Testování výkonnosti evolučních algoritmů při aplikaci na deterministický chaos

Bc. Zbyněk Molnár

Diplomová práce 2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

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

Jméno a příjmení:	**Bc. Zbyněk MOLNÁR**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Testování výkonnosti evolučních algoritmů při aplikaci na deterministický chaos**

Zásady pro vypracování:

1. **Vypracujte literární rešerši na dané téma.**
2. **Vypracujte studii o testování výkonnosti EA při aplikaci na optimalizaci řízení deterministického chaosu.**
3. **Proveďte popis optimalizace řízení chaosu pomocí EA, a uveďte co nejvíce příkladů chaotických systémů.**
4. **Využijte několik evolučních algoritmů (SOMA, DE, PSO... atd).**
5. **Jako výstup vytvořte aplikace v prostředí Mathematica a taktéž přehlednou statistiku včetně grafických výstupů.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **SCHUSTER, H. G. Handbook of Chaos Control. 2. Rev. Edition. Villey VCH, 2007. 849 s. ISBN 978-3527406050.**
2. **GONZALES-MIRANDA, J. M., Synchronization And Control Of Chaos: An Introduction For Scientists And Engineers. World Scientific Publishing Company, 2004. 224 s. ISBN 978-1860944888.**
3. **ZELINKA, Ivan. Aplikovaná Informatika. Zlín. UTB, 1999. 183 p. ISBN 80-214-1423-5.**
4. **ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. BEN, 2002, 190 p. ISBN 80-7300-069-5.**
5. **ZELINKA, I., OPLATKOVÁ, Z., OŠMERA, P., ŠEDA, M., VČELAŘ, F. Evoluční výpočetní techniky - principy a aplikace. BEN - technická literatura, Praha, 2008, ISBN 80-7300-218-3.**
6. **KVASNIČKA, V., POSPÍCHAL, J., TIŇO, P., Evolučné algoritmy. Bratislava : STU Press, 2000.**
7. **MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence, Academia, 1993, ISBN 80-200-0496-3.**
8. **MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence 4., Academia, 2003, ISBN 80-200-1044-0.**

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

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Tato práce se zabývá testováním výkonnosti vybraných evolučních algoritmů. Tyto evoluční algoritmy jsou testovány na aplikaci optimalizace řízení deterministického chaosu. Hlavní náplní teoretické části této práce je popis funkce a řídících parametrů evolučních algoritmů a dále pokračuje popisem diskrétních chaotických systémů (Logická rovnice, Henonova mapa), které jsou využity v praktické části při optimalizaci. Na začátku praktické části je popsán návrh účelových funkcí, které jsou použity pro testování. Pro optimalizace byly vybrány algoritmy SOMA, Diferenciální evoluce a Chaotická Diferenciální evoluce. Následuje optimalizace řídících parametrů těchto evolučních algoritmů, kterými jsou nastaveny evoluční algoritmy pro optimalizaci řízení chaosu. Nakonec jsou výsledky simulace porovnány mezi jednotlivými evolučními algoritmy.

Klíčová slova: evoluční algoritmy, deterministický chaos, Logická rovnice, Henonova mapa, SOMA, Diferenciální evoluce, Chaotická diferenciální evoluce, účelová funkce, optimalizace, nastavení parametrů

ABSTRACT

This thesis deals with the testing performance of selected evolutionary algorithm. These evolutionary algorithms are tested on the aplication of the optimization of deterministic chaos control. The main concern of the theoretical part of this work is a description of functions and control parameters of evolutionary algorithms and continues the description of discrete chaotic systems (Logistic equation, Henon map), which are used in the practical part for the optimization. At the beginning of the practical part describes the design of cost functions, which are used for testing. For optimization were selected algorithms SOMA, Differential evolution and Chaotic Differential evolution. Followed by the optimization of control parameters of evolutionary algorithms, which are set into evolutionary algorithms for optimization of chaos control. Finally, simulation results are compared between different evolutionary algorithms.

Keywords: evolutionary algorithms, deterministic chaos, Logistic equation, Henon map, Lorenz system, SOMA, Differential evolution, Chaotic Differential evolution, cost function, optimization, parameter estimation

Na tomto místě bych chtěl v první řadě poděkovat panu Ing. Romanu Šenkeříkovi, Ph.D. za odborné vedení práce, cenné rady v oblasti chaotických systémů a v neposlední řadě za připomínky a návrhy, které pomáhali dotvářet tuto práci. Dále bych chtěl poděkovat svým rodičům za podporu po celou dobu mého dosavadního studia a v neposlední řadě svým přátelům, kteří mi poskytli cenné rady i kritiku.

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 s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování 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 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 …….

podpis diplomanta

OBSAH

ÚVOD

V současné době se můžeme setkat s optimalizačními problémy, jejichž výpočetní náročnost neroste pouze zvyšujícím se počtem optimalizovaných parametrů, ale také různými omezeními v určitých částech intervalu těchto parametrů, vyplývajících z ekonomických nebo fyzikálních zákonitostí. Tyto optimalizované parametry navíc nemusí být ani stejného typu (reálný, celočíselný, diskrétní, logický).

Dříve se k optimalizačním problémům přistupovalo klasickým matematickým aparátem, který umožňoval řešení na problémech jednoduššího charakteru. Složitější problémy se sice daly řešit analytickým přístupem, ale museli jsme počítat jednak s delším časem potřebným k vyřešení problému ale i s výkonnějším řešitelem.

Na přelomu padesátých a šedesátých let minulého století byly vytvořeny první strategie algoritmů, nazýváme souhrnně evoluční algoritmy. Tento název byl odvozen podle principů, které svojí filozofií „vývinu“ řešení připomíná evoluci v přírodě. Vhodným funkčním předpisem se dá každý problém převést na problém matematický, jehož optimalizace vede k nalezení argumentů účelové funkce. Ta musí být přesně nadefinovaná pro optimalizační problém. Mezi výhody evolučních algoritmů můžeme počítat určitě širokou použitelnost a na rozdíl od analytických metod se u evolučních algoritmů můžeme řešení dočkat v relativně kratším čase. Mezi nevýhody se počítá jistá forma náhody, s kterou tyto algoritmy pracují, a proto výsledek není dopředu předvídatelný. Ruku v ruce s evolučními algoritmy se pro zvýšení výkonnosti řešení optimalizačních problémů využívají počítačové systémy založené na paralelizmu.

Úkolem této práce je otestovat výkonnost vybraných evolučních algoritmů. Pro tyto algoritmy vytvořit rozsáhlé simulace a výsledky prezentovat tak, aby byla možnost přímého porovnání výkonnosti. Jako sofistikovaný problém pro tyto simulace nám poslouží optimalizace řízení deterministického chaosu. Pro nastavení řídících parametrů evolučních algoritmů využít tzv. „meta“ přístupu, neboli využití podřízeného evolučního algoritmu k nalezení optimálních hodnot řídících parametrů pro evoluční algoritmus. Takto nastavené evoluční algoritmy použít pro simulaci optimalizace řízení chaotického systému. Výsledky statisticky vyhodnotit a výsledky porovnat a vyhodnotit výkonnost evolučních algoritmů na tomto konkrétním problému.

Ke splnění tohoto úkolu je ovšem nutné jisté teoretické nastínění problémů konkrétních evolučních algoritmů a chaotických systémů. Ty jsou popsány v teoretické části této práce.

Praktická část se zabývá hlavně tvorbou účelových funkcí, nastavením chaotických systémů, optimalizací parametrů pro evoluční algoritmy, samotným procesem optimalizace řízení chaosu a prezentací výsledků a porovnáním výkonnosti, opírající se o výsledky provedených simulací.

