k} , $k = 1, 2, 3, 4$ je však vypočítána jako medián z CR_{mem_k} . Dále jsou hodnoty CR_{mem_k} , CR_{m_k} i CR pro každou strategii aktualizovány po každém evolučním cyklu generace. [3]

3.4 JADE

Tato verze adaptivní DE rozšiřuje základní verzi DE o využití current-to-pbest mutační strategie, adaptivní řízení hodnot parametrů F a CR a o možnost využití archivu. Použití current-to-pbest spolu s možností archivu zajišťuje rychlou konvergenci JADE bez ztráty spolehlivosti. Šumový vektor je získáván podle výrazu (9) v případě, že není využíván archiv, nebo podle výrazu (10) v případě, že archiv využíván je.

$$v = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r1} - x_{r2}) \quad (9)$$

$$v = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r1} - \tilde{x}_{r2}) \quad (10)$$

Jedinec x_{pbest} je náhodně vybrán ze 100 p % nejlepších jedinců, kde platí, že $p \in (0, 1]$. Jedná-li se o variantu JADE bez archivu (9), jsou x_{r1} a x_{r2} náhodně vybraní jedinci z aktuální populace. V případě, že je realizována varianta s archivem (10), je x_{r1} náhodně vybraný jedinec z aktuální populace a \tilde{x}_{r2} je náhodně vybraný jedinec z množiny jedinců vzniklé spojením aktuální populace a archivu. Při zahájení JADE je archiv nastaven jako prázdná množina a v průběhu každého evolučního cyklu generace jsou do této množiny

přidání jedinci, kteří jsou do následující populace nahrazeni lepšími. Po ukončení evolučního cyklu generace je archiv, v případě že obsahuje více jedinců než NP, redukován na počet NP. Redukce počtu jedinců je provedena postupným náhodným odebíráním jedinců z archivu, dokud není počet jedinců roven NP.

Mutační konstanta F je adaptivně generována pro každý výpočet šumového vektoru. Mutační konstanta F je generována na základě Cauchyho rozdělení s parametrem polohy μ_F (11) a parametrem proměnlivosti 0.1. V případě, že je hodnota vygenerovaného $F > 1$, je nastavena na $F = 1$, a v případě, že je $F < 0$, je řídicí parametr F generován znovu.

$$\mu_F = (1 - c)\mu_F + c \cdot mean_l(S_F) \quad (11)$$

Hodnota c je řídicí parametr JADE a je z intervalu $[0,1]$. Hodnota $mean_l(S_F)$ je Lehmerův průměr počítaný podle následujícího výrazu:

$$mean_l(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \quad (12)$$

kde S_F je množina všech hodnot F , které byly v rámci generace úspěšné.

Práh křížení CR je, stejně jako parametr F , adaptivně generován pro každé křížení. Hodnota CR je náhodně generována z normálního rozložení $N(\mu_{CR}, 0.1)$, kde hodnota μ_{CR} je vypočítána následujícím výrazem:

$$\mu_{CR} = (1 - c)\mu_{CR} + c \cdot mean_A(S_{CR}) \quad (13)$$

kde S_{CR} je množina všech hodnot CR, které byly v rámci generace úspěšné, a výraz $mean_A(S_{CR})$ je aritmetickým průměrem množiny S_{CR} .

Výchozí hodnoty μ_F a μ_{CR} jsou nastaveny na 0.5 a jsou aktualizovány na konci každého generačního cyklu. [3]

3.5 EPSDE

Pro tuto variantu adaptivní DE je charakteristické, že součástí každého jedince je trojice řídicích parametrů (vybraná strategie, mutační konstanta F , práh křížení CR). Do těchto

trojic jsou vybírány hodnoty z množin dostupných hodnot (Tab. 2), definovaných pro každý parametr.

Tab. 2 Ukázka nastavení množin dostupných hodnot pro nastavení řídicích parametrů EPSDE [8]

Strategie	{ best/2/bin, rand/1/bin, current-to-rand/1 }
F	{ 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 }
CR	{ 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 }

Při vytváření první populace jsou vybrány trojice řídicích parametrů z nastavených množin náhodně. V průběhu evoluce je vždy, když se aktivní jedinec podílí na sestavení úspěšného zkušebního vektoru, přidána trojice řídicích parametrů z aktivního jedince a tím se úspěšná trojice dostává do nové populace. Úspěšná trojice je také uložena do množiny, jejíž velikost je dána jako parametr EPSDE. V případě, že je do nové populace přesunut aktivní jedinec (zkušební vektor nebyl úspěšný), je pro aktivního jedince vygenerovaná nová trojice hodnot řídicích parametrů z nastavených množin nebo je přiřazena aktivnímu jedinci jedna trojice z množiny úspěšných trojic. Obě metody výběru nové trojice řídicích parametrů pro aktivního jedince mají stejnou pravděpodobnost, že budou zvoleny.

4 GENEROVÁNÍ NÁHODNÝCH ČÍSEL

Z předchozích kapitol popisujících EVT, DE i adaptivní DE je z popisu jednotlivých kroků těchto algoritmu zřejmé, že proces generování náhodné číselné hodnoty v definovaném rozmezí je velmi častým jevem a ideální by pravděpodobně bylo, kdyby generované hodnoty byly skutečně náhodné.

Sekvence skutečně náhodných čísel by byla taková, kde by jednotlivé hodnoty byly zcela statisticky nezávislé. V počítačových systémech však není možné vytvořit algoritmus, který by takovou sekvenci statisticky nezávislých dat produkoval. Místo toho je používán tzv. generátor pseudonáhodných čísel, což je algoritmus, který generuje hodnoty vypadající pro pozorovatele, který nezná generující algoritmus, zcela náhodně. Takový algoritmus však při stejném nastavení vstupních parametrů generuje stejnou sekvenci pseudonáhodných čísel. Tento fakt může být například u ladění algoritmů či porovnávání algoritmů i výhodou. Pro reálné aplikace, například v kryptografii nebo při simulaci skutečně náhodného modelu, je taková vlastnost generátoru nežádoucí. Pro generování náhodných čísel je také možné využít chaotických systémů, které se projevují nepravidelným chováním.

4.1 Pseudonáhodná čísla

Generátory pseudonáhodných čísel vytváří na základě vstupních hodnot, tzv. semínka (seed), a definovaného algoritmu daného generátoru, dlouhé posloupnosti zdánlivě náhodných hodnot. V případě, že se v rámci posloupnosti vyskytne hodnota, která již v posloupnosti hodnot byla, jsou další hodnoty v posloupnosti periodicky opakovány. Pro kvalitní generátor pseudonáhodných čísel je důležité, aby opakovací perioda byla co nejdelší. Kvalita generátoru náhodných čísel bývá také ověřována pomocí metod statistické analýzy. Nejrozšířenějším generátorem pseudonáhodných čísel, který byl původně použit jako algoritmus pro implementaci základní funkce pro generování náhodného čísla (rnd) ve většině programovacích jazyků, je tzv. Lineární kongruenční generátor, který generuje posloupnost pseudonáhodných hodnot podle následujícího algoritmu:

$$\begin{aligned}x_{n+1} &= (ax_n + c) \bmod m, \\ m > 0 \wedge 0 \leq a < m \wedge 0 \leq c < m \wedge 0 \leq x_0 < m\end{aligned}\tag{14}$$

V rámci tohoto algoritmu platí, že perioda nemůže být delší než m . Při použití tohoto algoritmu je tedy třeba volit dostatečně velkou hodnotu m . Posloupnosti pseudonáhodných čísel generovaných tímto algoritmem mají však předvídatelný charakter. Existují i další generátory pseudonáhodných čísel s různou náročností výpočtu a kvalitou generovaných posloupností, čímž je dáno, že různé generátory jsou vhodné na specifický typ úloh. Mezi další generátory náhodných čísel patří například Mersenne Twister, Blum Blum Shub a další. [9]

4.2 Chaotické systémy

Chaotické systémy jsou modely deterministického chaosu, které jsou realizovány pomocí soustav diferenciálních rovnic. Chaotické systémy mohou být spojité nebo diskrétní. Pro řešení problematiky náhodných čísel jsou vhodné diskrétní chaotické systémy. Atributy deterministického chaosu jsou:

- velká citlivost i na malé změny počátečních podmínek
- systém je lokálně nestabilní avšak globálně stabilní
- chyby v měření počátečních podmínek exponenciálně narůstají – není možné předpovídat chování dynamického systému
- chaotické systémy jsou ze své nelineární, lze je však matematicky popsat. [10]

V systémech deterministického chaosu se dva nekonečně blízké stavy exponenciálně rozcházejí. Tento děj se nazývá bifurkací. Jedná se o náhlou změnu stability, která chaosu předchází. V případě, kdy v soustavě dochází ke vzniku na sebe navazujících bifurkací, říkáme, že nastává chaos. Díky tomuto jevu generuje chaotický systém reprezentovatelný sadou diferenciálních rovnic (Tab. 3) při mírně odlišných počátečních podmínkách zcela odlišné řady, pro pozorovatele náhodných, čísel. Charakter náhodných číselných řad však tyto systémy generují pouze s určitým nastavením parametrů.

Tab. 3 Chaotické systémy

Název	Zápis	Parametry
Lozi map	$X_{n+1} = 1 + a X_n + bY_n$ $Y_{n+1} = X_n$	$a = 1,7$ $b = 0,5$ (15)
Sinai	$X_{n+1} = X_n + Y_n + \delta \cos 2\pi Y_n (\text{mod } 1)$ $Y_{n+1} = X_n + 2Y_n (\text{mod } 1)$	$\delta = 0,1$ (16)
Disipative	$X_{n+1} = X_n + Y_{n+1} (\text{mod } 2\pi)$ $Y_{n+1} = bY_n + k \sin X_n (\text{mod } 2\pi)$	$b = 0,1$ $k = 8,8$ (17)
Ikeda	$X_{n+1} = \gamma + \mu(X_n \cos \omega - Y_n \sin \omega)$ $Y_{n+1} = \mu(X_n \sin \omega + Y_n \cos \omega)$ $\omega = \beta - \frac{\alpha}{1 + X_n^2 + Y_n^2}$	$\alpha = 6$ $\beta = 0,4$ $\gamma = 1$ $\mu = 0,9$ (18)
Arnold cat	$X_{n+1} = X_n + Y_n (\text{mod } 1)$ $Y_{n+1} = X_n + kY_n (\text{mod } 1)$	$k = 2$ (19)
Hénon map	$X_{n+1} = 1 - aX_n^2 + bY_n$ $Y_{n+1} = X_n$	$a = 1,4$ $b = 0,3$ (20)
Burgers map	$X_{n+1} = aX_n - Y_n^2$ $Y_{n+1} = bY_n + X_n Y_n$	$a = 0,75$ $b = 1,75$ (21)
Holmes cubic map	$X_{n+1} = Y_n$ $Y_{n+1} = -bX_n + dY_n - Y_n^3$	$b = 0,2$ $d = 2,77$ (22)
Chirikov	$X_{n+1} = X_n + Y_{n+1} (\text{mod } 2\pi)$ $Y_{n+1} = Y_n + k \sin X_n (\text{mod } 2\pi)$	$k = 1$ (23)

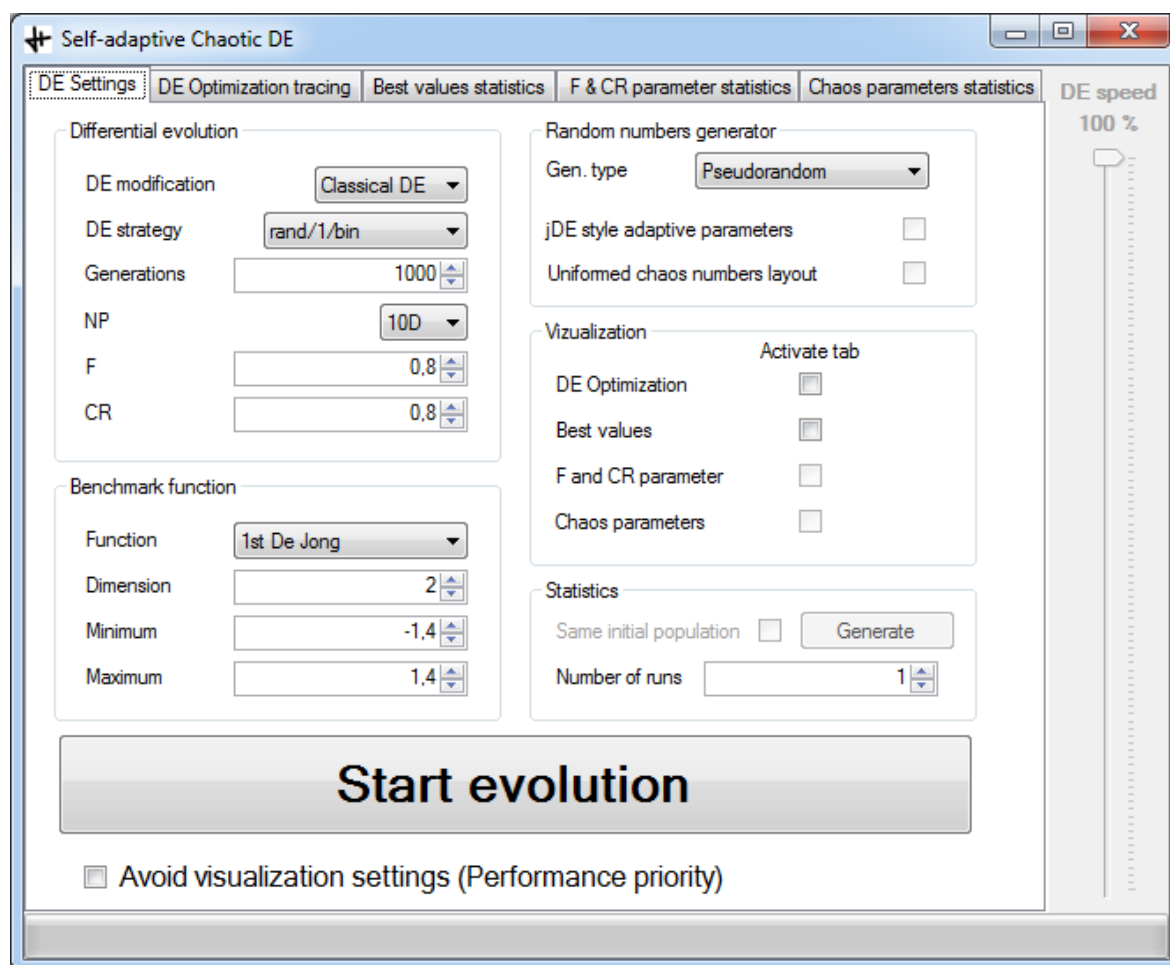
II. PRAKTICKÁ ČÁST

5 APLIKACE SELF-ADAPTIVE CHAOTIC DE

Pro implementaci aplikace realizující různé verze DE jsem, vzhledem k předchozím zkušenostem v oblasti frameworku .NET, zvolil programovací jazyk C# a vývojové prostředí Visual Studio 2012. Program je navržen jako klasická formulářová aplikace používající standardní ovládací prvky, které jsou součástí frameworku .NET.

V rámci aplikace je možné uživatelsky nastavit typ DE, strategii, parametry DE i různé verze generátoru náhodných čísel. Aplikace také umožňuje spustit DE v daném nastavení v sérii uživatelsky nastaveným počtem opakování a poté zobrazí statistické vyhodnocení série.

Programová implementace je realizována pomocí objektového přístupu a návrh umožňuje další snadné rozšíření o různé kombinace známých DE či další testovací nebo reálné optimalizační funkce.