I. TEORETICKÁ ČÁST

1 OPTIMALIZAČNÍ ALGORITMY

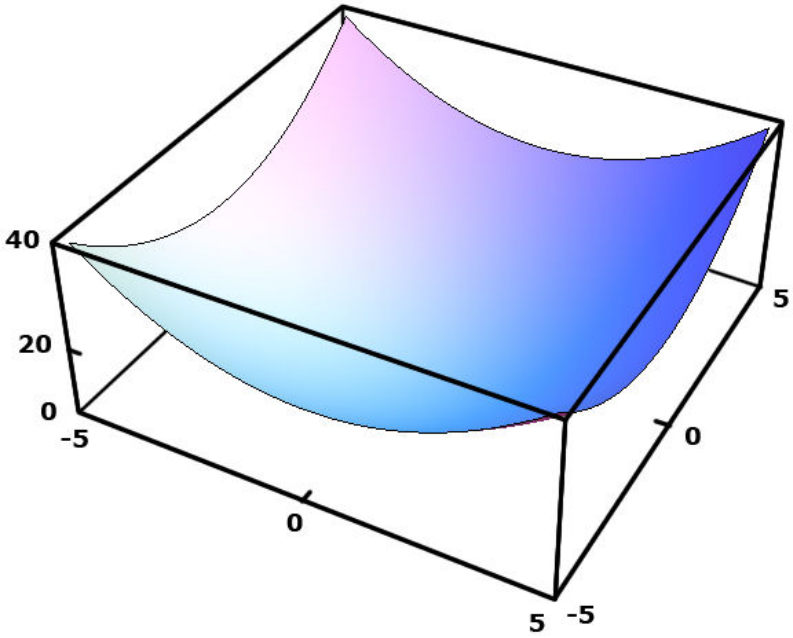
V inženýrské praxi se mohou objevit problémy, které se nedají vyřešit analytickou cestou, protože je to z hlediska času nereálné či nevhodné. Kvůli tomu byly vytvořeny optimalizační algoritmy, u kterých není nutné častého uživatelského zásahu při běhu. Obvykle jde o problémy nalezení optimální trajektorie robota, optimální tloušťky stěny tlakové nádoby či optimálního tvaru křídla letadla. Tyto problémy jsou obvykle definovány jako optimalizační úlohy. Řešený problém se převede na matematickou úlohu danou vhodným funkčním předpisem. Řešením problému potom bude optimalizace argumentů účelové funkce. Právě proto byla vytvořena skupina velice výkonných algoritmů, které obecně označujeme jako evoluční algoritmy.

1.1 No free lunch teorén

Tento teorén reprezentuje myšlenku, že neexistuje žádný algoritmus, který by řešil všechny problémy lépe než jiný algoritmus. Tedy že nemůžeme čekat, že každý algoritmus se hodí k řešení každého problému. Evoluční algoritmy jsou testovány na rozsáhlé třídě testovacích funkcí. Tyto výsledky se porovnávají, ale nemůžeme říct, že algoritmus A je zcela nepoužitelný. Určitě existuje množina problémů, na kterých algoritmus A pracuje velice úspěšně.

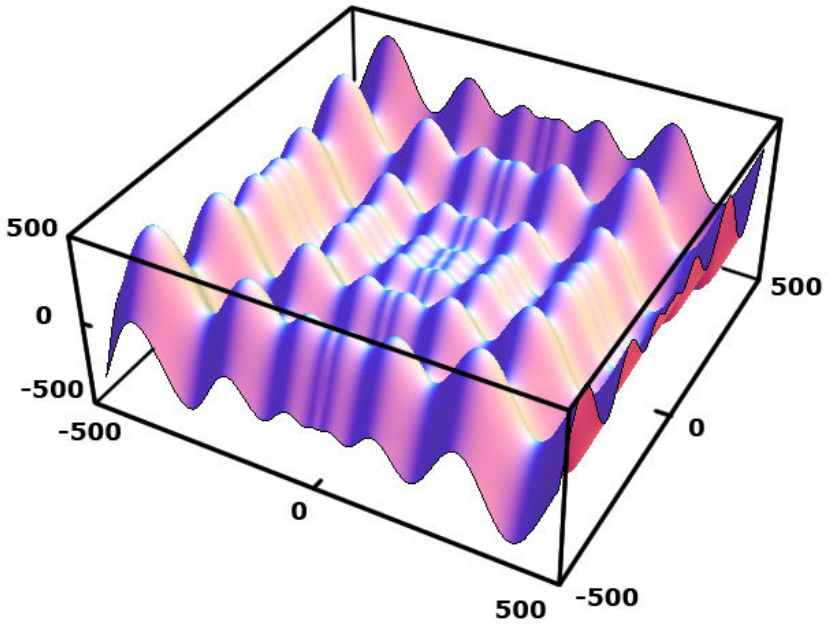
1.2 Účelová funkce

Je funkce, jejíž optimalizace vede k nalezení optimálních hodnot argumentů. Obvykle na ni nahlížíme jako na geometrický problém, v jehož rámci hledáme minimální, či maximální pozici na $N+1$ rozměrné ploše. N je počet argumentů optimalizované funkce. Pokud optimalizujeme funkci, která má pět argumentů, hledáme řešení v pětirozměrném prostoru. Jedna dimenze navíc se přidává pro hodnotu návratové funkce. Účelovou funkci, která má jen jeden extrém říkáme unimodální účelová funkce.



Obrázek 1. Unimodální účelová funkce

Naopak účelová funkce, která má více stejných globálních extrémů se nazývá multimodální účelová funkce.



Obrázek 2. Multimodální účelová funkce

Právě tvorba účelové funkce je jedním z nejsložitějších kroků v optimalizačním procesu. Potřebujeme znát vše o daném problému, čeho chceme dosáhnout a z čeho můžeme vycházet. Špatná tvorba účelové funkce může citlivě ovlivnit řešení problému.

1.3 Populace

Populace zahrnuje všechny jedince, kteří řeší daný problém. Právě práce s těmito populacemi je základním rysem evolučních algoritmů. Populace je obvykle prezentována jako matice *N* x *M*. Kde počet sloupců *M* určuje počet jedinců a počet řádků *N* určuje počet argumentů účelové funkce, které jsou dány problémem a jejichž optimální číselné řešení hledáme.

Počáteční populace musí být vytvořena tak, aby jedinci nezasahovali do oblasti, která není pro řešení problému důležitá, nebo jejichž řešením bychom dosáhli nereálných řešení. Pro tento případ se používá tzv. vzor (Specimen)

$$Specimen = \{\{Real, \{Lo, Hi\}\}, \{Integer, \{Lo, Hi\}\}\}, \dots, \{Real, \{Lo, Hi\}\}\} \quad (1)$$

Tento vzor popisuje pro každého vytvořeného jedince nejdříve typ proměnné, a potom rozsah hodnot, v kterých se může jedinec pohybovat. Například Real{Lo,Hi} představuje reálné číslo v rozsahu *Lo* (spodní hranice) a *Hi* (horní hranice)

1.4 Jedinci

Jedinec je vlastně řetězec vhodných symbolů. Každý symbol reprezentuje jeden argument řešeného problému. Při reprezentaci jedinců se používá několik způsobů. První je použit stejně jako u genetických algoritmů binární, který je tvořen pouze sekvencí nul a jedniček. Ovšem pro eliminaci tzv. skokové změny struktury se používá Grayův kód. Nevýhody binární reprezentace jedinců jsou v přesnosti. Pokud má jedinec málo míst za desetinou čárkou, může nabývat pouze určitých hodnot. Se zvýšením délky jedince ovšem rostou další problémy s operacemi v evolučních algoritmech. Dále můžeme jedince prezentovat jak formou celočíselnou tak reálnou popřípadě kombinací těchto dvou forem. U celočíselné formy používáme dvě strategie. První strategie vytvoří zaokrouhlení argumentů účelové funkce již v populaci a druhá strategie neprovede zaokrouhlení v populaci, ale až před dosazením do účelové funkce. Tím se zvýší diverzibilita a robustnost evolučního algoritmu. Dokonce existuje i typ evolučních algoritmů, který v jedinci obsahuje nenumerické hodnoty. Poslední formou je reprezentace pomocí stromu.

1.5 Omezování a ošetřování krizových stavů

V evolučních algoritmech je obvyklé, že kromě řešení, kterých chceme dosáhnout, se jedinci mohou dostat i do oblastí, které jsou pro nás nereálné a neopodstatněné. Součástí

evolučních algoritmů jsou i metody, které jedince udržují ve vymezeném prostoru možných řešení.

1.5.1 Omezení kladená na argumenty účelové funkce

Pokud se v jedinci objeví parametr, který překročí povolené hranice, potom se tento parametr vygeneruje zcela náhodně v místě, které je uvnitř povolených mezí. To zlepšuje diverzibilitu algoritmu, protože nový jedinec nevznikl křížením jedinců.

Druhým způsobem bylo vrácení jedince na hranici. Což se ovšem nepoužívá, protože tím narůstá čas k nalezení globálního extrému.

1.5.2 Penalizace

Pokud máme nějaké argumenty účelové funkce, které jsou pro nás neakceptovatelné (záporná tloušťka stěny), můžeme využít penalizaci funkce. To je úprava hodnot argumentů účelové funkce ve vybraných oblastech. Penalizace je vlastně takový nátlak na jedince, kteří se dostanou do těchto oblastí. Tito jedinci jsou v hodnotě účelové funkce znevýhodněni obrovskou hodnotou směrem k opačnému extrému. Potom buď opustí svá dosavadní místa, nebo nepostoupí do dalšího evolučního kola. Existují dvě metody:

- **Hard-constraints -** jedinec, který se nachází v zakázané oblasti je zrušen a vygenerován v povolené oblasti
- **Soft-constraints** - jedinec, který se objeví v zakázané oblasti, není zrušen, ale znevýhodněn vůči ostatním řešením.

2 SOMA

SOMA je vlastně zkratka, která znamená Samo-organizující se migrační algoritmus (**S**elf-**O**rganizing **M**igration **A**lgorithm). Tento algoritmus byl vytvořen I. Zelinkou v roce 1999. Řadíme jej mezi evoluční algoritmy a to i navzdory tomu faktu, že v průběhu samotného běhu algoritmu nedochází k vytváření nových potomků, jak je tomu u klasických evolučních algoritmů. Pracuje totiž s populacemi velmi podobně jako genetické algoritmy a výsledkem po jednom migračním kole (v analogii evolučních algoritmů evolučního cyklu) je velmi podobný s Diferenciální evolucí či genetickými algoritmy. Činnost algoritmu SOMA je založena na vektorových operacích, stejně jak je tomu u Diferenciální evoluce nebo u Particle swarm optimalization (PSO). Mnohem přesnější je ovšem řazení tohoto algoritmu mezi memetické nebo hejnové algoritmy. Klasický výběr rodičů z populace zde byl nahrazen principem, který by se dal v biologické analogii nazvat harémovou tvorbou potomků ve stádě.

2.1 Princip algoritmu SOMA

SOMA byl vyvinut na principech, které lze odpozorovat v přírodě a kterými se v sociálně-biologickém prostředí řídí inteligentní jedinci, jež kooperují na řešení společného úkolu. [4] Jako jednoduchý příklad principu algoritmu SOMA z přírody si můžeme popsat chování inteligentních jedinců reprezentovaných smečkou vlků. Jednotliví vlci putují prostorem a hledají místo s nejlepším zdrojem potravy pro smečku. I když pracují ve smečce a chtějí pro ni nalézt nejlepší zdroj potravy, nevědomky mezi s sebou soutěží. Každý jedinec chce být tím, který nalezne nejlepší zdroj. Hned vedle této fáze, která se nazývá soutěžení, je fáze nazvaná spolupráce jedinců. V naší modelové situaci to znamená, že jedinci si mezi sebou sdělují informace o jedinci, který zatím našel nejlepší zdroj potravy. Tento jedinec zůstává u svého zdroje a ostatní jedinci opustí své zdroje a migrují krajinou směrem k jedinci s nejlepším zdrojem potravy. Při svém putování se snaží prohledávat krajinu a nalézt ještě lepší zdroj potravy, než má aktuálně nejlepší jedinec. Tato fáze migrace se opakuje tak dlouho, než se všichni jedinci sejdou u nejlepšího zdroje potravy. Toto je velmi jednoduché přirovnání algoritmu SOMA ke smečce inteligentních jedinců. Pokud ovšem jedince nahradíme souborem parametrů, které chceme optimalizovat a krajinu (životní prostor) na účelovou funkci dokáže tento algoritmus díky své migraci najít velmi hluboké globální extrémy.

Tabulka 1. Srovnání biologické terminologie s terminologií SOMA

Biologická realita	Počítačová implementace
Členové smečky společenství	Jedinci v populaci, parametr PopSize
Člen společenství s nejlepším zdrojem	Leader, vedoucí aktuálního migračního kola
Potrava	Vhodnost, hodnota účelové (kriteriální, cenové) funkce, geometricky je to lokální či globální extrém na *N* rozměrné hyperploše
Životní prostor společenství	Hyperplocha daná účelovou funkcí
Migrace členů společenství v životním prostředí	Migrační kola v algoritmu SOMA

Vzhledem k tomu, že SOMA nevytváří nové jedince pomocí křížení a mutace, ale řídí se principy vycházejícími ze spolupráce inteligentních jedinců migrujících po prostoru možných řešení, byl pojem z evolučních algoritmů generace přejmenován na Migrační kolo. Jedinci se během hledání lepších řešení navzájem ovlivňují. Proto mohou v prostoru řešení vznikat skupiny jedinců, které se navzájem rozpadají a opět reorganizují při migraci přes prohledávaný prostor. Skupina jedinců neboli populace se vlastně sama řídí vzájemný pohyb jedinců. Tato vlastnost je v algoritmu SOMA popsána jako samo-organizace.

2.2 Parametry algoritmu SOMA

Kvalita běhu algoritmu SOMA je ovlivněn, stejně jako u ostatních evolučních algoritmů speciálními parametry. Tyto parametry se dělí do dvou základních skupin.

- Řídící parametry
- Ukončovací parametry

Řídící parametry určují kvalitu běhu algoritmu. To znamená, že jejich nastavení přímo ovlivňuje, jak se algoritmus přizpůsobí danému problému resp. hodnotě účelové funkce. Ukončovací parametry už podle názvu ukončují za předem definovaných podmínek běh algoritmu. Parametry obou těchto skupin musíme nastavit ještě před samotným spuštěním algoritmu na optimalizaci daného problému. Tyto parametry mohou nabývat různých hodnot v určitém rozsahu, který popisuje následující tabulka.