Obr. 4 Základní GUI aplikace

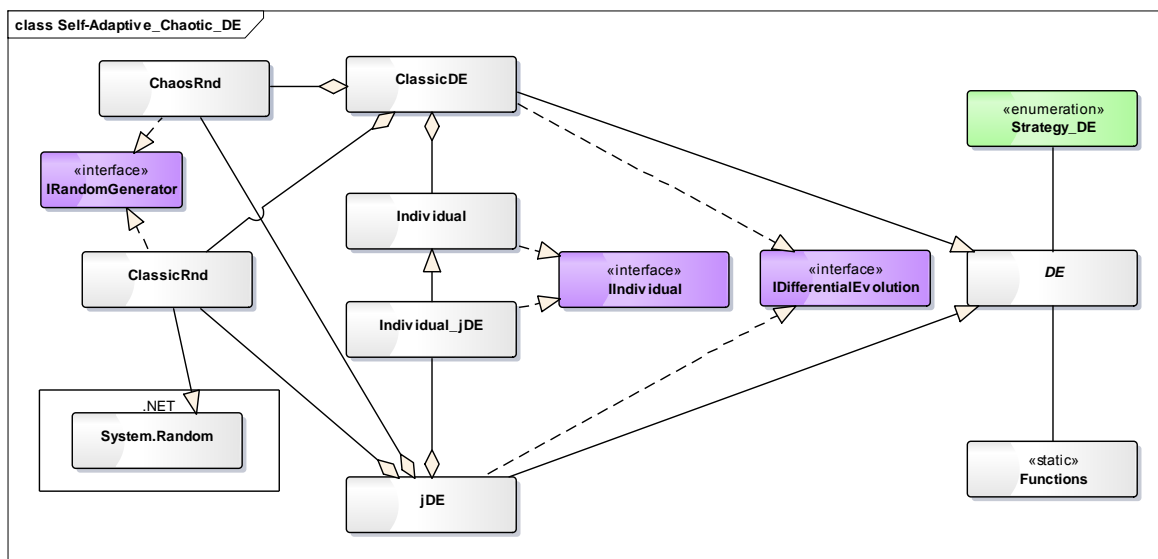
5.1 Objektový návrh aplikace

Objektový návrh aplikace je realizován s výhledem možného budoucího rozšíření o další verze DE. Základní schéma vychází z vytvoření rozhraní pro základní typy objektů, které jsou:

- jedinec – rozhraní **IIndividual**
- generátor náhodných čísel – rozhraní **IRandomGenerator**
- diferenciální evoluce – rozhraní **IDifferentialEvolution**

Díky těmto rozhráním je možné následně programovat aplikaci nezávisle na tom, která konkrétní verze DE, realizovaná třídou implementující IDifferentialEvolution, je použita. Stejně tak je díky rozhráním IIndividual a IRandomGenerator možné snadno realizovat různé kombinace technologií uvnitř implementací různých druhů diferenciálních evolucí.

Základní struktura tříd, bez návaznosti na implementaci grafického rozhraní aplikace, je znázorněna na Obr. 5.



Obr. 5 Diagram tříd aplikace bez GUI

5.1.1 Rozhraní pro implementaci tříd

IRandomGenerator – ve všech třídách, v rámci implementace aplikace, je definováno, že objekt, který realizuje generování náhodných čísel, musí implementovat toto rozhraní. V rámci rozhraní je definována metoda **NextDouble()**, která by měla generovat jako výsledek náhodné reálné číslo v rozmezí [0, 1]. Dále je definována metoda **Mutate()**, která při implementaci slouží k náhodnému vygenerování řídicích parametrů generátoru

náhodných čísel (pokud nějaké má), a vlastnost **IsAdaptive**, která zveřejňuje, jestli instance generátoru náhodných čísel byla vytvořena jako adaptivní – umožňuje změnu řídicích parametrů.

Individual – vzhledem k tomu, že všechny implementované varianty DE jsou třídy, které jsou specializací abstraktní třídy **DE**, je v rámci implementace této základní třídy využíváno toto rozhraní pro realizaci společné funkcionality. Jsou definovány následující metody:

- **Evaluate(CostFunction): double** – slouží k výpočtu hodnoty účelové funkce nad parametry jedince. Tvar účelové funkce je definován pomocí delegátu **CostFunction**, který umožňuje použít jako vstupní parametr libovolnou funkci, která má jako vstupní parametr pole hodnot typu double a výsledkem je hodnota opět typu double. Tím je zajištěna flexibilita při potenciálním rozšiřování aplikace.
- **TruncateValues(): void** – realizuje úpravu hodnot parametrů tak, aby byly vždy v rozmezí stanovených maximálních a minimálních hodnot.
- **RegenerateParams(IRandomGenerator): void** – slouží pro implementaci změny řídicích parametrů v adaptivních verzích DE, kdy jsou parametry F a CR součástí jedinců.

Dále jsou definovány vlastnosti **CostValue: double**, pro čtení poslední získané hodnoty účelové funkce pomocí metody **Evaluate**, a **Vector: double[]** pro získání všech hodnot parametrů jedince.

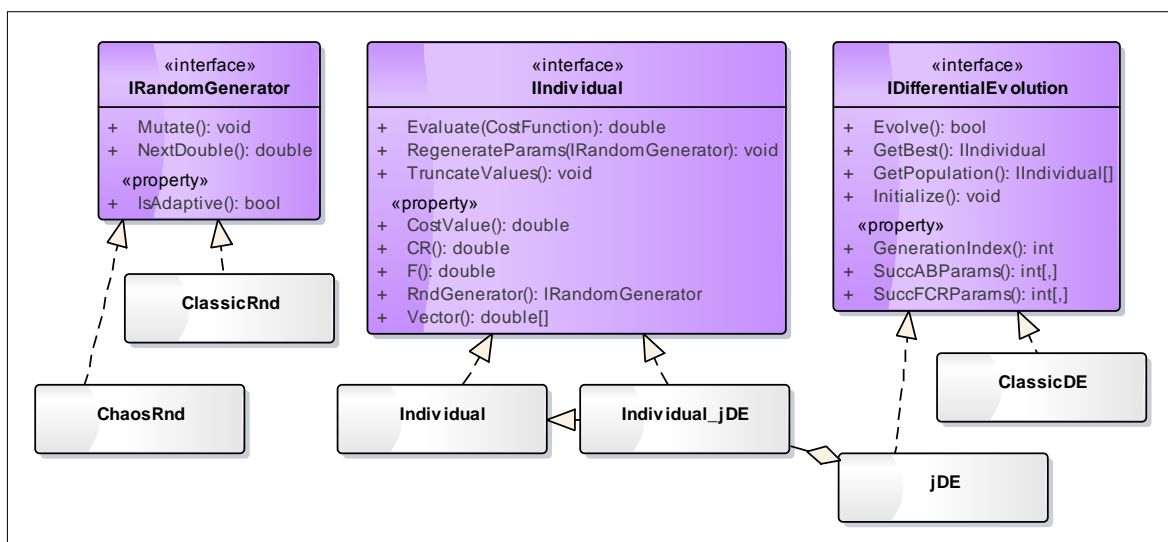
Další vlastnosti **CR: double**, **F: double** a **RndGenerator: IRandomGenerator** jsou využity pro implementaci adaptivních verzí DE a slouží k čtení řídicích parametrů a generátoru náhodných čísel přidruženého ke konkrétnímu jedinci.

IDifferentialEvolution – definuje základní společný rámec metod a vlastností, které by měly být společné pro implementace různých druhů DE. Rozhraní obsahuje následující metody:

- **Initialize(): void** – slouží k vytvoření první populace rodičů v DE
- **Evolve(): bool** – realizuje jeden evoluční cyklus implementované DE. Návrátová hodnota slouží pro zjištění, zda DE ještě pokračuje dále nebo již skončila.
- **GetBest(): Individual** – slouží pro získání nejlepšího jedince z aktuální populace v DE. Návrátová hodnota je opět definována rozhraním pro maximální obecnost a pružnost návrhu.

- **GetPopulation(): IIndividual[]** – slouží pro získání všech jedinců v aktuální generaci.

V rámci tohoto rozhraní jsou definovány i další vlastnosti. **GenerationIndex: int** slouží ke zveřejnění indexu identifikující kolikátá je aktuální generace jedinců v rámci průběhu evolučních cyklů DE. Vlastnosti **SuccABParams: int[]** a **SuccFCRParams: int[,]** jsou sice obecně definovány již v rozhraní, ale slouží pro statistické zpracování hodnot řídicích parametrů DE a generátorů náhodných čísel u adaptivních verzí. Na následujícím obrázku (Obr. 6) jsou znázorněny detaily všech uvedených rozhraní a schematicky implementační třídy.



Obr. 6 Schéma rozhraní a implementačních tříd

5.1.2 Implementace tříd

Třída ClassicRnd – je specializací třídy `System.Random` z frameworku .NET, která slouží pro generování pseudonáhodných čísel. Implementuje rozhraní **IRandomGenerator** a instance této třídy jsou použity jako jedna z možných variant způsobu generování náhodných čísel v procesu všech DE.

Třída ChaosRnd – implementuje rozhraní **IRandomGenerator**. Vnitřní implementace generování náhodných čísel je postavena na základě chaotického systému **Lozi map**.

$$\begin{aligned}
 X_{n+1} &= 1 + a|X_n| + bY_n \\
 Y_{n+1} &= X_n
 \end{aligned}
 \tag{24}$$

V rámci konstruktoru třídy jsou parametry a výchozí podmínky, jejichž význam je znázorněn předchozím výrazem (24), nastaveny na následující hodnoty:

- $a = 1.7$
- $b = 0.5$
- $X_0 = -0.1 - r, r \in [0, 0.05]$
- $Y_0 = +0.1 + r, r \in [0, 0.05]$

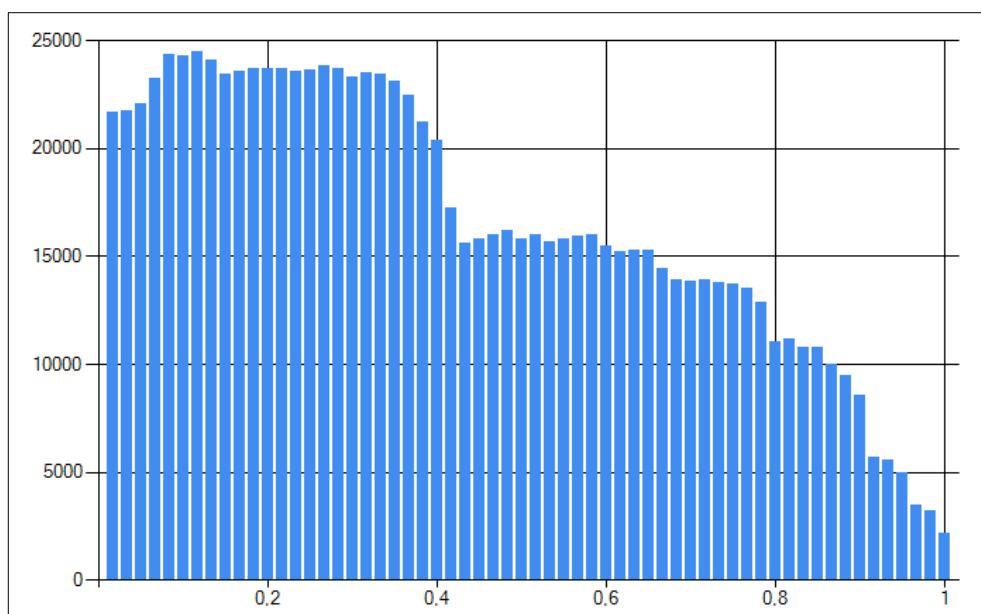
Třída pro generování hodnot a manipulaci s parametry a, b používá následující metody:

SetPool(): void – nastavuje množinu možných variant (Obr. 7) nastavení parametrů a, b , u kterých bylo prokázáno chaotické chování [11]. Z této množiny jsou pak náhodně vybírány kombinace parametrů v případě adaptivního nastavení DE pro parametry chaotického generátoru hodnot.

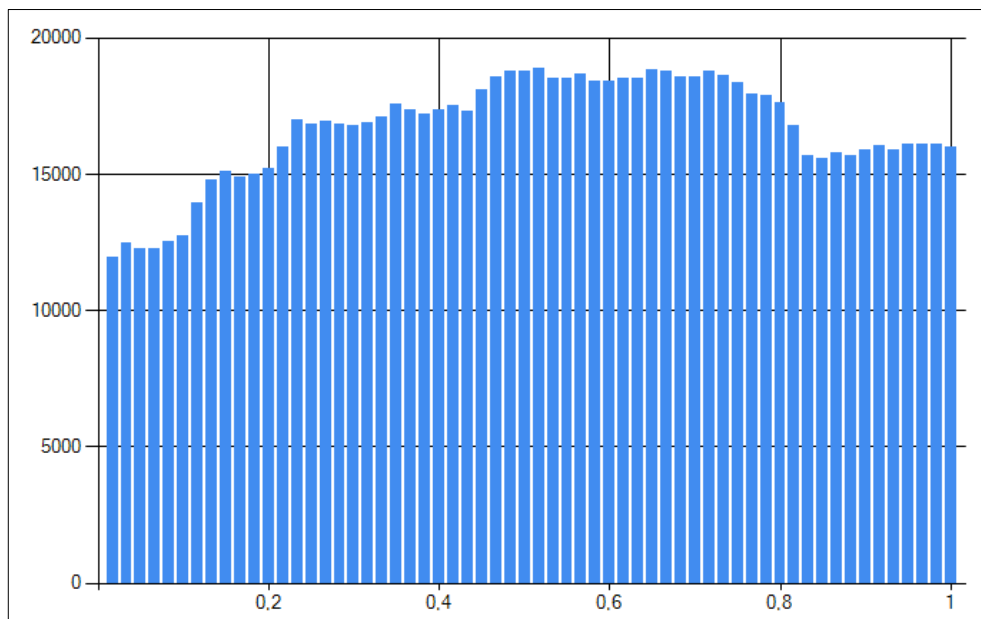
DE Version	Original Schwefel's test function	shifted 1 st De Jong's function	shifted Ackley's original function	shifted Rastrigin's function
a=1.3 b=0.1	-6323.84	66.93173	17.35857	208.913
a=1.3 b=0.2	-9760.51	5.001269	9.939031	67.4757
a=1.4 b=0.1	-11774.1	0.042298	1.731801	24.85462
a=1.4 b=0.2	-12228.6	4.05E-19	2.05E-10	16.45284
a=1.4 b=0.3	-12412.9	3.56E-28	0.111731	15.83748
a=1.5 b=0.1	-11462.4	0	4E-15	95.21326
a=1.5 b=0.2	-11479.9	0	4E-15	97.04095
a=1.5 b=0.3	-11290.4	0	4E-15	97.93962
a=1.5 b=0.4	-12070.2	0	4E-15	89.73319
a=1.6 b=0.1	-11768.9	0	4.12E-15	98.31299
a=1.6 b=0.2	-11677.5	0	4E-15	101.3721
a=1.6 b=0.3	-11759	0	4E-15	100.9559
a=1.6 b=0.4	-11000.6	0	3.88E-15	108.1069
a=1.6 b=0.5	-10602.5	0	4E-15	105.949
a=1.7 b=0.1	-11966.2	0	4.12E-15	98.42761
a=1.7 b=0.2	-10990.7	0	4E-15	102.5946
a=1.7 b=0.3	-10618	0	4E-15	107.0337
a=1.7 b=0.4	-9866.49	0	4E-15	111.6011
a=1.7 b=0.5	-9058.77	0	4E-15	114.8259

Obr. 7 Vhodné nastavení parametrů pro Lozi map [11]

NextDouble(): double – v rámci této metody je v závislosti nastavení generátoru buď načtena další hodnota z předem vygenerované posloupnosti, nebo je aktuálně vypočtena z předchozích hodnot X_n, Y_n . Získaná náhodná hodnota je poté upravena do intervalu $[0, 1]$. Protože statisticky Lozi map generuje hodnoty v intervalu $[0, 1]$ rozložené více vlevo (Obr. 8), je možné programově zapnout optimalizaci generovaných hodnot, která probíhá úpravou hodnot do intervalu $[0, 2]$ a poté převrácením hodnot v části intervalu $[1, 2]$ zpět do intervalu $[0, 1]$ osově přes hodnotu 1. Rozložení takto upravených hodnot je více vyrovnané (Obr. 9) a při použití u některých verzí DE generuje lepší výsledky.



Obr. 8 Histogram hodnot systému Lozi map přepočtených do $[0, 1]$

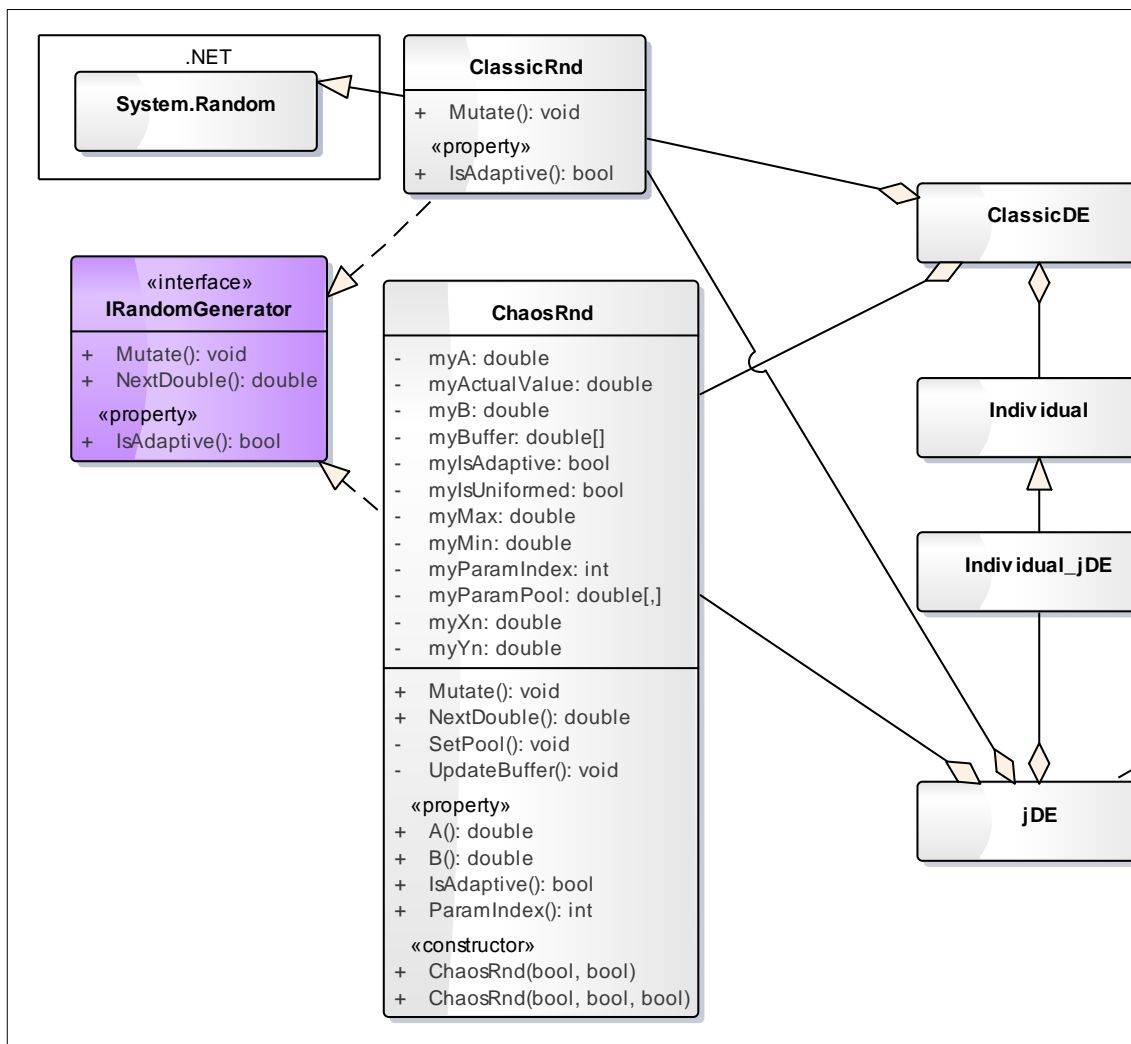


Obr. 9 Histogram hodnot systému Lozi map přepočtených do $[0, 1]$ po optimalizaci

UpdateBuffer(): void – realizuje vygenerování posloupnosti dalších 10000 čísel do připraveného pole, ze kterého jsou následně načítány v metodě NextDouble, pokud to odpovídá nastavení generátoru.

Mutate(): void – v případě, že je generátor nastaven jako adaptivní, vybere náhodně z množiny možných nastavení nové hodnoty a přiřadí je do parametrů a, b .

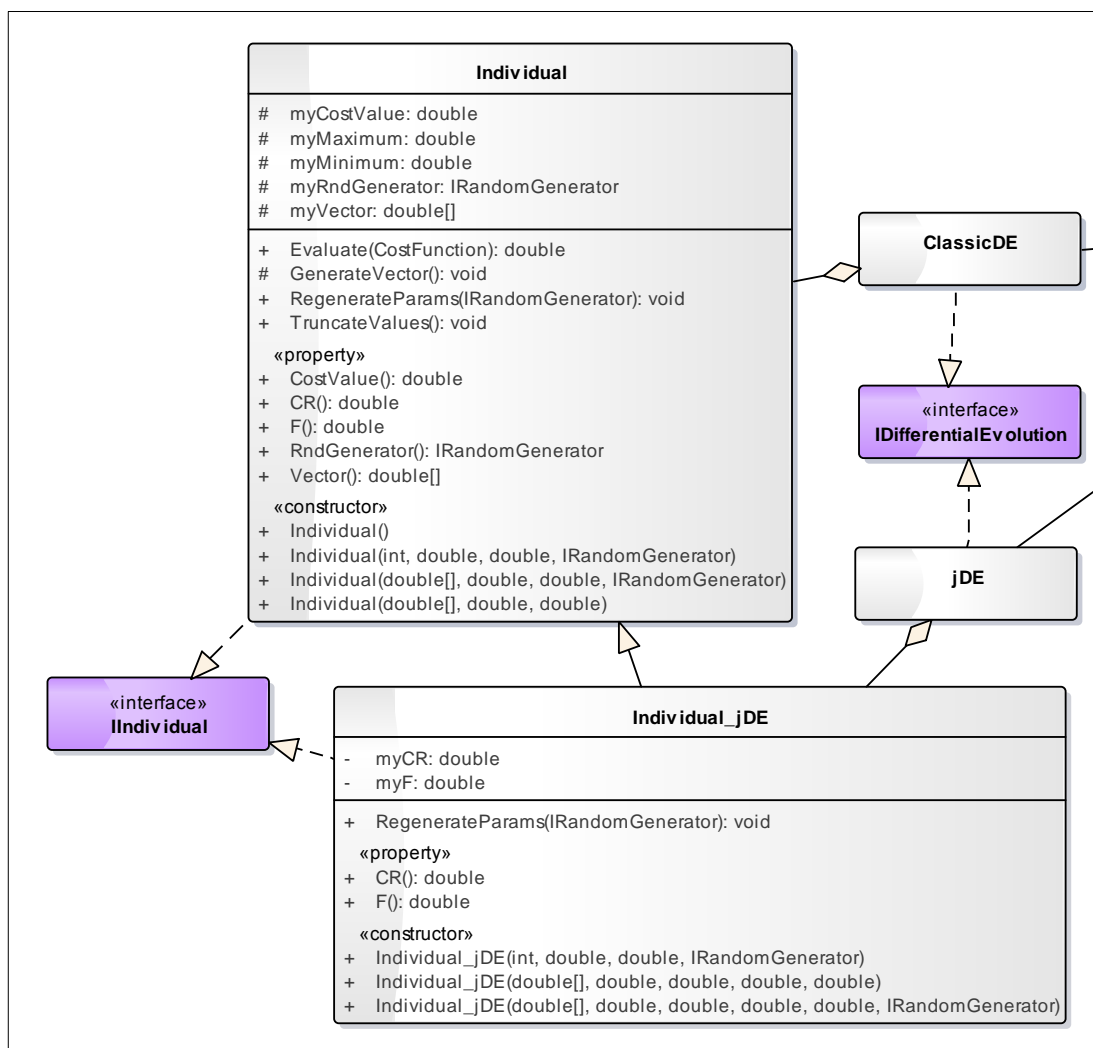
Vlastnosti zveřejňují pro čtení nastavení generátoru. **A: double** a **B: double** hodnoty parametrů generátoru, **IsAdaptive: bool** informaci, zda je generátor nastaven jako adaptivní a **ParamIndex: int** index identifikující, která kombinace parametrů je právě nastavená, což slouží především pro statistické vyhodnocení na celé DE při adaptivním nastavení generátoru. Následující schéma (Obr. 10) zobrazuje detaily tříd pro generátory náhodných čísel v rámci části diagramu tříd.



Obr. 10 Zobrazení detailů tříd generátorů náhodných čísel

Třída Individual – implementuje rozhraní **IIndividual** a reprezentuje jedince klasické verze DE s možností volitelného uložení generátoru náhodných čísel pro možnost adaptivní změny parametrů objektů třídy **ChaosRnd**. Mimo implementaci všech metod a vlastností obsahuje třída **Individual** ještě metodu **GenerateVector(): void**, která slouží k vytvoření výchozích náhodných hodnot parametrů jedince podle nastavených omezení.

Třída Individual_jDE – dědí všechny metody a vlastnosti z třídy **Individual** a také implementuje rozhraní **IIndividual**. Rozdílem oproti rodičovské třídě **Individual** je doplnění proměnných pro uložení adaptivních parametrů **F** a **CR**, přepsání metody **RegenerateParams(IRandomGenerator): void** a přepsání vlastností **F: double** a **CR: double**. Následuje schéma zobrazující detaily těchto dvou implementovaných tříd reprezentujících jedince různých verzí DE (Obr. 11).



Obr. 11 Detaily tříd implementujících jedince DE v části diagramu tříd

Třída DE – je abstraktní a tvoří společný základ pro implementaci všech typů DE. Definuje abstraktní metody **Initialize(): void** a **Evolve(): bool**, které korespondují i s rozhraním **IDifferentialEvolution** a které je nutné realizovat v každé implementaci DE, jelikož reprezentují základní kroky DE. Metody reprezentující mutaci jsou přímo implementovány, je však možné je v děděných třídách přepsat. Jedná se o metody:

- **Rand1(IIndividual, int): double[]**
- **Best1(IIndividual, int): double[]**
- **RandToBest1(IIndividual, int): double[]**
- **Rand2(IIndividual, int): double[]**
- **Best2(IIndividual, int): double[]**

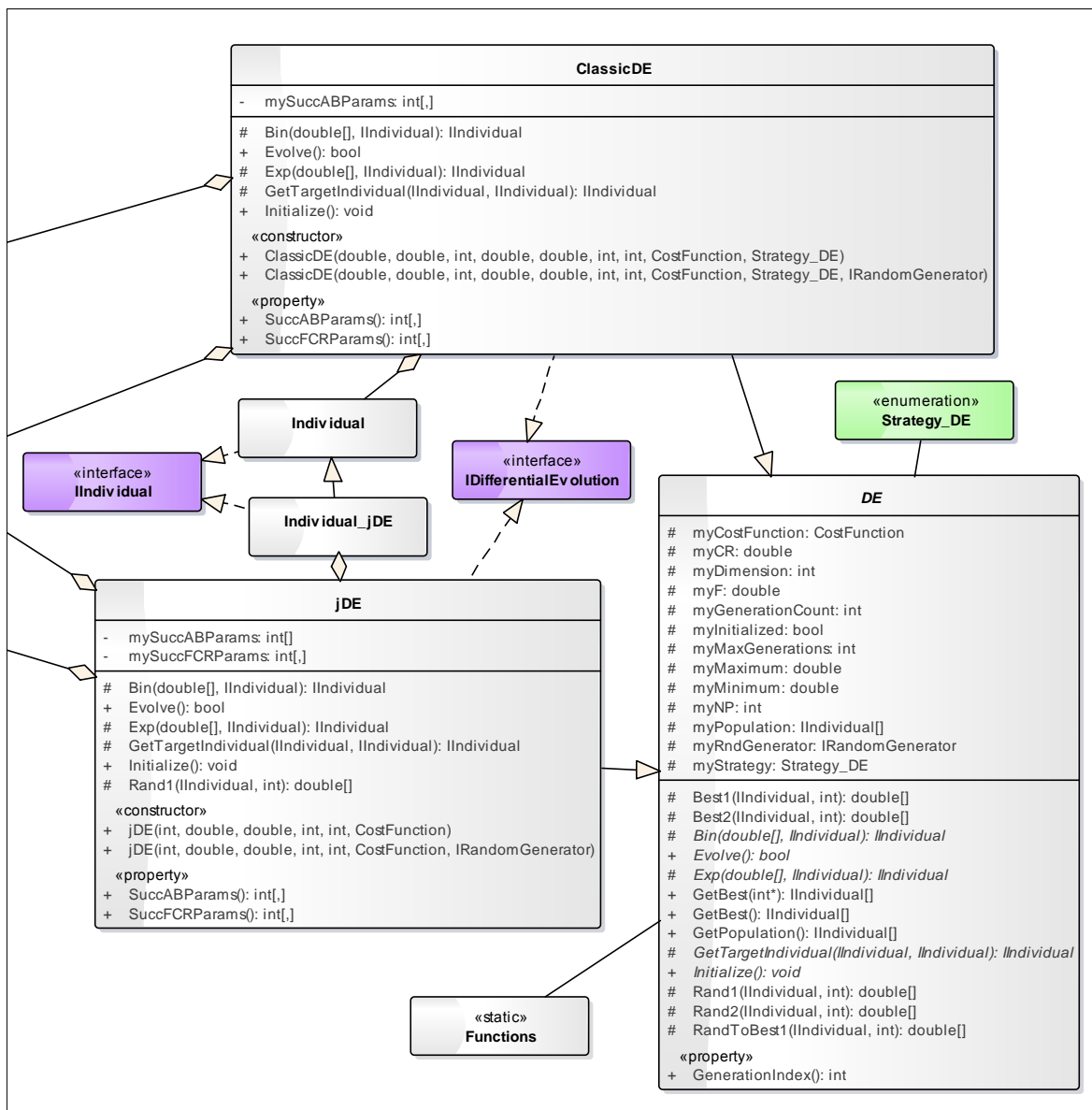
Tyto metody implementují k názvům adekvátní strategie výpočtu šumového vektoru DE. Dále abstraktně definuje metody **Bin(double[], IIndividual): IIndividual** a **Exp(double[], IIndividual): IIndividual**, které v třídách potomků budou reprezentovat adekvátní strategie křížení DE. V rámci této abstraktní třídy jsou již implementovány metody a vlastnosti, které budou potřeba v děděných třídách při implementaci rozhraní **IDifferentialEvolution** a to:

- **GetPopulation(): IIndividual[]**
- **GetBest(): IIndividual**
- **GenerationIndex: int**
- **GetTargetIndividual(IIndividual, IIndividual): IIndividual** – realizuje výběr jedince do nové populace podle principů základní verze DE.

V rámci třídy **DE** jsou také definovány privátní proměnné, které budou potřeba ve všech verzích děděných tříd, reprezentujících varianty DE. Tyto proměnné je možné vidět na schématu tříd reprezentujících varianty DE (Obr. 12).

Třída ClassicDE – je specializací třídy **DE** a zároveň implementuje rozhraní **IDifferentialEvolution**. Realizovány jsou pouze abstraktní metody z třídy **DE** a metody a vlastnosti, které implementují **IDifferentialEvolution** (Obr. 12).

Třída jDE – je také specializací třídy **DE** a také implementuje rozhraní **IDifferentialEvolution**. Vnitřní realizace abstraktních metod z třídy **DE** je však odlišná oproti třídě **jDE**, protože zahrnuje princip adaptivnosti řídicích parametrů **F** a **CR**. Vzhledem k tomu, že v rámci **jDE** je používána mutační strategie **Rand1**, je v rámci této třídy příslušným způsobem metoda **Rand1(IIndividual, int): double[]** přepsána. Je přepsána také metoda **GetTargetIndividual(IIndividual, IIndividual): IIndividual**, protože **jDE** pracuje, na rozdíl od klasické DE, v rámci procesu výběru nového jedince s parametry **F** a **CR** uvnitř jedinců. Tato třída je také doplněna o privátní proměnné, které slouží pro statistické vyhodnocování hodnot řídicích parametrů **F** a **CR** (Obr. 12).