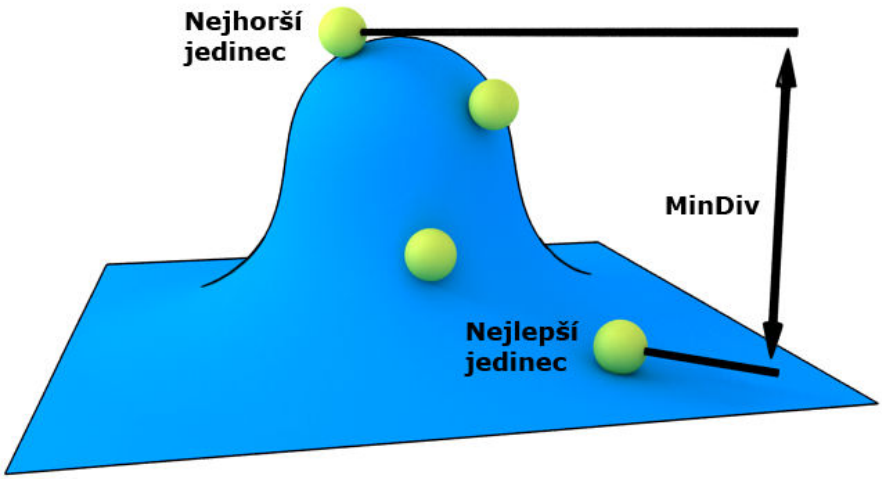
Tabulka 2. Popis parametrů a jejich rozsahy pro algoritmus SOMA

Parametr	Doporučený rozsah	Poznámka
PathLength	< 1,1 ; 5 >	Řídící parametr
Step	< 0,11 ; PathLength >	Řídící parametr
PRT	< 0 ; 1 >	Řídící parametr
D	< Dáno problémem >	Počet argumentů účelové funkce
PopSize	< 10 ; definuje uživatel >	Řídící parametr
Migrace	< 10 ; definuje uživatel >	Ukončovací parametr
MinDiv	< Libovolný;definuje uživatel >	Ukončovací parametr

- **PathLength** tento parametr by měl být nastaven v rozsah < 1,1; 5 > a tato hodnota reprezentuje, na jakou pozici se dostane aktivního jedinec vůči vedoucímu jedinci (leaderovi) během migračního kola. Pokud nastavíme hodnotu parametru *PathLength* = 1 potom se aktivní jedinec zastaví přímo na pozici Leadera. Pokud nastavíme hodnotu *PathLength* > 1, tak se aktivní jedinec zastaví za vedoucím jedincem. Při hodnotě dva se potom aktivní jedinec zastaví ve stejné vzdálenosti od vedoucího jedince jako na začátku ale na opačné straně od vedoucího jedince. Měli bychom se vyvarovat hodnotám *PathLength* < 1, protože to vede k degradaci migračního procesu, protože se aktivní jedinec zastaví před vedoucím jedince a tím se nachází pouze lokální extrémy. Testováním bylo dokázáno, že hodnota *PathLength* = 3 je dostačující pro všechny problémy.
- **Step** tento parametr se nastavuje v rozsahu < 0,11; *PathLength* > a určuje, jak se cesta aktivního jedince navzorkuje. To je, jak bude tato cesta mapována. Pokud máme jednoduchou unimodální funkci, můžeme si dovolit zvolit vysokou hodnotu pro urychlení chodu algoritmu. Ale pokud nemáme ponětí o průběhu účelové funkce, měli bychom volit malé hodnoty tohoto parametru. Čím menší hodnotu nastavíme, tím podrobněji bude tato cesta aktivního jedince prozkoumána a tím máme větší pravděpodobnost, že najdeme globální extrém. Dále je důležité, aby parametr *Step* nebyl celočíselným násobkem *PathLength*. Tedy vzdálenosti, kterou má aktivní jedince projít. To vede k diverzibilitě populace, která potom může rychle skončit v lokálním extrému, protože každý jedinec může být přitažen vedoucím jedincem. Proto je lepší hodnota 0,11 než 0,1.
- **PRT** tento parametr nastavujeme v rozsahu < 0; 1 > a je to nejdůležitější a nejcitlivější parametr. Označujeme jej jako perturbaci. Hlavně z něj se vypočítá perturbační vektor *PRTVector*, který nám udává, zda aktivní jedinec bude

postupovat přímo k vedoucímu jedinci, nebo ne. Nejoptimálnější hodnota je 0,1. Čím je hodnota *PRT* větší, tím narůstá konvergence algoritmu k nalézání lokálních extrémů. Pokud nastavíme hodnotu *PRT* = 1 pak stochastické chování z algoritmu zcela vypadne a algoritmus se chová zcela deterministicky, což se nehodí pro multimodální funkce. Vyšší hodnoty parametru se používají pro nízkodimenzionální funkce s velkým počtem jedinců populace.

- **D** parametr je dán samotným problémem a udává počet argumentů účelové funkce. Tento parametr můžeme změnit jen v případě, že předefinujeme daný problém.
- **PopSize** tento řídící parametr se nastavuje v rozsahu < 10; dáno uživatelem > a reprezentuje počet jedinců v populace. Teoreticky by mohla být minimální populace se dvěma jedinci. V praxi ovšem se takto nastavený algoritmus chová podobně jako deterministické metody. Hodnota *PopSize* se nejčastěji nastavuje jako 0,3 až 0,5 hodnoty parametru *D*. Samozřejmě u jednoduché unimodální funkce není problém nastavit hodnotu na nízké číslo a naopak u složité se klidně *PopSize* může rovnat hodnotě parametru *D*
- **Migrace** je parametr, který se nastavuje v rozsahu < 10; dáno uživatelem >. Určuje, kolikrát se daná populace jedinců přeorganizuje na hyperploše než skončí. Je to tedy ukončovací parametr. V terminologii evolučních algoritmů se tento parametr uvádí pod názvem generace.
- **MinDiv** si může uživatel zvolit podle svého uvážení. Je to vlastně parametr, který udává minimální diverzitu. Je to vlastně rozdíl v hodnotě nejlepšího a nejhoršího jedince v daném migračním kole. Pokud je hodnota rozdílu menší než hodnota nastavená v parametru *MinDiv*, tak se algoritmus ukončí. Samozřejmě pro nastavení této hodnoty bych měl znát účelovou funkci. Musíme vědět, jakých hodnot může nabývat a podle toho se rozhodneme při jaké rozdílové hodnotě, už bude v globálním extrému funkce. Pokud tedy nastavíme tento algoritmus na velkou hodnotu, skončí dřív, než se dostane do globálního extrému. Když jej nastavíme na minimální hodnotu, tak se algoritmus zastaví po vypršení *Migrace*.



Obrázek 3. Minimální diverzibilita

Můžeme využít i zjednodušené nastavování. To nastavíme *MinDiv* na zápornou hodnotu. Tím docílíme, že podmínka nebude nikdy reálně splněna a tento parametr eliminujeme. *PathLength* nastavíme na hodnotu 3 a perturbaci *PRT* na hodnotu 0,1. Potom už jen nastavíme zbývající 3 parametry podle požadavku problému.

2.3 Populace

Algoritmus SOMA tvoří populaci stejně, jako bylo popsáno v kapitole 1.3

2.4 Perturbace

Perturbace je vlastně analogie mutace z evolučních algoritmů. Pojem byl změněn jen kvůli terminologii, výsledný efekt je ale stejný. Oba termíny do algoritmů vnášejí náhodu, pomocí generátoru náhodných čísel. Důvod pro použití jiného termínu v algoritmu SOMA je ten, že v ní nedochází ke klasické mutaci nového jedince, jak je tomu u klasických genetických algoritmů, nýbrž jen k rušení přímého pohybu k vedoucímu jedinci. Perturbaci nejvíce ovlivňuje parametr *PRT*. Ten je jedním s nastavitelných parametrů algoritmu SOMA. Právě z něj se počítá perturbační vektor pro každého jedince zvlášť. Tento vektor se počítá znovu pro každého jedince každé migrační kolo.

$$rnd_j < PRT\ pak\ PRTVector = 1\ jinak = 0, \qquad j = 1, \dots, N \tag{2}$$

To jsou pravidla pro generování perturbačního vektoru. Vždy se začne generovat náhodné číslo v intervalu 0 a 1. To se porovnává s hodnotou *PRT* a pokud je hodnota náhodného čísla menší, pak se *PRTVector* = 1. Pokud je tomu naopak *PRTVector* = 0.

Výkonnost algoritmu SOMA lze zlepšit tím, že se perturbační vektor *PRTVector* generuje před každým skokem. Trajektorie aktivního jedince je tím daleko složitější, nicméně to zvyšuje šanci na nalezení globálního extrému. V původní verzi SOMA byl *PRTVector* generován ještě před tím než začal jedinec putovat a byl pro všechny skoky shodný.

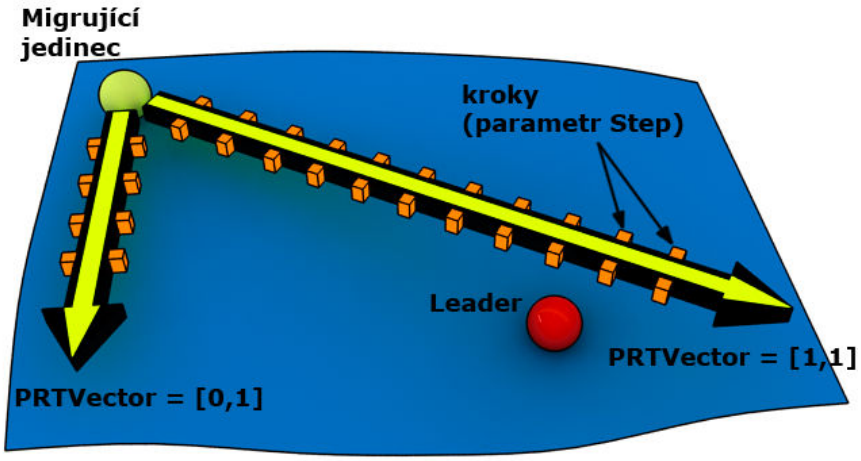
2.5 Křížení

SOMA má opět zcela jiný pohled na křížení, než klasické evoluční algoritmy. Ty pod pojmem křížení vytváří nového potomka, který je vygenerován na pozici, která je složena z pozic již existujících rodičů. Potom se zjišťuje kvalita tohoto potomka a následně přijetí, nebo odmítnutí do množiny pro novou populaci. SOMA se v křížení spíše podobá algoritmu PSO, SS atd. Při putování aktivního jedince k vedoucímu jedinci po hyperploše dochází k navzorkování této trasy po skoku, který reprezentuje parametr *Step*. Při každém tomto skoku se vytvoří nový potomek a z této sady potomků se zachová jen ten nejlepší. Jedincovo putování se řídí rovnicí

$$\vec{r} = \overrightarrow{r_0} + \vec{m}tPRT\vec{V}ector \quad \text{kde } t \in < 0, po\ Step\ až\ po\ PathLength > \tag{3}$$

$$x_{i,j}^{ML+1} = x_{i,j,start}^{ML} + \left(x_{L,j}^{ML} - x_{i,j,start}^{ML}\right) t\ PRTVector_j, \text{kde } t \in < 0, PathLength > \tag{4}$$

Jak je vidět *PRTVector* ovlivňuje směr pohybu jedince. Pokud jsou všechny prvky rovny jedna, aktivní jedinec se pohybuje přímo k vedoucímu jedinci. Pokud jsou ovšem některé prvky nulové, dojde k zmražení jedné souřadnice, což vede k odklonění směru cesty od vedoucího jedince a tím k větší pravděpodobnosti nalezení globálního extrému. Tímto krokem se stává algoritmus daleko robustnější.



Obrázek 4. PRTVector ovlivňující směr pohybu jedince

2.6 Strategie algoritmu SOMA

Strategie je vlastně způsob komunikace a geometrického posouvání jedinců populace po hyperploše. V současné době existuje několik strategií algoritmu SOMA

- „**Všichni k jednomu**" (AllToOne). Tato strategie postupuje po vytvoření populace následovně. Každý jedinec populace se ohodnotí účelovou funkcí, ten který bude na nejlepší pozici, se stane vedoucím jedincem (Leaderem) pro následující migrační kolo. Vedoucí jedinec zůstává v následujícím migračním kole na svém místě a ostatní jedinci se přesunují směrem k němu po skocích určených parametrem *Step*. Po každém tomto skoku zjistí hodnotu účelové funkce, a pokud je lepší než předcházející hodnota, uloží si ji. Takto projedou celou cestu *PathLength* a uložená nejlepší pozice se stane výchozí pozicí jedince pro další migrační kolo. Předtím než aktivní jedinec započne svoji cestu k vedoucímu jedinci je vytvořen vektor o velikosti počtu argumentů účelové funkce, který reprezentuje *PRTVector*, který jak bylo popsáno v předešlé kapitole, zajišťuje směr putování aktivního jedince. To se děje pro každý nový běh algoritmu, takže směr, který měl v předešlém kole, nemusí mít v dalším kole. Po ukončení migračního kola se opět sdělí účelové hodnoty všech jedinců na nových pozicích. To je fáze spolupráce. Jedinec s nejlepším výsledkem se zvolí jako nový vedoucí jedinec (Leader). Tento postup se opakuje stále dokola, dokud není splněna jedna z ukončovacích

podmínek. Pokud rozdíl mezi nejlepším a nejhorším jedincem není menší než parametr *MinDiv* a nebo jestli počet migračních kol algoritmu nepřesáhne počet určený parametrem Migrace.

- „**Všichni ke všem**" (AllToAll) Jak již název napovídá, u této strategie se všichni jedinci přesouvají k ostatním jedinců. Neexistuje zde vedoucí jedinec. Přesouvání v migračním kole je stejné jak u předešlé strategie Všichni k jednomu s tou výjimkou, že po dokončení aktuální migrace se aktivní jedinec přesune na svoji startovací pozici z předešlého migračního kola a provede se posun k dalšímu jedinci. Tato strategie je výpočetně náročnější, nicméně se prohledá daleko více prostoru a tím je větší pravděpodobnost nalezení globálního extrému.
- „**Adaptivně všichni ke všem**" (AllToAllAdaptive) Tato strategie je naprosto totožná s e strategií Všichni ke všem s tím rozdílem, že aktivní jedinec se nevrací na startovní pozici (nejlepší pozice v předešlém migračním kole), ale když je nějaká hodnota aktuální migrace lepší, než nejlepší hodnota kterou si jedinec uložil do paměti, tak se na ni přesune a z této pozice se potom vydává na putování k dalším jedincům v příslušném migračním kole.
- „**Všichni k jednomu náhodně**" (AllToOneRand) Tato strategie má opět vedoucího jedince (Leadera), který ovšem není určen nejlepším výsledkem účelové funkce, ale je vybrán zcela náhodně
- „**Svazky**" (Clusters) je strategie, která se dá použít jako doplněk všech předchozích strategií. Je to vlastně rozdělení všech jedinců do podskupin. V každé z těchto podskupin probíhá samotný algoritmus SOMA. Samotní jedinci se mohou při putování dostat do jiné skupiny. To, ke které skupině jedinec patří, se zjišťuje pomocí vztahu. Parametr IND_i představuje i-tý parametr účelové funkce, který testujeme a parametr CC_i je stejný i-tý parametr jen u vedoucího jedince (Leadera). HB_i a LB_i jsou potom rozsahy parametrů, kterých mohou nabývat. *HB* pro horní hranici a *LB* pro spodní hranici rozsahu. Pokud se stane, že se jedinec dostane daleko od všech skupin, stane se vlastním svazkem a putuje ke všem strategií AllToAll.

3 DIFERENCIÁLNÍ EVOLUCE

Tento typ evolučního algoritmu je velice podobný genetickým algoritmům, se kterými má několik podobných rysů. Byl vytvořen v roce 1995 Kennethem Pricem a Rainerem Stormem. Tito dva autoři se domluvili na spolupráci a chtěli vylepšit genetické žíhání pro aplikaci na složitější problémy. První změnou bylo předělání genetického žíhání z reprezentace binárních čísel do dekadických a tím samozřejmě předělání logických operací na vektorové. Pak už byla jen otázka času, kdy Kenneth přišel na diferenciální mutaci. Ta generuje zkušební řešení, což je přičtení diference dvou náhodně vybraných jedinců (vektorů) ke třetímu jedinci (vektoru). První varianta Diferenciální evoluce je vlastně tato diferenciální mutace a metody selekce genetického žíhání. Principy žíhání se v testech ukázaly jako nadbytečné a proto byly zcela vypuštěny. R. Storm potom vytvořil zvláštní populaci, což byla populace, která soupeřila o místo v populaci až po naplnění této zvláštní populace. Tato druhá varianta poskytovala dobré výsledky na testovacích funkcích, nicméně pro širší záběr optimalizačních problémů byla tato varianta slabá. Proto byla vytvořena třetí varianta Diferenciální evoluce a ta byla prezentována v článku Dr. Dobb's a tím se představila širší veřejnosti na úrovni odborného periodika.