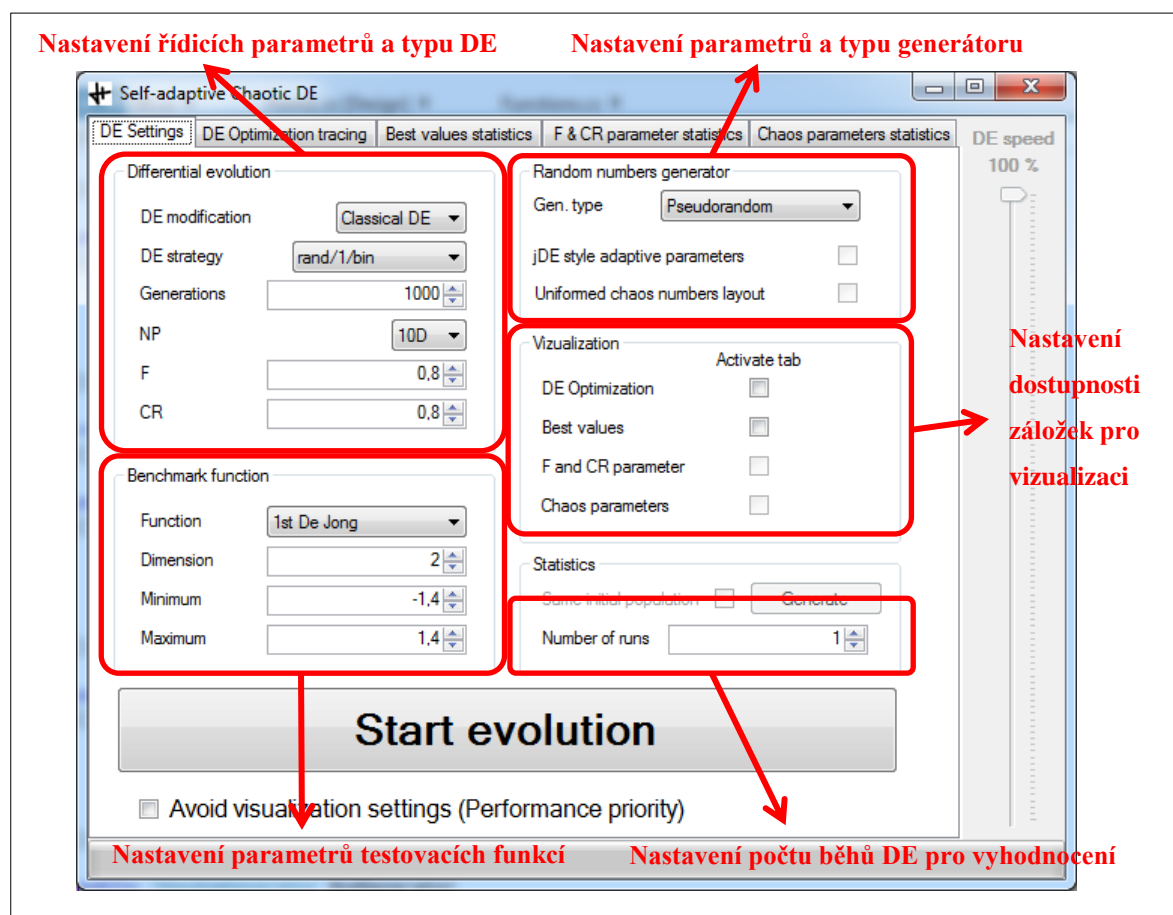


Obr. 12 Detaily tříd implementujících varianty DE v části diagramu tříd

Třída Functions – je statická a obsahuje implementace testovacích funkcí, které odpovídají definovanému delegátu CostFunction a slouží pouze k systematickému uložení těchto funkcí v rámci návrhu tříd. Obsahuje tyto testovací funkce: První de Jongova funkce, Druhá de Jongova funkce, Třetí de Jongova funkce, Čtvrtá de Jongova funkce, Rastriginova funkce, Schwefelova funkce, Griewangkova funkce, Sinová obálková sinusoidální funkce, Roztažená sinusoidální V funkce, Ackleyho funkce I, Ackleyho funkce II, Michalewiczova funkce a Mastersova funkce. Libovolné další testovací funkce je možné snadno doplnit.

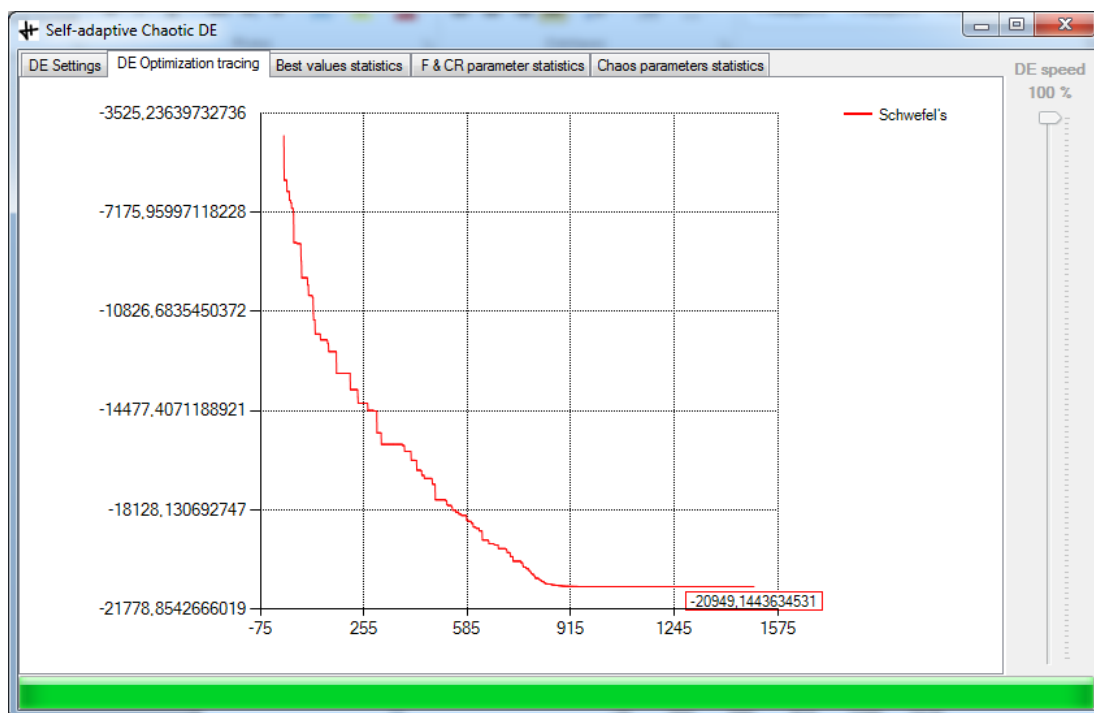
5.2 Ovládání aplikace

Aplikace je ovládána pomocí standardních ovládacích prvků formulářových aplikací. Grafické rozhraní umožňuje nastavení řídicích parametrů DE, omezení pro nastavení parametrů jedinců i způsob vizualizace průběhu evolučních cyklů (Obr. 13). Ovládací prvky jsou aktivní pouze v případě, že je to v aktuálním nastavení logické.



Obr. 13 Nastavení DE v rámci GUI aplikace.

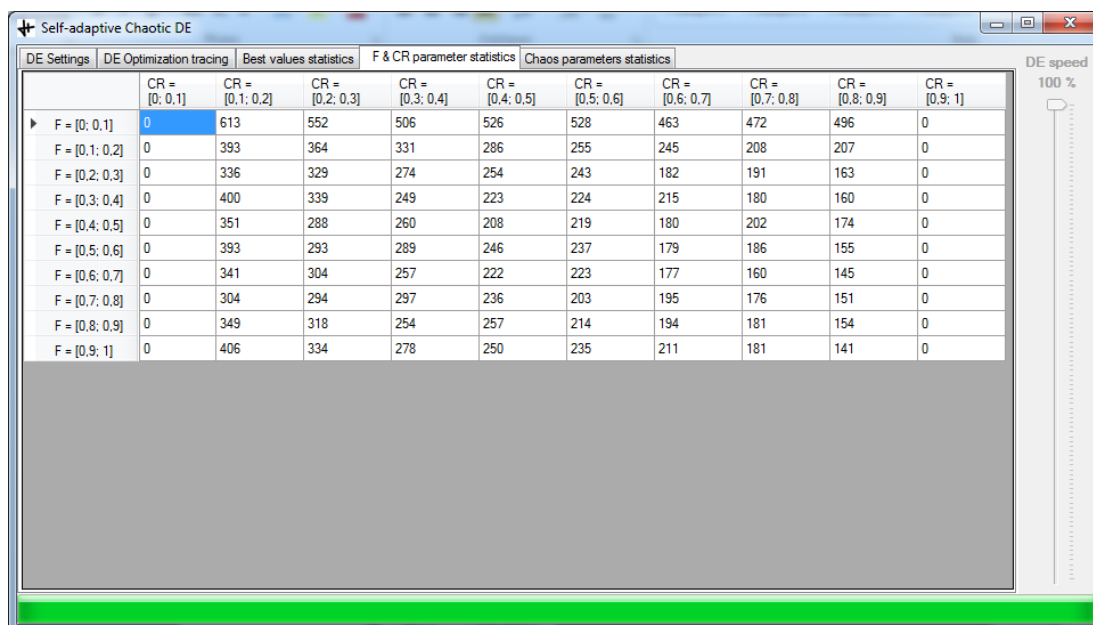
Samotný průběh DE je implementován v samostatném vlákne, takže vizualizace průběhu optimalizace grafem se mění v průběhu DE (Obr. 14). Aplikace také umožňuje základní tabulkové vyhodnocení četnosti výskytu hodnot řídicích parametrů (Obr. 16), v případě adaptivního nastavení DE nebo generátoru náhodných čísel (Obr. 17), po uživatelsky zvoleném počtu běhů DE. Dále aplikace zobrazuje v tabulce hodnoty výsledku optimalizace pro jednotlivé běhy v sérii a vypočítává průměrnou a nejlepší hodnotu série výsledků (Obr. 15).



Obr. 14 Vizualizace průběhu DE v grafu

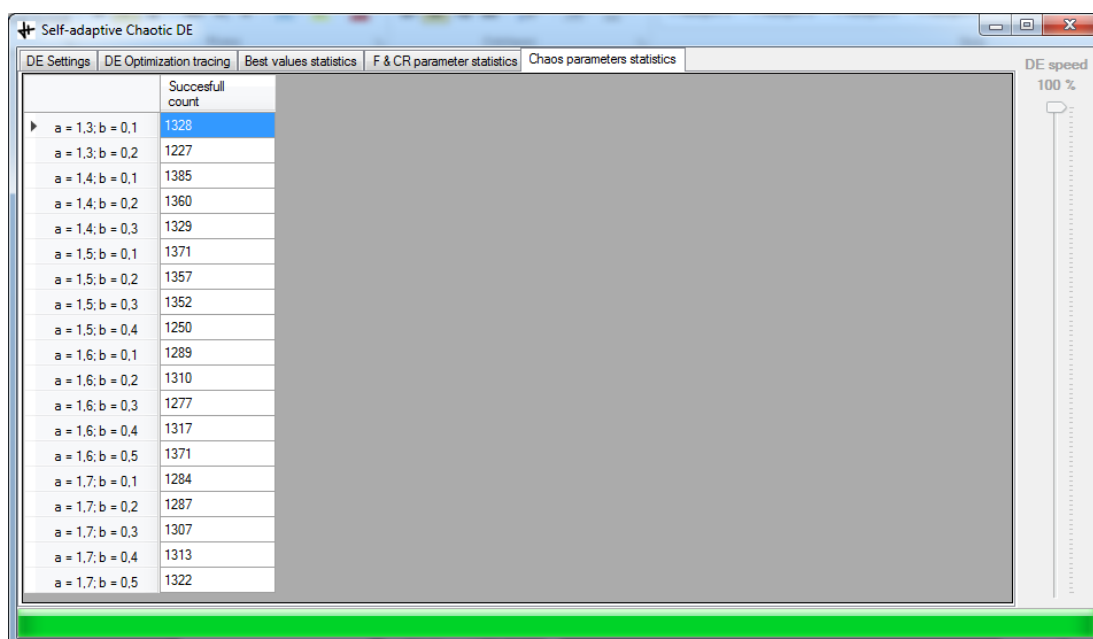
	Result of run
Gen n.1	-2266.13089857...
Gen n.2	-2505.59039590...
Gen n.3	-2449.65628538...
Gen n.4	-2441.29588099...
Gen n.5	-2233.82491828...
Gen n.6	-2274.58056353...
Gen n.7	-2157.97238504...
Gen n.8	-2356.28082211...
Gen n.9	-2531.89158988...
Gen n.10	-2522.81132375...
Gen n.11	-2295.85911969...
Gen n.12	-2475.23885828...
Avg value	-2375.92775345...
Best value	-2531.89158988...

Obr. 15 Tabulkové zobrazení výsledků optimalizací a jejich vyhodnocení



DE Settings	CR = [0; 0.1]	CR = [0.1; 0.2]	CR = [0.2; 0.3]	CR = [0.3; 0.4]	CR = [0.4; 0.5]	CR = [0.5; 0.6]	CR = [0.6; 0.7]	CR = [0.7; 0.8]	CR = [0.8; 0.9]	CR = [0.9; 1]
F = [0; 0.1]	0	613	552	506	526	528	463	472	496	0
F = [0.1; 0.2]	0	393	364	331	286	255	245	208	207	0
F = [0.2; 0.3]	0	336	329	274	254	243	182	191	163	0
F = [0.3; 0.4]	0	400	339	249	223	224	215	180	160	0
F = [0.4; 0.5]	0	351	288	260	208	219	180	202	174	0
F = [0.5; 0.6]	0	393	293	289	246	237	179	186	155	0
F = [0.6; 0.7]	0	341	304	257	222	223	177	160	145	0
F = [0.7; 0.8]	0	304	294	297	236	203	195	176	151	0
F = [0.8; 0.9]	0	349	318	254	257	214	194	181	154	0
F = [0.9; 1]	0	406	334	278	250	235	211	181	141	0

Obr. 16 Příklad zobrazení četnosti parametrů F a CR v rámci několika běhů DE



DE Settings	Successful count
a = 1.3; b = 0.1	1328
a = 1.3; b = 0.2	1227
a = 1.4; b = 0.1	1385
a = 1.4; b = 0.2	1360
a = 1.4; b = 0.3	1329
a = 1.5; b = 0.1	1371
a = 1.5; b = 0.2	1357
a = 1.5; b = 0.3	1352
a = 1.5; b = 0.4	1250
a = 1.6; b = 0.1	1289
a = 1.6; b = 0.2	1310
a = 1.6; b = 0.3	1277
a = 1.6; b = 0.4	1317
a = 1.6; b = 0.5	1371
a = 1.7; b = 0.1	1284
a = 1.7; b = 0.2	1287
a = 1.7; b = 0.3	1307
a = 1.7; b = 0.4	1313
a = 1.7; b = 0.5	1322

Obr. 17 Zobrazení četnosti parametrů a , b adaptivní verze ChaosRnd Lozi map v rámci několika běhů DE

6 VÝSTUPY TESTOVÁNÍ

6.1 Testovací funkce

V rámci aplikace je implementováno třináct testovacích funkcí (Tab. 4), které byly využívány v průběhu odlaďování aplikace. Pro testování všech kombinací implementovaných možných kombinací nastavení DE a generátorů náhodných čísel byly však využity pouze čtyři, které korespondují se základním testováním při výzkumu použití Lozi map pro generování náhodných čísel [11].

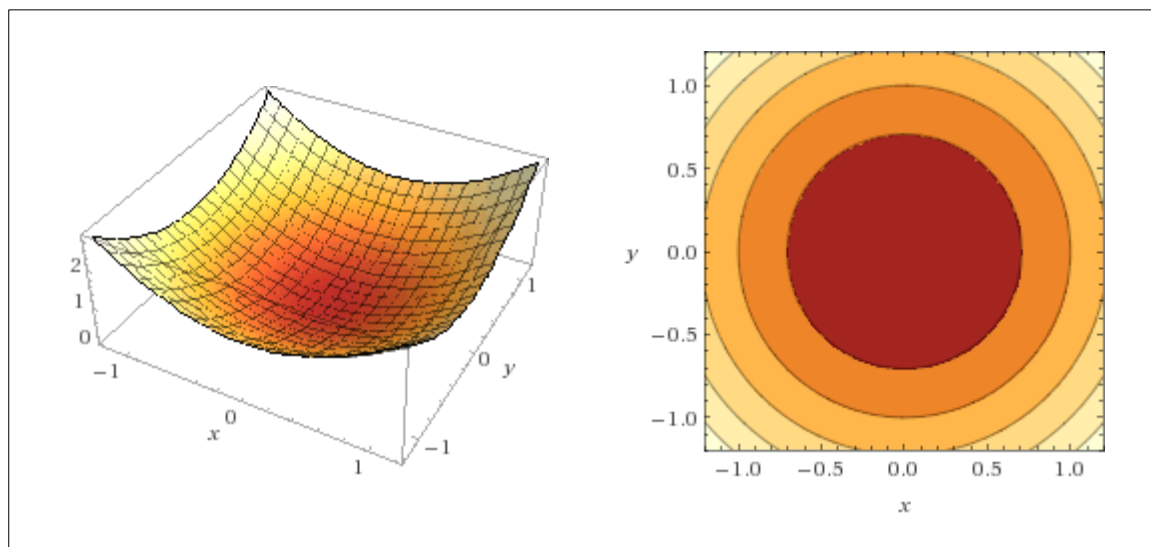
Tab. 4 Testovací funkce implementované v aplikaci

Název testovací funkce	Tvar funkce
První De Jongova	$\sum_{i=1}^D x_i^2 \quad (16)$
Druhá De Jongova	$\sum_{i=1}^{D-1} (x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \quad (17)$
Třetí De Jongova	$\sum_{i=1}^D x_i \quad (18)$
Čtvrtá De Jongova	$\sum_{i=1}^D i x_i^4 \quad (19)$
Rastriginova	$10D + \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i)) \quad (20)$
Schwefelova	$\sum_{i=1}^D -x_i \sin(\sqrt{ x_i }) \quad (21)$
Griewangkova	$1 + \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) \quad (22)$

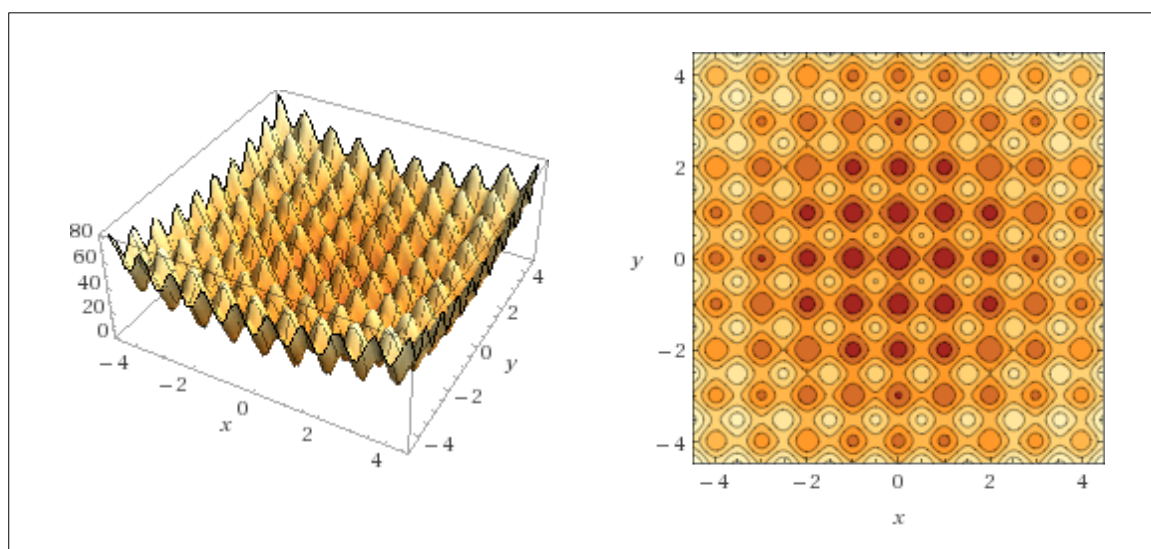
Sinová obálková sinusoidální	$-\sum_{i=1}^{D-1} 0,5 + \frac{\sin\left(\sqrt{x_i^2 + x_{i+1}^2} - 0,5\right)^2}{\left(1 + 0,001(x_i^2 + x_{i+1}^2)\right)^2} \quad (23)$
Roztažená sinusoidální V	$\sum_{i=1}^{D-1} \left(\sqrt[4]{x_i^2 + x_{i+1}^2} \sin\left(50 \sqrt[10]{x_i^2 + x_{i+1}^2}\right)^2 + 1 \right) \quad (24)$
Ackleyho I	$\sum_{i=1}^{D-1} \left(\frac{\sqrt{x_i^2 + x_{i+1}^2}}{e^{\frac{1}{5}}} + 3(\cos(2x_i) + \sin(2x_{i+1})) \right) \quad (25)$
Ackleyho II	$\sum_{i=1}^{D-1} \left(20 + e - e^{\frac{\cos(2\pi x_i) + \cos(2\pi x_{i+1})}{2}} - 20 \left(e^{-0,2 \cdot \sqrt{\frac{x_i^2 + x_{i+1}^2}{2}}} \right) \right) \quad (26)$
Michalewiczova	$-\sum_{i=1}^D \sin(x_i) \sin^{20}\left(\frac{ix_i^2}{\pi}\right) \quad (27)$
Mastersova	$-\sum_{i=1}^{D-1} \left(e^{\frac{-(x_i^2 + x_{i+1}^2 + 0,5x_i x_{i+1})}{8}} \cos\left(4\sqrt{x_i^2 + x_{i+1}^2 + 0,5x_i x_{i+1}}\right) \right) \quad (28)$

Pro statistické vyhodnocení a vyvození závěrů byly vybrány následující funkce:

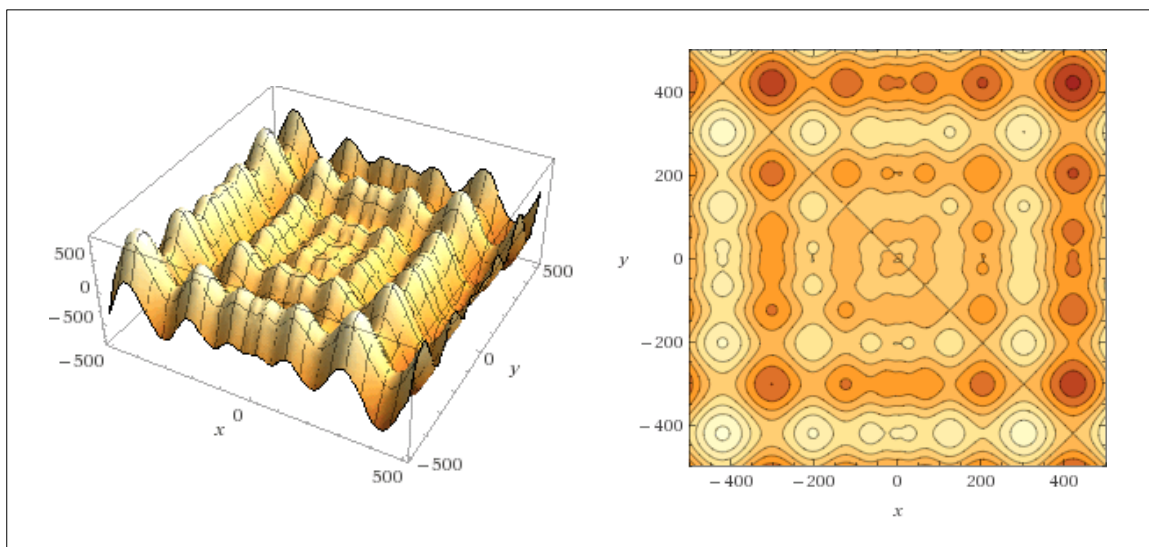
- První De Jongova funkce (16) (Obr. 18)
- Rastriginova funkce (20) (Obr. 19)
- Schwefelova funkce (21) (Obr. 20)
- Ackleyho funkce II (26) (Obr. 21)



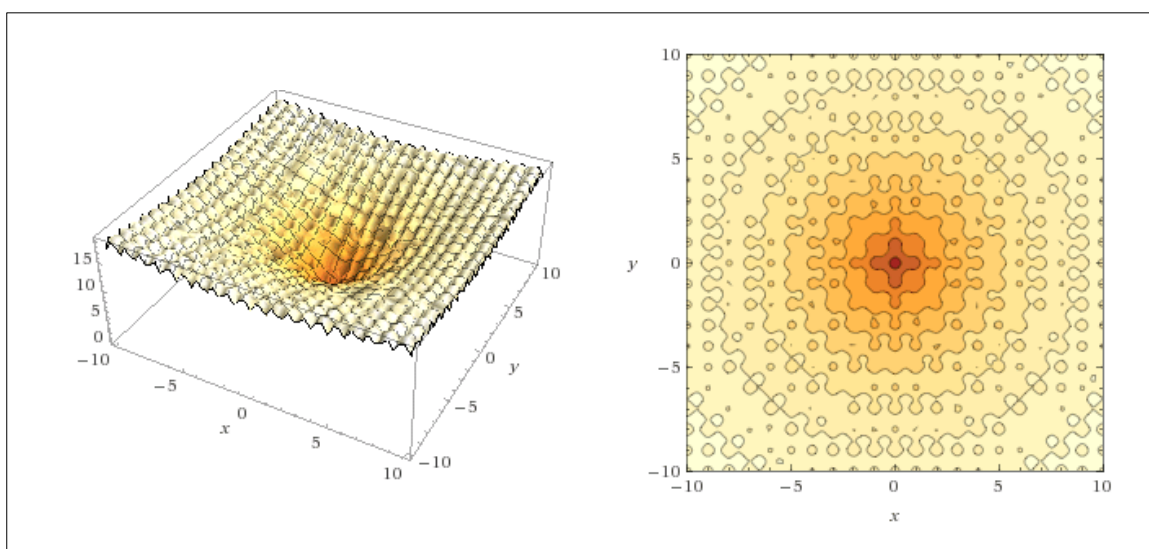
Obr. 18 První De Jongova funkce [12]



Obr. 19 Rastriginova funkce [12]



Obr. 20 Schwefelova funkce [12]

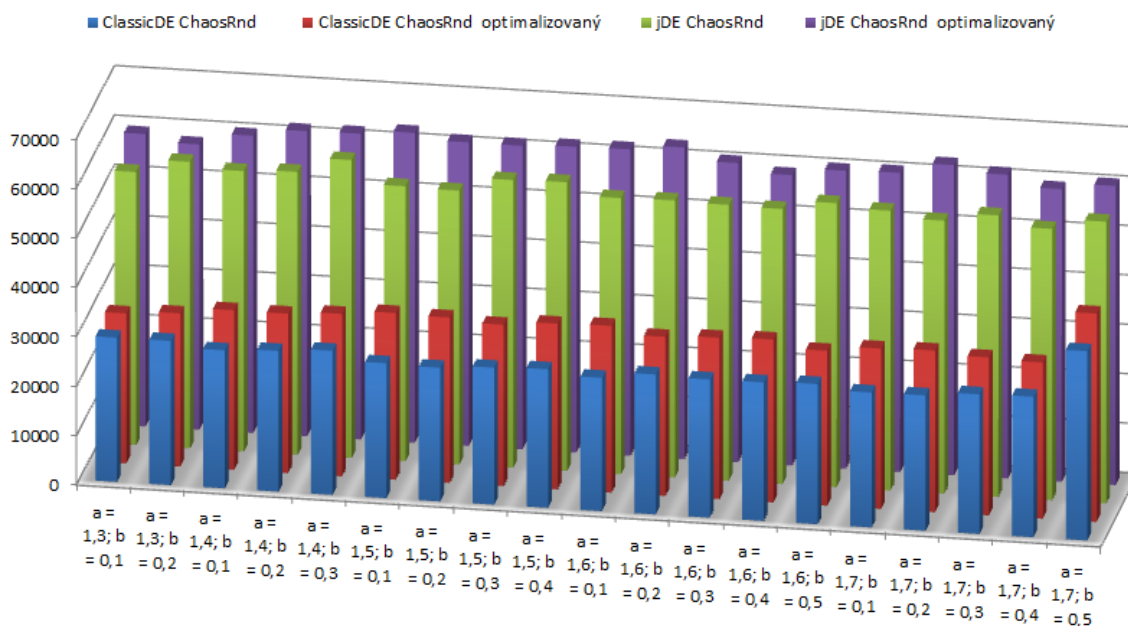


Obr. 21 Schwefelova funkce [12]

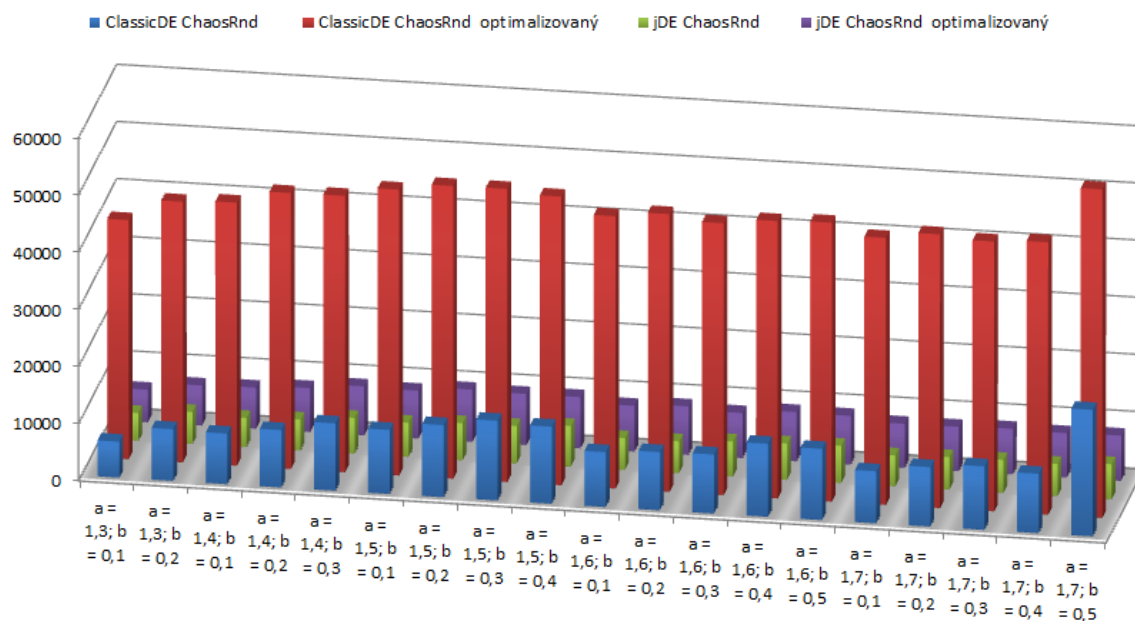
Na těchto čtyřech zobrazených funkcích bylo prováděno testování s cílem zjistit výkonnost různých nastavení adaptivnosti u řídicích parametrů DE a chaos generátoru náhodných čísel ve srovnání s výkonností klasické DE volitelně doplněné o adaptivní chaos generátor náhodných čísel.

6.2 Adaptivní hodnoty ChaosRnd

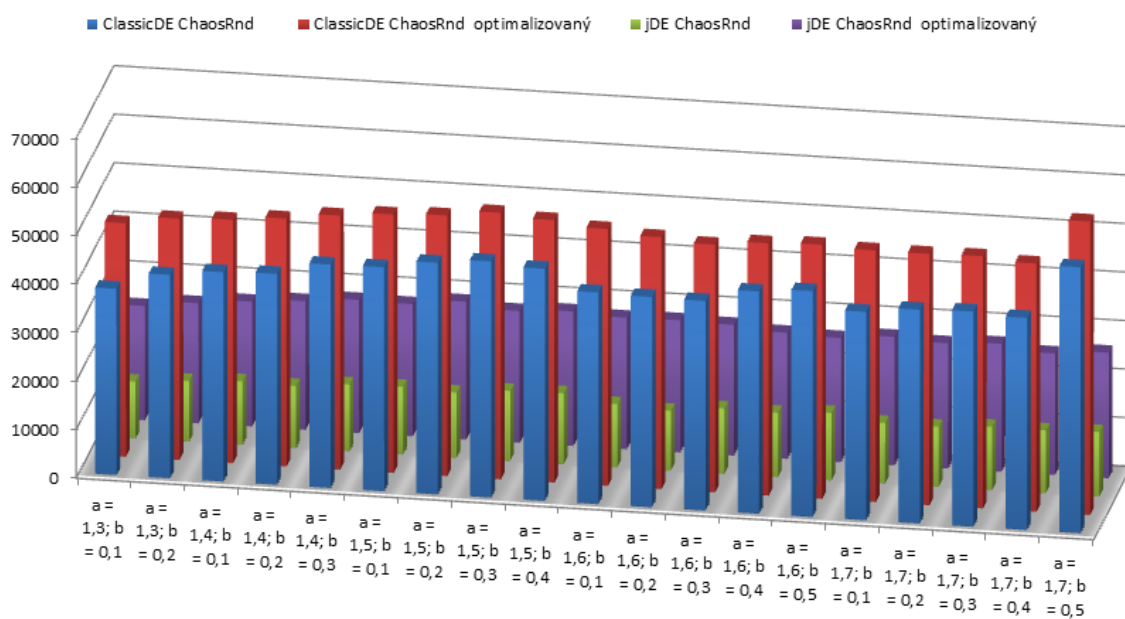
Vzhledem k faktu, že v rámci implementace třídy ChaosRnd jsou možné kombinace parametrů generovány jako výběr náhodného indexu z předdefinovaného jednorozměrného pole všech kombinací parametrů a index je náhodně generován pomocí systémového generátoru pseudonáhodných čísel třídy System.Random, lze očekávat rovnoměrné rozložení četnosti nově vybraných kombinací. Na následujících grafech, (Obr. 22) až (Obr. 25), je vidět rozložení četnosti kombinací a , b parametrů použitého Lozi map chaotického systému, které se podílely úspěšně na tvorbě nových jedinců, a přežili tedy do další generace. Vzhledem k tomu, že různá nastavení parametrů a , b mohou nastat jen v případě adaptivních verzí chaotického generátoru náhodných čísel, jsou pozorovány pouze tyto dvě varianty v kombinaci s klasickou DE a jDE. Adaptivní změny parametrů hodnot probíhají podle stejného algoritmu, jako u jDE.