3.1 Parametry Diferenciální evoluce

Stejně jako u ostatních evolučních algoritmů je činnost i kvalita Diferenciální evoluce ovlivněna řídícími parametry.

- **CR** tento parametr bývá označována za práh křížení. Může nabývat hodnot v rozsahu <0; 1>. Pokud máme separabilní funkci, je dobré nastavit tento parametr na hodnotu blízkou nule. V případě že nastavíme parametr CR=0, se potom mutace nedostane do zkušebního jedince a ten je potom kopií aktuálního čtvrtého rodiče. Naopak pokud nastavíme CR = 1, potom zkušební jedinec bude tvořen třemi náhodně vybranými rodiči z populace. Ale tím se evoluční algoritmus změní na náhodné hledání, protože jedinci jsou vybráni náhodně a ne na základě jejich kvality. Těmto hraničním bodům bychom se měli vyvarovat a nenastavovat je.
- **D** je označení parametru popisujícího dimenzi. Je to vlastně počet argumentů účelové funkce, který je dám řešeným problémem.
- **NP** je parametr popisující počet jedinců v populaci. Z charakteru Diferenciální evoluce víme, že minimální počet jedinců by neměl být menší než čtyři. Čtyři je

totiž minimální hodnota populace, při které Diferenciální evoluce ještě pracuje. Rozsah populace by měl být v rozmezí <10 *D* až 100 *D*>, ale jde hlavně o zkušenost s tímto algoritmem a snaha nastavit ho co nejefektivněji parametr na daný problém.

- **F** je parametr určený pro mutační konstantu. Hodnota parametru se pohybuje v intervalu < 0; 2 >.
- **Generation** je ukončovací parametr. Udává, kolikrát proběhne vytvoření nové generace do ukončení algoritmu.

Doporučené hodnoty pro nastavení Diferenciální evoluce jsou vypsány v následující tabulce.

Tabulka 3. Doporučené hodnoty pro Diferenciální evoluci

Řídící	Interval	Doporučená	Poznámka
NP	<10 D; 100 D>	10D	Velikost populace, pokud je vysoce multimodální funkce potm 100 *D*
F	<0 , 2>	0,3 -0,9	Mutační konstanta
CR	<0 ,1>	0,8 – 0,9	Práh křížení, *CR*<<1 jestli je funkce separabilní a *CR* = 1 pro neseparabilní
Generace	Uživatel		Počet kol šlechtění populace

3.2 Populace

Tvorba prvotní populace i s omezením hraničních hodnot je zcela stejné jak bylo popsánu u evolučních algoritmů v kapitole 1.3

3.3 Mutace

Mutace je základem tvoření nových jedinců u každého evolučního algoritmu. Zvláštností pro Diferenciální evoluci je použití čtyř rodičů na vytvoření nového potomka. Pro každého jedince jsou náhodně vybráni tři další jedinci, kteří spolu vytvoří tzv. šumový vektor *v*. Tento šumový vektor je vlastně mutací těchto tří vektorů. Mutaci vytvoříme odečtením prvního jedince od druhého, následně vynásobíme mutační konstantou *F* a výsledný vektor se přičte k poslednímu třetímu vektoru.

$$v_j = x_{r3,j}^G + F\left(x_{r1,j}^G - x_{r2,j}^G\right) \tag{5}$$

3.4 Křížení

Další zvláštnosti Diferenciální evoluce je, že proces křížení jedinců dochází až po samotné mutaci. U genetických algoritmů je tomu naopak. První je vytvořen jedinec křížením a až potom je upraven mutací. Již při popisu Diferenciální evoluce jsme napsali, že nový jedinec je tvořen ze čtyř rodičů a pro šumový vektor jsme zatím použili jen tři. Ten čtvrtý přichází na řadu právě při křížení. Vytvoříme tzv. zkušební vektor, který se vytváří šumovým vektorem a právě čtvrtým jedincem. Pro vytvoření zkušebního vektoru se využívá parametr *CR*, který porovnáváme s náhodně vygenerovaným číslem. Vždy pracujeme se sobě odpovídající parametry. Vezmeme si např. první parametr šumového vektoru a první parametr čtvrtého jedince. Pokud je vygenerované číslo menší jak parametr *CR*, potom vybereme do zkušebního vektoru první parametr ze šumového vektoru a naopak. Takto postupujeme pro druhý parametr, třetí atd. Až vyplníme celý zkušební vektor, tak se z něj stane nový jedinec, který soutěží se čtvrtým jedincem o místo v nové populaci.

3.5 Princip Diferenciální evoluce

Princip je totožný s každým evolučním algoritmem. Během generace se snažíme najít co nejoptimálnější parametry jedince (nejlepší hodnotu účelové funkce). Generace jsou vlastně jednotlivé cykly, které se opakují, dokud není splněna podmínka ukončovacího parametru.

1. Nastavíme parametry algoritmu, jak již bylo popsáno výše (viz. *Tabulka 3*.) a nadefinujeme hranice a nepřípustné hodnoty pomocí vzoru (Specimena).
2. Vytvoříme počáteční populaci, ke které použijeme nadefinovaný vzor (Specimen) z předešlého kroku. Populace je vlastně definovaná jako tabulka (*N+1* x *M*), kde kromě parametrů, které jsou dány problémem, musíme počítat ještě s jedním prvkem navíc a to pro hodnotu účelové funkce.
3. Během každé generace se vybírá jeden jedinec za druhým až do konce populace a podrobují se postupnému evolučnímu šlechtění. A pro každého jedince se vytvoří cyklus, který je popsán v následujícím bodě.
4. Vždy se vezme jeden jedinec. K němu náhodně vybereme další tři jedince. Prvního jedince odečteme od druhého a dostaneme tzv. diferenční vektor. Ten potom vynásobíme mutační konstantou *F* a tím získáme *váhovaný diferenční vektor*. *Šumový vektor* potom dostaneme přičtením třetího jedince k *váhovanému*

diferenčnímu vektoru. Potom přijde na řadu křížení a vytvoříme tzv. *zkušební vektor*. Ten se plní prvek po prvku podle podmínky, jestli je vygenerované náhodné číslo menší než parametr *CR*. Pokud ano příslušný prvek se doplní ze *šumového vektoru*, pokud je větší, bere se příslušný prvek z vektoru čtvrtého jedince. Takto vyplněný *zkušební vektor* potom soutěží s *cílovým vektorem* (čtvrtý jedinec) o to, který má lepší hodnotu účelové funkce a ten je uložen do *cílového vektoru*. Tak máme zajištěno, že se do nové generace dostanou jen jedinci, kteří mají lepší vlastnosti z pohledu hodnoty účelové funkce. Tento cyklus opakujeme pro všechny jedince v populaci.

5. Diferenciální evoluci ukončuje parametr *Generation*, který je jediným ukončovacím parametrem v základní verzi algoritmu. Udává, kolikrát se má generace obměnit než algoritmus skončí.

Pokud použijeme Diferenciální evoluci na šlechtění parametrů Diferenciální evoluce, mluví potom o meta-Diferenciální evoluci. Vždy pro populaci parametrů vytvoříme samotnou Diferenciální evoluci. Je jasné, že nám tím stoupne i výpočetní složitost algoritmu.