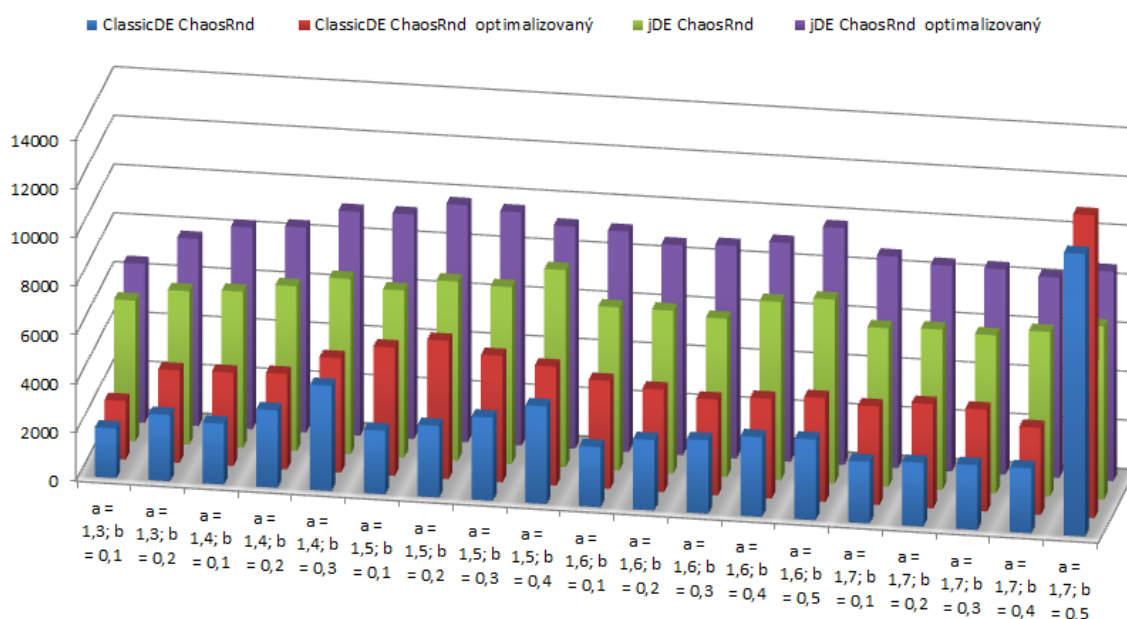
Obr. 22 Kombinovaný histogram četností úspěšných kombinací parametrů a , b adaptivních verzí ChaosRnd u První De Jongovi funkce



Obr. 23 Kombinovaný histogram četností úspěšných kombinací parametrů a , b adaptivních verzí ChaosRnd u Rastriginovi funkce



Obr. 24 Kombinovaný histogram četností úspěšných kombinací parametrů a , b adaptivních verzí ChaosRnd u Schwefelovy funkce



Obr. 25 Kombinovaný histogram četností úspěšných kombinací parametrů a , b adaptivních verzí ChaosRnd u Ackleyho funkce II

6.3 Adaptivní hodnoty parametrů jDE

Při generování nových náhodných hodnot řídicích parametrů F a CR jsou hodnoty generovány podle doporučení uvedeného v popisu algoritmu v rozmezí $F = [0.1, 0.9]$ a $CR = [0, 1]$. Bylo ověřeno, že rozložení hodnot generovaných pro všechny jedince (nejen úspěšné) je rovnoměrné v celé oblasti kombinací všech hodnot.

Například pro První De Jongovu funkci, při padesáti bžích jDE v nastavení $NP = 50$, $D = 30$ a $G_{\max} = 1500$, bylo vygenerováno průměrně 46843,75 nastavení F a CR v každé kombinaci hodnot parametrů vzorkované po 0,1 z povoleného rozsahu. Rozdíl mezi minimální četností a maximální četností kombinací nastavení byl v rozmezí od 2951 po 3935 se směrodatnou odchylkou v rozmezí od 652,8 po 724,2. Pro ostatní funkce platí podobné statistiky.

Oproti tomu statistiky úspěšných kombinací nastavení řídicích parametrů F a CR vykazují výrazně vyšší četnost generovaných hodnot v oblasti nízkých hodnot a to u všech čtyřech testovacích funkcí. Grafické znázornění tohoto jevu je na obrázku (Obr. 26) a výrazněji na obrázku (Obr. 27), kde v levé části je podmíněným formátováním zvýrazněno rozložení všech generovaných kombinací pro všechny použité konfigurace generátoru náhodných čísel a v pravé části je stejným způsobem zvýrazněno rozložení kombinací řídicích parametrů F a CR , které byly úspěšné při tvorbě nových jedinců.

ClassicRnd									
46557	47529	47586	46776	47136	46386	46477	46047	47404	46860
47545	46973	46148	47210	48200	46375	46888	46977	48354	47642
46282	47133	46237	46723	46094	47529	47379	46418	48745	46794
46466	45997	47969	47128	47415	47085	45985	46297	46842	46361
46901	46992	46079	46725	48001	46564	47230	47081	47812	47061
47001	46537	46638	46701	46974	46667	45524	45521	46006	47099
47418	46989	47060	48132	46809	46575	45031	46115	46143	46720
46682	46571	46750	47116	46527	47397	46058	46418	46967	46959
ChaosRnd									
46561	46348	46760	47968	46796	46802	45972	46684	47349	46887
45812	47114	47650	48307	47246	48036	47852	46355	46075	46421
46361	47428	46810	47120	47597	45387	46873	46611	46022	47387
47125	46527	46630	46002	46036	45402	45679	47238	45885	46727
46342	47020	46328	46583	48247	46523	45984	46764	48054	46555
47781	47797	46143	47338	47406	46806	46178	47557	47935	46601
46927	46744	47262	46776	45396	47517	46570	46881	46126	45940
48338	46762	47520	47514	46665	46742	46358	47084	47763	46831
ChaosRnd optimalizovaný									
46635	46541	48461	47374	47762	47267	47326	46951	46586	47680
46232	47167	47955	46470	45931	46452	46292	46100	47423	45628
46098	47227	45897	46345	46662	47065	47517	46725	47512	46611
46605	46840	46199	47363	46815	46719	46890	46176	47480	47549
47681	46476	47838	47177	46790	47969	45685	47352	47255	45578
46303	46663	46743	45926	47551	45499	46765	47417	46376	47401
47552	47272	46534	45262	46551	47133	46746	46571	46293	47723
46832	46638	47341	46287	46459	47669	46776	45968	47434	46822
ChaosRnd adaptivní									
46357	46989	48026	46524	46846	46013	47003	46816	46030	47671
46668	46923	46502	47226	47359	47398	45452	46195	46810	47005
46468	46739	47385	45849	48783	46767	47701	48420	46868	46944
45481	47776	47781	47183	47337	46700	47079	47120	46511	46773
46152	46044	46129	46405	46840	46257	46678	47418	46014	46660
46830	46512	46765	46796	46699	46712	47144	48784	45440	46503
47028	47137	46858	46730	46896	46614	46288	47258	47140	47407
45093	46555	46950	47812	47034	47773	45997	47054	47216	46400
ChaosRnd adaptivní optimalizovaný									
46749	46906	47106	47810	47083	47001	48792	46061	46565	45998
47125	46975	47742	47512	47399	47197	47637	46599	47836	46281
46596	46683	47851	45918	47490	46062	48057	46941	46365	47829
47015	46170	44857	45960	46385	47079	46950	46740	46225	45581
45427	47377	47311	47143	46443	46235	46045	46231	45222	46791
47992	46077	46759	47041	47663	47530	47503	46464	47005	47519
47511	46844	47218	46643	46153	46393	46155	46492	47074	47093
45853	46658	47125	47823	45920	46706	47011	47729	46261	47932
ClassicRnd									
24194	24636	24038	24516	23679	23723	23428	23286	22110	22541
24634	23819	23815	24272	23278	23522	23319	22596	22339	22598
23793	23569	23698	23459	23769	23035	23036	22559	21797	22148
24380	23911	23227	23653	23153	22651	22465	22415	21553	20903
23056	23413	23954	23326	22852	22983	22881	22025	21861	20984
22781	22527	22695	22777	21581	22623	21870	21182	20575	20159
21705	22318	22483	22196	22211	22101	21591	21017	20980	20428
21575	21973	22432	22012	20987	21485	20636	20656	20291	19823
ChaosRnd									
21970	19246	17461	15782	15438	14599	14518	13497	13058	13126
21388	17802	16517	14656	13844	13332	13108	12549	12347	12106
21950	16790	14983	13618	12482	12258	11496	11031	10934	10553
20106	15686	14083	12262	11276	10704	10072	9421	9198	9267
18927	14539	12691	10762	9862	9161	8670	8211	7834	7482
17586	13052	11191	9054	7933	7466	6979	6654	6470	6051
17073	11811	8953	7488	6793	6259	5499	5188	4987	4790
15849	9781	7618	6394	5204	4654	4406	4266	3732	3855
ChaosRnd optimalizovaný									
26274	24111	22279	20820	19151	17803	17090	16056	15736	14987
25814	23437	21439	19469	18184	17150	15789	15069	14663	14104
24472	22174	19557	17991	16357	14983	14287	13224	12843	12269
23755	21022	18743	16316	14733	13456	12338	11390	11004	10393
23896	19919	16663	15092	12551	11257	10360	9498	8756	8328
21800	17692	14919	12350	10641	9496	8322	7442	6938	6365
20613	16649	13023	10499	9027	7444	6586	5626	5421	4637
19383	14271	10952	8810	7236	5732	5059	4326	3905	3432
ChaosRnd adaptivní									
25788	22585	20562	18976	18569	16666	16594	15729	15402	15750
25393	22107	18979	18275	17240	15599	15138	15043	14656	14202
23714	20730	18653	16265	15261	14227	13675	13356	12830	12753
23208	19214	16306	14389	13797	12065	11725	11247	11270	10462
22256	17873	15131	12601	11802	10364	9936	9444	9240	9071
20523	16210	12877	10773	9795	8949	8335	7891	7250	7410
19637	14046	11122	8987	8025	6996	6464	6139	5721	5711
18075	12075	9308	7622	6239	5691	5041	4794	4347	4468
ChaosRnd adaptivní optimalizovaný									
27255	24721	22487	20656	19466	17793	17220	16120	15987	15130
25908	23814	22074	20231	18845	16964	16289	15370	14668	14359
24903	22504	20043	18459	17260	15633	14346	13658	13155	12293
24286	20722	18517	16892	15363	13662	12708	11752	11133	10420
23258	20211	16813	14981	13272	12019	10825	9732	8937	8290
21502	18019	14991	12754	11242	9860	8361	7928	6984	6384
20561	16311	13458	11062	9310	7802	6890	5971	5423	4647
19346	14555	11774	9299	7665	6317	5418	4494	3965	3557

Obr. 26 Vizualizace rozložení četnosti výskytu kombinací nastavení F a CR při jDE na První De Jongově funkci

ClassicRnd									
11886	6486	4119	2786	2041	1522	1227	989	709	537
8412	3608	1946	1291	858	674	491	432	392	331
7426	3158	1547	1074	737	523	437	396	331	289
7553	2923	1624	865	713	486	394	293	264	195
7361	2703	1338	767	574	353	210	239	194	127
6650	2406	1077	576	441	289	188	171	125	86
6559	2346	1122	571	384	199	170	108	100	69
7644	3067	1256	637	282	211	114	69	65	42
ChaosRnd									
11621	5823	3500	2241	1644	1306	1036	895	757	625
8292	3019	1482	961	720	700	395	355	334	263
7311	2382	1186	804	591	422	396	258	268	240
7192	2281	1100	627	410	352	304	261	178	194
6904	2001	922	472	342	246	183	199	159	124
6069	1819	698	391	268	176	134	118	131	91
6424	1581	628	304	197	124	101	90	57	76
7190	1936	636	235	143	99	80	61	45	39
ChaosRnd optimalizovaný									
12830	7365	4628	3101	2085	1569	1135	814	642	565
9140	4332	2219	1308	968	616	492	487	303	344
8672	3808	1868	1108	788	550	399	311	335	226
8851	3864	1847	1018	656	431	328	258	191	210
8111	3599	1716	850	491	364	289	186	137	160
7361	2955	1362	763	470	280	237	165	115	103
7583	3164	1206	660	367	242	122	103	73	38
8714	3864	1678	801	378	236	139	65	45	28
ChaosRnd adaptivní									
11439	5665	3426	2425	1964	1532	1011	880	766	692
7780	2649	1468	1053	688	606	489	309	248	302
6817	2136	1282	803	515	386	428	379	219	199
7004	2271	1129	676	385	312	247	295	242	144
6590	2013	914	511	348	238	214	214	155	130
6112	1666	673	387	237	168	167	117	136	95
6501	1568	608	313	206	150	107	84	45	69
6966	1855	709	311	205	133	85	64	48	46
ChaosRnd adaptivní optimalizovaný									
12841	7331	4848	3308	2216	1559	1186	989	728	676
8903	4367	2410	1472	964	671	543	415	432	342
8215	3835	1880	1065	753	488	487	386	339	250
8580	3685	1908	1152	684	504	393	289	229	209
8070	3566	1704	1008	632	402	332	220	160	127
7156	3066	1491	781	520	310	250	154	120	101
7397	3187	1529	792	488	260	192	110	90	55
8819	3845	1770	928	501	276	137	91	69	53

Obr. 27 Vizualizace rozložení četnosti výskytu úspěšných kombinací nastavení F a CR při jDE na Rastriginově funkci

6.4 Efektivita variant DE

Pro testování výkonnosti funkcí bylo nastavení základních parametrů všech variant DE nastaveno následujícím způsobem:

- počet generací $G_{\max} = 1500$
- počet jedinců populace $NP = 50$
- strategie u klasické verze **DE/DE/rand/1/bin**
- nastavení F a CR u klasické verze DE vycházelo z nejlepších výsledků předešlé studie [11] pro použité funkce a hodnoty byly nastaveny na:
 - $F = 0.4$ a $CR = 0.4$ pro **První De Jongovu** funkci
 - $F = 0.5$ a $CR = 0.1$ pro **Rastriginovu** funkci a **Schweffelovu** funkci
 - $F = 0.4$ a $CR = 0.3$ pro **Ackleyho** funkci II

- dimenze u všech testů **D = 30**
- **počet opakování** DE s každou konfigurací pro statistické vyhodnocení **50**

U každé funkce byly provedeny testy pro každou kombinaci klasické DE a pěti typů generátoru náhodných čísel i každou kombinaci jDE a stejných pěti typů generátoru náhodných čísel. Všechny kombinace jsou uvedeny v následující tabulce (Tab. 5).

Tab. 5 Kombinace verzí DE a implementovaných generátorů náhodných čísel

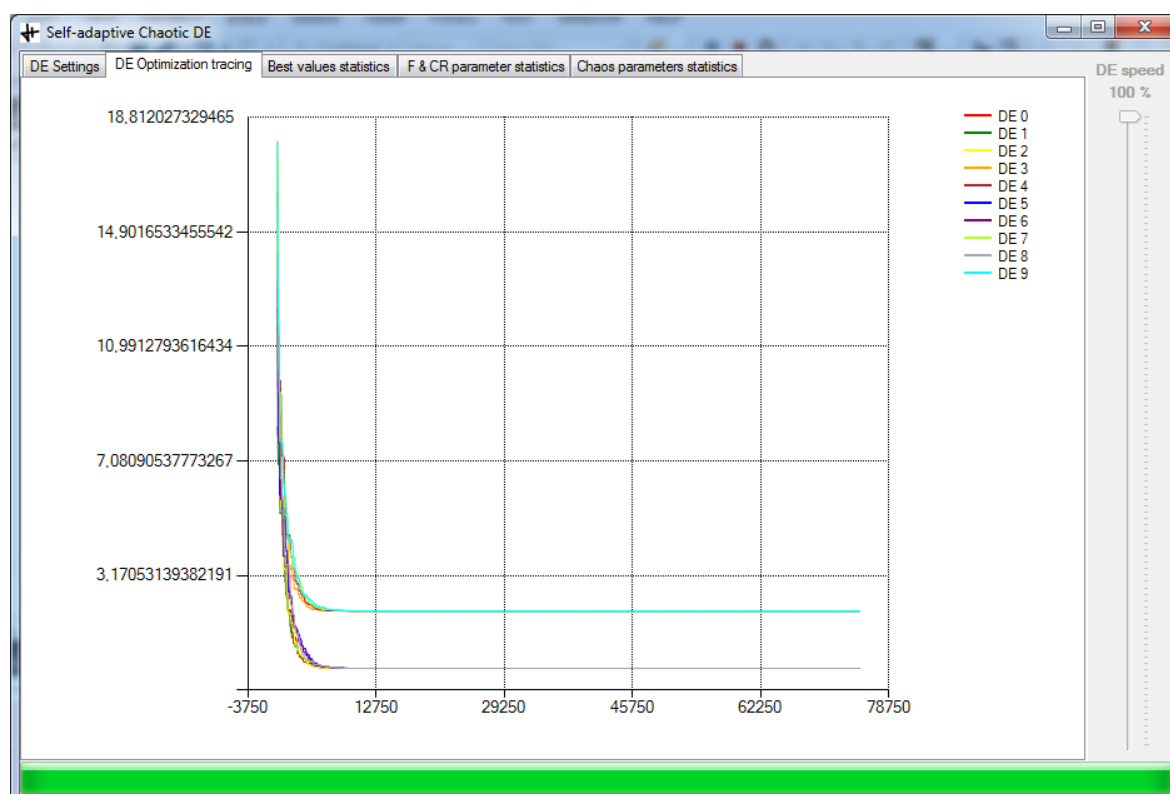
verze DE	generátor náhodných čísel	označení v grafech
Classic DE	ClassicRnd	DE 0
Classic DE	ChaosRnd	DE 1
Classic DE	ChaosRnd optimalizovaný	DE 2
Classic DE	ChaosRnd adaptivní	DE 3
Classic DE	ChaosRnd adaptivní optimalizovaný	DE 4
jDE	ClassicRnd	DE 5
jDE	ChaosRnd	DE 6
jDE	ChaosRnd optimalizovaný	DE 7
jDE	ChaosRnd adaptivní	DE 8
jDE	ChaosRnd adaptivní optimalizovaný	DE 9

Vzhledem k úmyslu vyhodnotit vliv adaptivních verzí generátoru náhodných čísel byly výsledky pro klasickou DE a jDE vyhodnocovány samostatně, aby vynikl vliv použité adaptivní verze na výkon DE.

6.4.1 První De Jongova funkce

Vzhledem k tomu, že První De Jongova funkce je unimodální a jedná se z pohledu optimalizace o poměrně jednoduchý problém, je vcelku překvapivé, že docházelo u použití adaptivních verzí ChaosRnd spolu s klasickou DE ke stoprocentní stagnaci optimalizace na hodnotě 1,96, jak je vidět z grafu (Obr. 28) i z tabulky výsledných hodnot (Tab. 6). U ostatních variant DE probíhala optimalizace podle očekávání velice rychle (cca 10000 ohodnocení účelové funkce). I když jsou rozdíly u dalších variant DE malé, tak jak je z tabulky vidět nejlepších hodnot dosahuje, také díky optimálnímu nastavení řídicích parametrů, klasická DE ve spojení se základní verzí ChaosRnd a to ve všech statistických hodnotách.

Vzhledem k malým rozdílům u výsledků v rámci variant jDE nelze jednoznačně posoudit vliv použitých generátorů čísel na výkon jDE u této funkce, z tabulky (Tab. 7) je však možné vidět (nejlepší hodnoty jsou tučně), že ačkoliv nedosáhla adaptivní varianta ChaosRnd nejlepšího výsledku, tak však má nejstabilnější průběh, s druhou nejlepší hodnotou výsledku.



Obr. 28 Porovnání průběhu optimalizace variant DE pro První De Jongovu funkci

Tab. 6 Srovnání generátorů náhodných čísel s klasickou DE na První De Jongově funkci po 1500 generacích

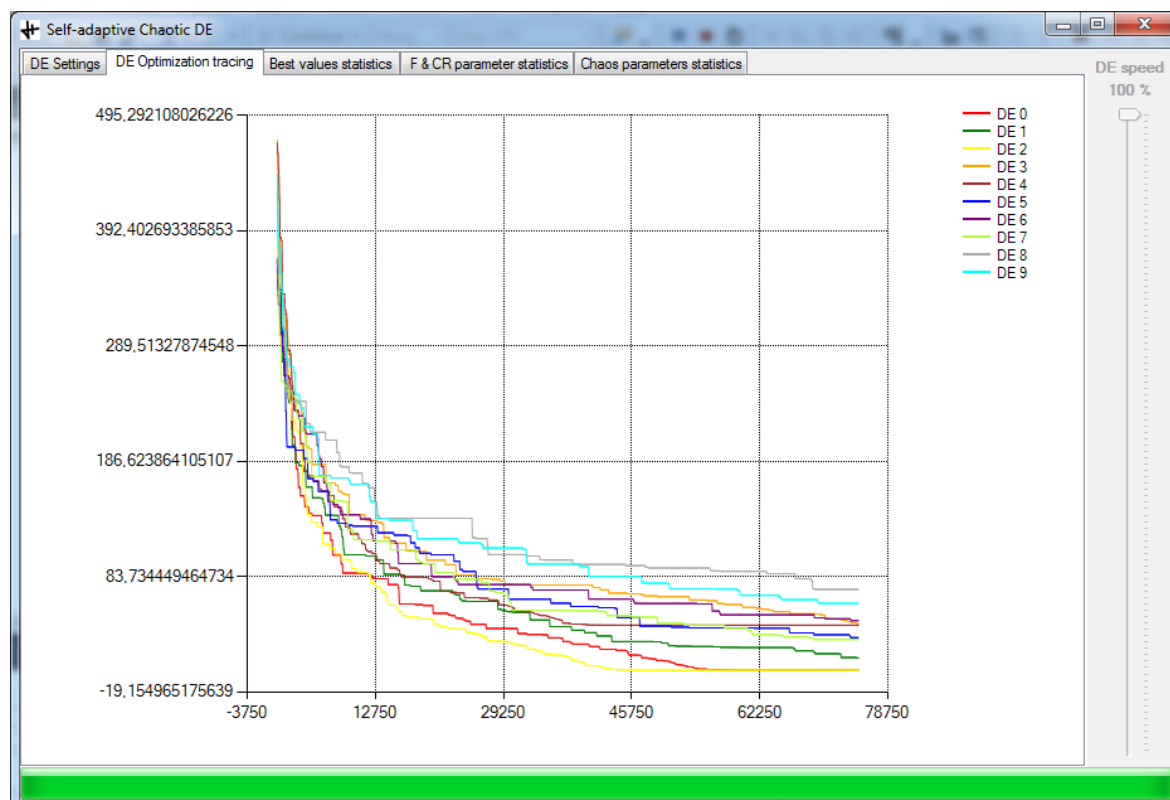
ClassicRnd	Průměr	Nejlepší	Medián	Sm. Odch.	Rozptyl
ClassicRnd	1,25E-31	1,81E-32	1,00E-31	9,77E-32	9,54E-63
ChaosRnd	6,51E-32	9,16E-33	4,50E-32	9,37E-32	8,78E-63
ChaosRnd opt.	1,73E-31	2,33E-32	1,25E-31	1,50E-31	2,24E-62
ChaosRnd adapt.	1,96E+00	1,96E+00	1,96E+00	1,78E-15	3,16E-30
ChaosRnd adapt. opt.	1,96E+00	1,96E+00	1,96E+00	1,78E-15	3,16E-30

Tab. 7 Srovnání generátorů náhodných čísel s jDE na První De Jongově funkci po 1500 generacích

jDE	Průměr	Nejlepší	Medián	Sm. Odch.	Rozptyl
ClassicRnd	1,26E-23	7,39E-25	7,79E-24	1,65E-23	2,73E-46
ChaosRnd	6,92E-25	9,08E-27	3,58E-25	9,76E-25	9,53E-49
ChaosRnd opt.	5,66E-23	5,85E-24	4,22E-23	6,41E-23	4,11E-45
ChaosRnd adapt.	5,24E-25	2,58E-26	3,00E-25	5,84E-25	3,41E-49
ChaosRnd adapt. opt.	6,39E-23	2,98E-24	3,41E-23	7,89E-23	6,22E-45

6.4.2 Rastriginova funkce

Při optimalizaci Rastriginovi funkce dosahuje zdaleka nejlepších výsledků klasická DE, a to v kombinaci s optimalizovaným ChaosRnd generátorem a srovnatelných výsledků s pseudonáhodným generátorem čísel. Všechny ostatní DE mají výsledky optimalizace horší a všechny čtyři verze s adaptivními verzemi ChaosRnd vykazují výsledky optimalizace nejméně úspěšné. Výrazně lepší výsledek klasické DE je pravděpodobně způsoben optimálně nastavenými řídicími parametry F a CR. V případě klasické DE je varianta s optimalizovaným ChaosRnd generátorem nejlepší ve všech statistických hodnotách (Tab. 8) a také dosahuje téměř optimálního řešení nejdříve (cca 40000 CFE), jak je vidět z obrázku (Obr. 29). U všech variant jDE není rozdíl mezi výsledky tak velký, ale jak je vidět v tabulce (Tab. 9), je opět nejúspěšnější varianta s optimalizovaným ChaosRnd generátorem ve všech porovnávaných hodnotách.



Obr. 29 Porovnání průběhu optimalizace variant DE pro Rastriginovu funkci

Tab. 8 Srovnání generátorů náhodných čísel s klasickou DE na Rastriginově funkci po 1500 generacích

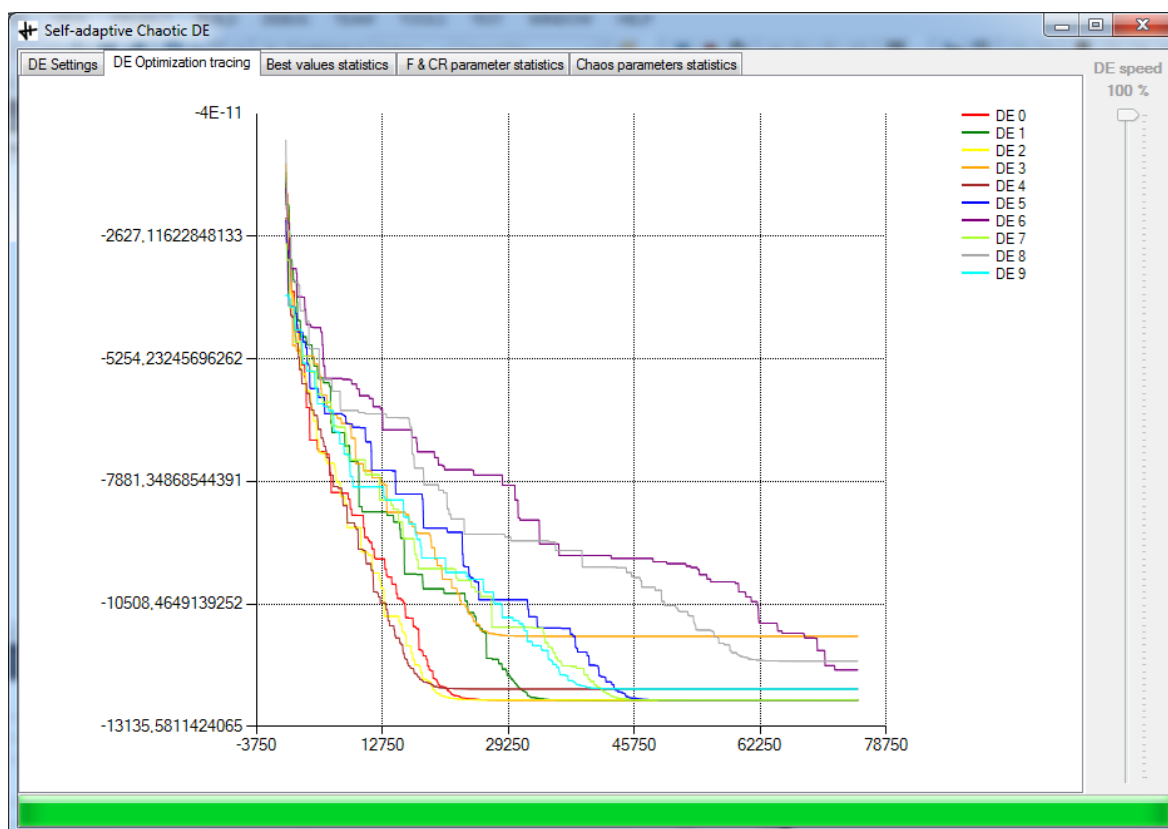
ClassicRnd	Průměr	Best	Medián	Sm. Odch.	Rozptyl
ClassicRnd	2,02E-06	2,92E-08	6,10E-07	3,94E-06	1,56E-11
ChaosRnd	1,16E+01	4,35E-02	1,46E+01	7,40E+00	5,47E+01
ChaosRnd opt.	6,58E-10	2,12E-11	4,28E-10	7,63E-10	5,82E-19
ChaosRnd adapt.	4,68E+01	4,08E+01	4,35E+01	8,78E+00	7,71E+01
ChaosRnd adapt. opt.	4,21E+01	4,03E+01	4,12E+01	1,68E+00	2,82E+00

Tab. 9 Srovnání generátorů náhodných čísel s jDE na Rastriginově funkci po 1500 generacích

jDE	Průměr	Best	Medián	Sm. Odch.	Rozptyl
ClassicRnd	3,00E+01	1,51E+01	3,07E+01	4,72E+00	2,22E+01
ChaosRnd	3,57E+01	1,90E+01	3,50E+01	6,56E+00	4,31E+01
ChaosRnd opt.	2,48E+01	1,06E+01	2,30E+01	4,51E+00	2,03E+01
ChaosRnd adapt.	7,31E+01	5,00E+01	7,39E+01	6,99E+00	4,89E+01
ChaosRnd adapt. opt.	5,94E+01	4,85E+01	5,89E+01	4,71E+00	2,22E+01

6.4.3 Schwefelova funkce

V případě optimalizace Schwefelovi funkce, opět díky nastavení optimálních parametrů F a CR , dosahuje nejlepších výsledků klasická DE. Adaptivní jDE však dosahuje optimálních výsledků také. Jak je vidět v tabulce pro klasickou DE (Tab. 10) i v tabulce pro jDE (Tab. 11), dosahují obě verze DE optimálních výsledků ve spojení s pseudonáhodným generátorem čísel a optimalizovaným ChaosRnd generátorem. V případě optimalizovaného ChaosRnd je dokonce dosaženo optimálního výsledku ve všech bžích DE. Jak je vidět z grafu (Obr. 30), klasická DE s optimalizovaným ChaosRnd dosahuje optima i nejdříve. Z grafu je také vidět, že u adaptivních verzí ChaosRnd dochází k zastavení optimalizace vlivem stagnace nebo nalezení suboptima. Z grafu je také vidět, že adaptivní optimalizovaná verze ChaosRnd s klasickou DE nejrychleji konverguje k řešení, náhle však je optimalizace zastavena.



Obr. 30 Porovnání průběhu optimalizace variant DE pro Schwefelovu funkci

Tab. 10 Srovnání generátorů náhodných čísel s klasickou DE na Schwefelově funkci po 1500 generacích

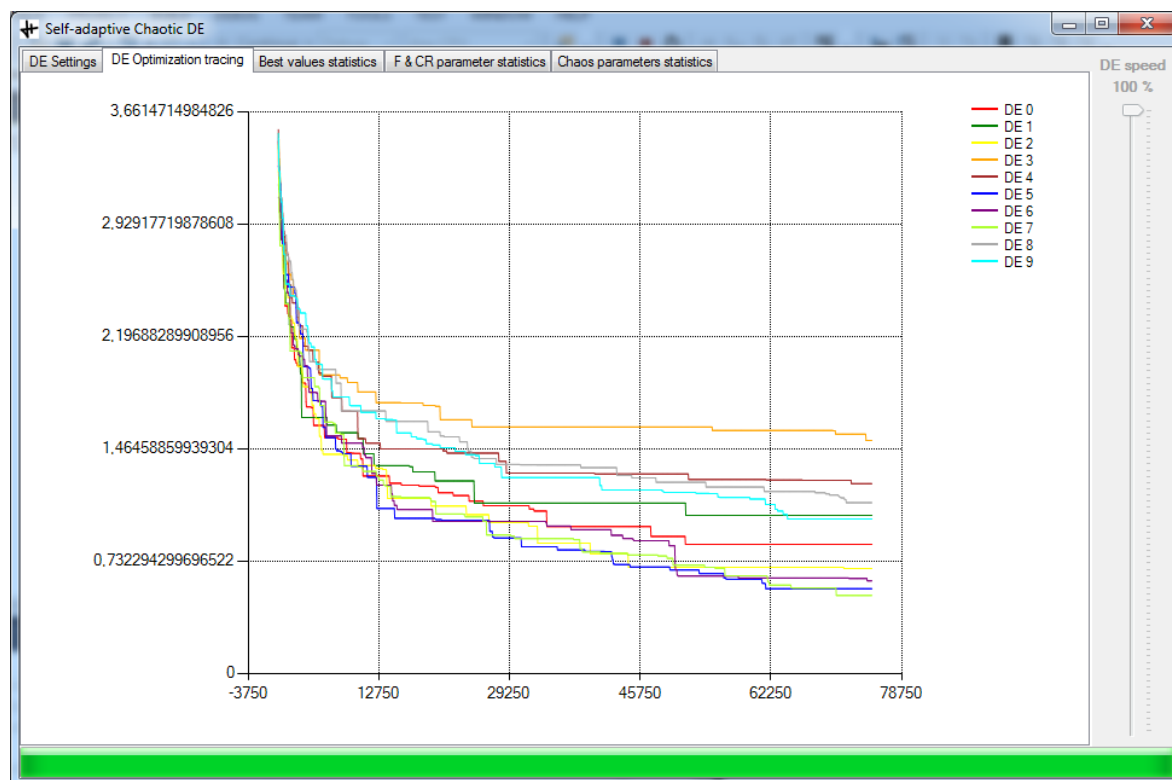
ClassicRnd	Průměr	Best	Medián	Sm. Odch.	Rozptyl
ClassicRnd	-1,26E+04	-1,26E+04	-1,26E+04	2,86E+01	8,15E+02
ChaosRnd	-1,24E+04	-1,26E+04	-1,25E+04	1,16E+02	1,34E+04
ChaosRnd opt.	-1,26E+04	-1,26E+04	-1,26E+04	0,00E+00	0,00E+00
ChaosRnd adapt.	-1,15E+04	-1,19E+04	-1,15E+04	2,03E+02	4,11E+04
ChaosRnd adapt. opt.	-1,23E+04	-1,23E+04	-1,23E+04	8,27E+01	6,84E+03