3.6 Varianty Diferenciální evoluce

Zatím jsem v této práci popisoval základní variantu Diferenciální evoluce a to konkrétně DERand1Bin. To ovšem není jediná strategie. Existují i jiné varianty, které se liší v různých výpočtech. Nejčastěji ve výpočtu šumového vektoru *v*.

- Varianta 1 DE/best/1/exp

$$v_j = x^G_{best,j} + F\left(x^G_{r2,j} - x^G_{r3,j}\right) \tag{6}$$

- Varianta 2 DE/rand/1/exp

$$v_j = x^G_{r1,j} + F\left(x^G_{r2,j} - x^G_{r3,j}\right) \tag{7}$$

- Varianta 3 DE/rand-to-best/1/exp

$$v_j = x^G_{i,j} + \lambda.\left(x^G_{best,j} - x^G_{i,j}\right) + F\left(x^G_{r1,j} - x^G_{r2,j}\right) \tag{8}$$

- Varianta 4 DE/best/2/exp

$$v_j = x^G_{best,j} + F\left(x^G_{r1,j} + x^G_{r2,j} - x^G_{r3,j} - x^G_{r4,j}\right) \tag{9}$$

- Varianta 5 DE/rand/2/exp

$$v_j = x_{r5,j}^G + F\left(x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G\right) \tag{10}$$

- Varianta 6 DE/best/1/bin

$$v_j = x_{best,j}^G + F\left(x_{r2,j}^G - x_{r3,j}^G\right) \tag{11}$$

- Varianta 7 DE/rand/1/bin

$$v_j = x_{r1,j}^G + F\left(x_{r2,j}^G - x_{r3,j}^G\right) \tag{12}$$

- Varianta 8 DE/rand-to-best/1/bin

$$v_j = x_{i,j}^G + \lambda . \left(x_{best,j}^G - x_{i,j}^G\right) + F\left(x_{r1,j}^G - x_{r2,j}^G\right) \tag{13}$$

- Varianta 9 DE/best/2/bin

$$v_j = x_{best,j}^G + F\left(x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G\right) \tag{14}$$

- Varianta 10 DE/rand/2/bin

$$v_j = x_{r5,j}^G + F\left(x_{r1,j}^G + x_{r2,j}^G - x_{r3,j}^G - x_{r4,j}^G\right) \tag{15}$$

Jak je vidět, některé varianty vypadají naprosto shodně. To je ovšem jen na první pohled, protože výpočty nejsou dány nejen šumovým vektorem, ale mohou se lišit v jiných detailech.

3.7 Stagnace

Stagnace je negativní jev Diferenciální evoluce, při níž dochází bez zřejmých důvodů k zastavení vývoje hodnoty účelové funkce k optimálnějším hodnotám ještě před dosažením globálního extrému. Populace zůstává stále diverzibilní, ale optimalizační proces nepokračuje. A to i v případech kdy vznikají nový potomci a populace nekonvergovala k lokálnímu extrému.

4 ROJENÍ ČÁSTIC (PSO)

Zkratka PSO v sobě skrývá anglický název Particle swarm optimalization. Tato optimalizační technika byla vyvinuta v roce 1995 Russellem Eberhartem a Jamesem Kennedym. Inspiraci našli v sociálním chování živočišných společenstev např. rybích a ptačích hejn. Základem je opět práce s populacemi a hledání jejich optimálního řešení vytvářením nových a lepších generací, jako u SOMY a Diferenciální evoluce. Na rozdíl od nich ovšem nemá žádné operátory jako křížení a mutace. Částice zjišťují, která z částic (jedinců) má v prostoru řešení nejlepší vhodnost a její trajektorii následují.

4.1 Princip Rojení částic (PSO)

Vezměme si jako příklad hejno ptáků. Toto hejno ptáků bude mít za úkol najít nejvyšší místo. Hejno je rozprostřeno po prostoru a vždy si ukládá hodnotu nejlepší částice. Ostatní částice se snaží následovat právě tuto nejlepší částici.

Každá částice je tedy reprezentována parametry (souřadnicemi v prostoru) rychlostí, která usměrňuje jejich let a vhodností, která se vypočítá dosazením parametrů do účelové funkce a vždy se uloží jen ta nejlepší hodnota aktuální částice.

Celá populace je vygenerována náhodně na prostoru řešení. Každý jedinec má také vygenerován vektor rychlosti, který má za úkol udávat směr jedince v příštím kroku. Každému jedinci je vypočítána účelová funkce a ten s nejlepší hodnotou je uložen do společné paměti populace. Tato hodnota je označována jako *gBest*. Každý jedinec má ovšem ještě svoji paměť, kde ukládá nejlepší pozici, na které zatím byl. Pokud se dostane na pozici, jejíž hodnota účelové funkce je lepší než dosavadní nejlepší pozice jedince, uloží si ji do paměti *pBest*. Přesun a korekce rychlosti jedince se provádí pomocí následujících rovnic.

$$v_d(t+1) = v_d(t) + c_1.rand.\left(pBest_{i,d} - x_{i,d}(t)\right) + c_2.rand.\left(gBest_d - x_{i,d}(t)\right) \tag{16}$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_d(t+1) \tag{17}$$

$v_d(t+1)$ - rychlost jedince v následujícím kroku
$v_d(t)$ - rychlost jedince v tomto kroku
$x_{i,d}(t+1)$ - pozice v následujícím kroku

$x_{i,d}(t)$ - pozice jedince v tomto kroku
$pBest_{i,d}$ - nejlepší dosavadní pozice daného jedince
$gBest_d$ - nejlepší nalezená pozice v populaci
$rand$ - náhodné číslo v intervalu (0, 1)
c_1, c_2 - učící faktor (obvykle se nastavuje na hodnotu 2)

Po tom co se jedinci přesunou nově vypočítanou rychlostí na nově vypočítanou pozici a v ní vypočítají hodnotu účelové funkce, se celý cyklus opakuje znovu. Z rovnic lze vyvodit, že se jedinec může vydat třemi směry.

- **Individuální** – jedinec pokračuje dále svojí vlastní cestou
- **Konzervativní** – jedinec se vrací na pozici, kde našel zatím svoji nejlepší hodnotu účelové funkce
- **Přizpůsobivý** – následuje jedince, který zatím našel nejlepší hodnotu účelové funkce

Obrázek 5. Pohyb jedince ovlivněný pBest a gBest jedinci

Dalším omezením je parametr *Vmax*, který byl vytvořen jako kontrola proti prudkému zvyšování rychlosti částic. Tyto částice potom měly tendenci vylétávat přes hranice prohledávané oblasti. Postup při překročení hodnoty *Vmax* je následující. Buď se

vygeneruje náhodně nová rychlost, která je menší jako hodnota *Vmax*, nebo se rychlost částice v místě překročení omezí jen na rychlost *Vmax*.

4.2 Parametry Rojení částic (PSO)

Jako každý algoritmus, i pro rojení částic musíme nastavit před samotným během algoritmu několik parametrů, se kterými tato metoda pracuje. Jejich správné nastavení může zlepšit kvalitu řešení.