Tab. 11 Srovnání generátorů náhodných čísel s jDE na Schwefelově funkci po 1500 generacích

jDE	Průměr	Best	Medián	Sm. Odch.	Rozptyl
ClassicRnd	-1,26E+04	-1,26E+04	-1,26E+04	1,36E-07	1,84E-14
ChaosRnd	-1,18E+04	-1,26E+04	-1,18E+04	6,57E+02	4,32E+05
ChaosRnd opt.	-1,26E+04	-1,26E+04	-1,26E+04	2,71E-08	7,35E-16
ChaosRnd adapt.	-1,16E+04	-1,20E+04	-1,16E+04	2,83E+02	8,00E+04
ChaosRnd adapt. opt.	-1,23E+04	-1,23E+04	-1,23E+04	2,20E-10	4,86E-20

6.4.4 Ackleyho funkce II

U optimalizace Ackleyho funkce II jsou výsledky podobné, jako u předchozích funkcí. Nejlepších výsledků dosahují u klasické DE i jDE varianty s pseudonáhodným generátorem náhodných čísel a s optimalizovaným ChaosRnd generátorem. Překvapivě, vzhledem k výsledkům uvedených v předcházející studii [11], nenašla ani jedna implementovaná varianta optimální řešení. Z grafu (Obr. 31), z tabulky pro klasickou DE (Tab. 12), i z tabulky vyhodnocení pro jDE (Tab. 13), je však zřejmé, že adaptivní verze ChaosRnd, v rámci obou verzí DE, dosahují statisticky viditelně horších výsledků.



Obr. 31 Porovnání průběhu optimalizace variant DE pro Ackleyho funkci II

Tab. 12 Srovnání generátorů náhodných čísel s klasickou DE na Ackleyho funkci II po 1500 generacích

ClassicRnd	Průměr	Best	Medián	Sm. Odch.	Rozptyl
ClassicRnd	7,91E-01	6,09E-01	8,00E-01	6,32E-02	4,00E-03
ChaosRnd	1,01E+00	8,11E-01	1,01E+00	7,39E-02	5,46E-03
ChaosRnd opt.	7,45E-01	6,30E-01	7,38E-01	5,19E-02	2,70E-03
ChaosRnd adapt.	1,50E+00	1,33E+00	1,50E+00	5,31E-02	2,82E-03
ChaosRnd adapt. opt.	1,23E+00	1,14E+00	1,23E+00	4,67E-02	2,18E-03

Tab. 13 Srovnání generátorů náhodných čísel s jDE na Ackleyho funkci II po 1500 generacích

jDE	Průměr	Best	Medián	Sm. Odch.	Rozptyl
ClassicRnd	5,06E-01	4,25E-01	4,94E-01	5,59E-02	3,13E-03
ChaosRnd	5,86E-01	4,16E-01	5,96E-01	7,47E-02	5,58E-03
ChaosRnd opt.	4,64E-01	2,92E-01	4,53E-01	5,87E-02	3,45E-03
ChaosRnd adapt.	1,15E+00	1,02E+00	1,15E+00	4,91E-02	2,41E-03
ChaosRnd adapt. opt.	1,02E+00	9,28E-01	1,02E+00	4,56E-02	2,08E-03

7 SHRUTÍ VÝSLEDKŮ TESTOVÁNÍ

U statistického pozorování hodnot řídicích parametrů ChaosRnd je vidět, že četnost podílení se na tvorbě nových úspěšně mutovaných jedinců je rovnoměrně téměř rozložena na všechny kombinace parametrů (čtenější výskyt $a=1.7$ a $b=0.5$ je dán výchozím nastavením první generace). Ačkoliv je z předchozích výzkumů i z této práce zřejmé, že použití chaosu pro generování náhodných čísel, přináší vylepšení výkonu různých verzí DE, tak adaptivní přenos generátoru a jeho nastavení řídicích parametrů s jedincem, jak je to implementováno v této práci, zlepšení optimalizace nepřináší spíše naopak.

Při statistickém pozorování řídicích parametrů jDE, bylo zjištěno, že u všech testovaných funkcí dochází k úspěšné tvorbě nových mutovaných jedinců v populaci statisticky shodným způsobem více pomocí parametrů F a CR s nižšími hodnotami. Tato shoda může být způsobena tím, že všechny použité funkce jsou separabilní [3].

Ačkoliv je při testování velice úspěšná klasická DE, je třeba brát v potaz, že byly pro testování použity dříve zjištěné dobré řídicí parametry F a CR pro jednotlivé optimalizace. V případě reálných využití DE na optimalizačních problémech, které nemají známé řešení a nejsou předem známy optimální hodnoty nastavení F a CR, budou jistě adaptivní verze DE, výkonnějším řešením. Testování ukázalo, že využití chaotického systému v jDE může také vylepšit výkon tohoto algoritmu.

ZÁVĚR

Cílem práce bylo naprogramovat adaptivní verzi DE s adaptivní verzí chaotického systému pro generování náhodných čísel a porovnat tuto verzi se základními verzemi DE. Pro splnění toho cíle byla vytvořena aplikace, umožňující uživatelsky měnit kombinace adaptivní jDE a klasické DE s různými verzemi generátorů náhodných čísel.

V průběhu implementace se ukázalo, že existuje mnoho různých variant, jakým způsobem vytvořit adaptivní generátor náhodných čísel postavený na chaotickém systému. Pro implementaci byl zvolen Lozi map systém z důvodu známých kombinací parametrů generujících chaotické chování systému, pro mutaci parametrů systému byla zvolena shodná strategie s jDE a v rámci objektového řešení aplikace byl celý adaptivní generátor, jako objekt, přidružen k jedincům v individuálních instancích.

V průběhu testování bylo zjištěno, že využití této implementované adaptivní verze, ve spojení s klasickou DE i adaptivní jDE, nepřináší zlepšení výkonnosti, ale spíše naopak. Samotný fakt, že tato konkrétní implementace nepřináší zlepšení řešení optimalizačních úloh, neznamená, že není možné vytvořit implementaci adaptivní verze chaotického generátoru náhodných čísel, který zlepšení přinese.

Do práce nebylo možné zahrnout všechny potenciálně možné varianty implementací chaotického systému, protože samotných chaotických systémů je větší množství a v kombinacích s různými programátorskými přístupy při implementaci vzniká velké množství způsobů, jak danou problematiku řešit. Cílem práce bylo vytvořit jednu z možných variant řešení daného tématu otestovat efektivitu takového řešení. Do budoucna se však tímto směrem otvírá velký prostor k dalšímu výzkumu.

CONCLUSION

The aim of the Master's Thesis was to develop an adaptive version of DE with adaptive version of chaotic systems for generating random numbers and compare this version with basic versions of DE. To fulfill that goal the application allowing a user to change the combination of adaptive terms and classical DE with different versions of random number generators was created.

During implementation, it turned out that there are many different variants how possibly create an adaptive random number generator based on the chaotic system. To implement was chosen Lozi map system because of known combinations of parameters generating chaotic system behavior. It was chosen the same strategy for system parameters mutation like in jDE. In the context of object-oriented application solution, the entire adaptive generator as an object associated with individuals in individual instances was implemented.

It was found, during testing phase, that the use of an implemented adaptive version used in connection with classical DE or adaptive jDE, does not improve performance, but it is rather the opposite. The fact that this particular implementation does not improve process of solving optimization problems does not mean that it cannot be created a chaotic implementation of an adaptive version of the random number generator, which will bring improvement.

Into this Master's Thesis it was not possible to include all the possible variants of potentially chaotic system implementations, because there are many chaotic systems existing and in combination with various programmers' approaches in the implementation a large amount of ways to solve the issue is possible. The aim was to create one of the possible options the topic and test the effectiveness of such a solution. In the future, however, this way opens up a large area for further research.

SEZNAM POUŽITÉ LITERATURY

- [1] MAYR, Ernst. *What evolution is*. London: Weidenfeld & Nicolson, 2002, 270 s. ISBN 0-7538-1368-8.
- [2] ZELINKA, Ivan. *Evoluční Výpočetní Techniky: Principy a Aplikace*. 1. vyd. Praha: BEN - technická literatura, 2009. ISBN 978-80-7300-218-3
- [3] ZELINKA, Ivan, et al. *Handbook of Optimization: From Classical to Modern Approach*. 2012. vyd. Berlín: Springer-Verlag, 2012. ISBN 978-3-642-30503-0.
- [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] ZELINKA, Ivan. *Umělá inteligence v problémech globální optimalizace*. 1. vyd. Praha: BEN - technická literatura, 2002. ISBN 80-7300-069-5.
- [6] LAMPINEN, Jouni, ZELINKA, Ivan. *Mechanical engineering design optimization by differential evolution*. In: *New ideas in optimization*,. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999, s.127-146. ISBN:0-07-709506-5
- [7] BREST, J., GREINER, S., BOSKOVIC, B., MERNIK, M., ZUMER, V., *Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems*. In: *Evolutionary Computation, IEEE Transactions* [online] , vol.10, no.6, s. 646-657, Dec. 2006 Dostupné z: doi: 10.1109/TEVC.2006.872133
- [8] MALLIPEDDI, R., P. N. SUGANTHAN, Q. K. PAN AND M. F. TASGETIREN *Differential evolution algorithm with ensemble of parameters and mutation strategies*. *Applied Soft Computing*, 2011, 11(2), 1679-1696.
- [9] MIHULA, Lukáš. *Generátory pseudonáhodných čísel*. Ostrava, 2014. Bakalářská práce. VŠB v Ostravě, Fakulta elektrotechniky a informatiky. Vedoucí práce Mgr. Bohumil Krajc, Ph.D.
- [10] 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

- [11] ŠENKEŘÍK, Roman. *On the Parameter Settings for the Chaotic Dynamics Embedded Differential Evolution*. In: *2015 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION*, 2015
- [12] KAZÍKOVÁ, Anežka. *Algoritmus diferenciální evoluce s prvky deterministického chaosu (CHaosDE) v prostředí C/C++*. Zlín, 2013. Bakalářská práce. UTB ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Ing. Roman Šenkeřík, Ph.D.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EVT	Evoluční výpočetní techniky.
DE	Diferenciální evoluce.
CR	Práh křížení.
F	Mutační konstanta.
NP	Velikost populace.
G_{\max}	Počet generací diferenciální evoluce.
jDE	Adaptivní varianta diferenciální evoluce.
CFE	Počet ohodnocení účelové funkce.

SEZNAM OBRÁZKŮ

<i>Obr. 1 Obecný cyklus evolučního algoritmu</i>	14
<i>Obr. 2 Binomické křížení [5]</i>	18
<i>Obr. 3 Exponenciální křížení [5]</i>	19
<i>Obr. 4 Základní GUI aplikace</i>	30
<i>Obr. 5 Diagram tříd aplikace bez GUI.....</i>	31
<i>Obr. 6 Schéma rozhraní a implementačních tříd.....</i>	33
<i>Obr. 7 Vhodné nastavení parametrů pro Lozi map [11]</i>	34
<i>Obr. 8 Histogram hodnot systému Lozi map přepočtených do $[0, 1]$.....</i>	35
<i>Obr. 9 Histogram hodnot systému Lozi map přepočtených do $[0, 1]$ po optimalizaci</i>	36
<i>Obr. 10 Zobrazení detailů tříd generátorů náhodných čísel</i>	37
<i>Obr. 11 Detaily tříd implementujících jedince DE v části diagramu tříd</i>	38
<i>Obr. 12 Detaily tříd implementujících varianty DE v části diagramu tříd.....</i>	40
<i>Obr. 13 Nastavení DE v rámci GUI aplikace.</i>	41
<i>Obr. 14 Vizualizace průběhu DE v grafu</i>	42
<i>Obr. 15 Tabulkové zobrazení výsledků optimalizací a jejich vyhodnocení</i>	42
<i>Obr. 16 Příklad zobrazení četnosti parametrů F a CR v rámci několika běhů DE.....</i>	43
<i>Obr. 17 Zobrazení četnosti parametrů a, b adaptivní verze ChaosRnd Lozi map rámcí několika běhů DE.....</i>	43
<i>Obr. 18 První De Jongova funkce [12]</i>	46
<i>Obr. 19 Rastriginova funkce [12]</i>	46
<i>Obr. 20 Schwefelova funkce [12]</i>	47
<i>Obr. 21 Schwefelova funkce [12]</i>	47
<i>Obr. 22 Kombinovaný histogram četností úspěšných kombinací parametrů a, b adaptivních verzí ChaosRnd u První De Jongovi funkce.....</i>	48
<i>Obr. 23 Kombinovaný histogram četností úspěšných kombinací parametrů a, b adaptivních verzí ChaosRnd u Rastriginovi funkce</i>	49
<i>Obr. 24 Kombinovaný histogram četností úspěšných kombinací parametrů a, b adaptivních verzí ChaosRnd u Schwefelovy funkce</i>	49
<i>Obr. 25 Kombinovaný histogram četností úspěšných kombinací parametrů a, b adaptivních verzí ChaosRnd u Ackleyho funkce II.....</i>	50
<i>Obr. 26 Vizualizace rozložení četnosti výskytu kombinací nastavení F a CR při jDE na První De Jongově funkci</i>	51

<i>Obr. 27 Vizualizace rozložení četnosti výskytu úspěšných kombinací nastavení F a CR při jDE na Rastriginově funkci</i>	<i>52</i>
<i>Obr. 28 Porovnání průběhu optimalizace variant DE pro První De Jongovu funkci</i>	<i>54</i>
<i>Obr. 29 Porovnání průběhu optimalizace variant DE pro Rastriginovu funkci</i>	<i>56</i>
<i>Obr. 30 Porovnání průběhu optimalizace variant DE pro Schwefelovu funkci</i>	<i>57</i>
<i>Obr. 31 Porovnání průběhu optimalizace variant DE pro Ackleyho funkci II.....</i>	<i>59</i>

SEZNAM TABULEK

<i>Tab. 1 Mutační strategie [4]</i>	<i>17</i>
<i>Tab. 2 Ukázka nastavení množin dostupných hodnot pro nastavení řídicích parametrů EPSDE [8]</i>	<i>25</i>
<i>Tab. 3 Chaotické systémy</i>	<i>28</i>
<i>Tab. 4 Testovací funkce implementované v aplikaci.....</i>	<i>44</i>
<i>Tab. 5 Kombinace verzí DE a implementovaných generátorů náhodných čísel</i>	<i>53</i>
<i>Tab. 6 Srovnání generátorů náhodných čísel s klasickou DE na První De Jongově funkci po 1500 generacích</i>	<i>54</i>
<i>Tab. 7 Srovnání generátorů náhodných čísel s jDE na První De Jongově funkci po 1500 generacích.....</i>	<i>55</i>
<i>Tab. 8 Srovnání generátorů náhodných čísel s klasickou DE na Rastriginově funkci po 1500 generacích</i>	<i>56</i>
<i>Tab. 9 Srovnání generátorů náhodných čísel s jDE na Rastriginově funkci po 1500 generacích</i>	<i>56</i>
<i>Tab. 10 Srovnání generátorů náhodných čísel s klasickou DE na Schwefelově funkci po 1500 generacích</i>	<i>58</i>
<i>Tab. 11 Srovnání generátorů náhodných čísel s jDE na Schwefelově funkci po 1500 generacích</i>	<i>58</i>
<i>Tab. 12 Srovnání generátorů náhodných čísel s klasickou DE na Ackleyho funkci II po 1500 generacích</i>	<i>59</i>
<i>Tab. 13 Srovnání generátorů náhodných čísel s jDE na Ackleyho funkci II po 1500 generacích</i>	<i>59</i>

SEZNAM PŘÍLOH

P I: CD s textem práce, zdrojovými kódy a spustitelnou aplikací