- **Dimenze** je parametr, který stejně jako u jiných algoritmů reprezentuje popis problému. To je počet argumentů účelové funkce. Každý jedinec má vytvořený vektor s pozicemi a rychlostmi, který má počet argumentů zcela totožný s parametrem dimenze.
- **Rozsah** můžeme nastavit, jak již bylo popsáno v odstavci populace u SOMY pomocí vzorového jedince tzv. specimenem. Rozsah tedy ohraničuje prohledávaný prostor. Udává pro každou dimenzi, v jaké oblasti může ležet řešení.
- **Počet částic** je parametr, který se u algoritmu SOMA nebo Diferenciální evoluce nazývá populace a stejně jako u nich by se měl volit v rozsahu 10.*D*. Typicky volíme 20 a 40 částic. Můžeme klidně volit i 100.*D* částic, ale je jasné, že čím víc částic používáme, tím nám rostě strojový čas a výpočet řešení dosáhneme za delší dobu.
- **Vmax** udává maximální rychlost částic. Příliš velká hodnota *Vmax* způsobuje, že částice mohou vylétávat z oblasti omezené rozsahem a jsou potom znovu náhodně generovány. To má za následek degradaci rojení částic na náhodné prohledávání. Naproti tomu velmi malá hodnota způsobuje, že se daný problém řeší velmi podrobně, ale zase na malé oblasti. Nejčastěji se hodnota *Vmax* volí jako 1/20 rozsahu.
- **Učící faktor c_1 a c_2** rozhodují o směru putování částic prostorem. Parametr c_1 upřednostňuje návrat na zatím nejlepší pozici jedince před postupem k nejlepší hodnotě celé populace. Na rozdíl od něj parametr c_2 upřednostňuje postup k nejlepší pozici celé populace. To ale není ještě zcela definitivní. Oba tyto parametry jsou ještě násobeny náhodně generovaným číslem v intervalu (0, 1), které tuto prioritu mohou dokonce i obrátit. Optimální nastavení těchto parametrů je obvykle $c_1=c_2=2$, ale existují i speciální problémy, které vyžadují speciální

nastavení pro efektivní podání výsledku řešení. Pro tyto případy se volí rozsah mezi [0, 4].

- **Setrvačnost** je další parametr, který se vkládá před předcházející rychlost. Byl představen v roce 1998 a používá se k zapomínání rychlosti. Označuje se jako *w* (váha) a pokud je menší jak jedna, potom dochází k zapomínání rychlosti.

$$v_d(t+1) = w.v_d(t) + c_1.rand.\left(pBest_{i,d} - x_{i,d}(t)\right) + c_2.rand.\left(gBest_d - x_{i,d}(t)\right) \quad (18)$$

Tato hodnota nemusí být nastavena napevno. Hodnota setrvačnosti se v průběhu algoritmu může postupně lineárně měnit. Někdy bývá nastavována počáteční a koncová hodnota setrvačnosti w_{start} a w_{end}. To se potom parametr setrvačnosti *w* vypočítá podle následujícího vztahu.

$$w = w_{start} - \frac{(w_{start} - w_{end}).iterace}{Migrace} \quad (19)$$

Tímto vztahem je zaručeno, že zezačátku algoritmus prohledává prostor po velkých skocích a postupem ke globálnímu extrému se skoky zmenšují.

- **Constriction faktor χ** je parametr který má podobnou vlastnost jako setrvačnost. Vkládá se i na stejné místo v rovnici. Obvyklá hodnota tohoto parametru je 0,729.

$$v_d(t+1) = \chi\left[v_d(t) + c_1.rand.\left(pBest_{i,d} - x_{i,d}(t)\right) + c_2.rand.(gBest_d - xi,d(t))\right. \quad (20)$$

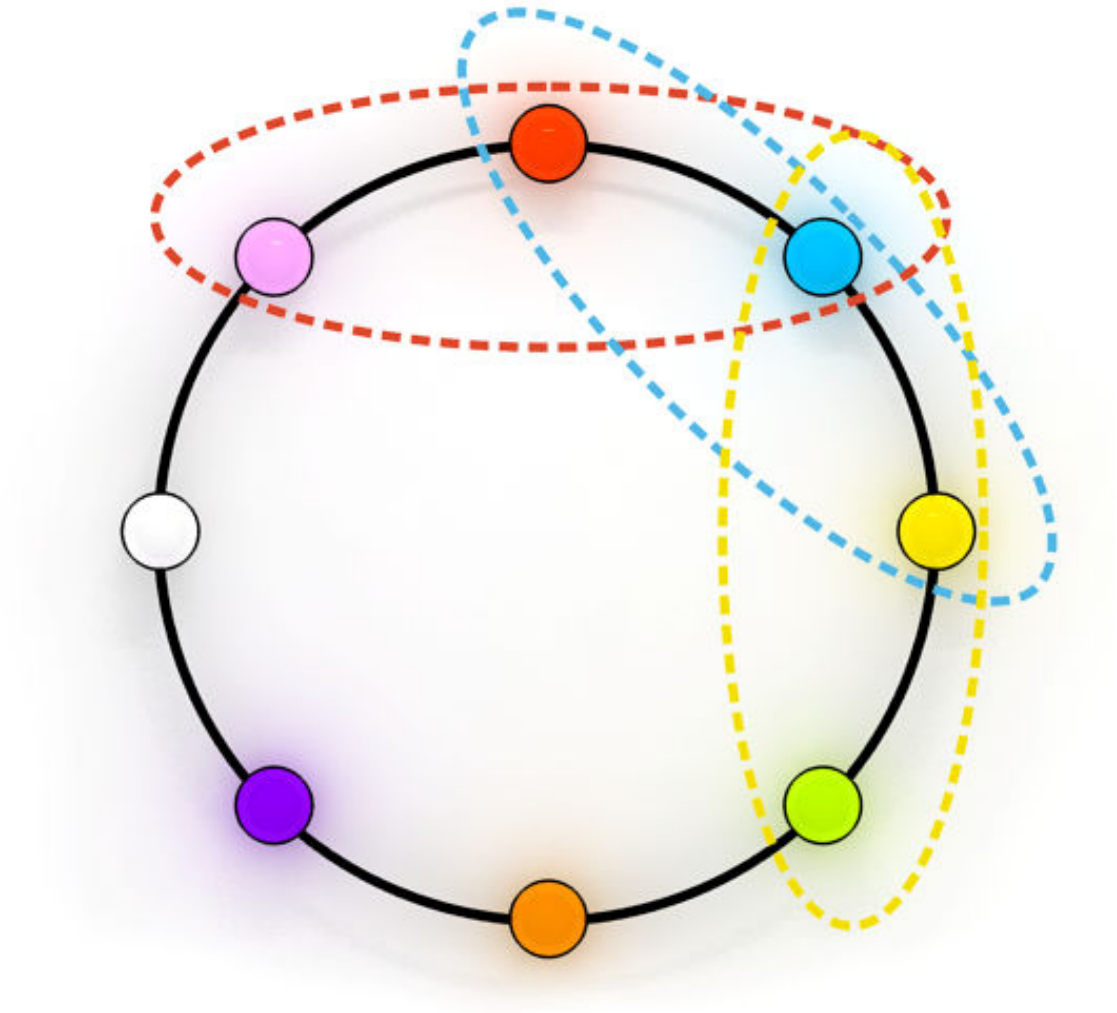
4.3 Varianty algoritmu Rojení částic (PSO)

I algoritmus rojení částic nezůstal dlouho ve své základní variantě a již brzo po jeho vytvoření se začali objevovat jeho novější varianty. Tady je soupis těch neznámějších.

- **Sousedství** je varianta, která se od základní varianty liší tím, že jednotlivé částice jsou rozděleni do skupin. Tyto skupiny se mohou podřizovat dvěma pravidlům.

Buď se řídí geografickým pravidlem, které rozděluje částice do skupin podle oblasti, ve které se nacházejí. Tady je ovšem problém že tyto geografické hranice musíme předem určit, aby bylo jasné, která částice ještě patří do skupiny a která už ne. Druhým pravidlem je sociální. To se používá nejčastěji, protože nezáleží, na jakém místě se částice nachází, ale záleží jen na pořadovém čísle částice. Při použití sousedství se objevuje nový parametr a to je množství jedinců v sousedství. Výhodou je, že tento parametr není příliš citlivý na běh algoritmu. Pro sociální sousedství se používají nejrůznější topologie.

Nejpoužívanější topologií je kruh, kde sousedem částice je částice s nejbližším pořadovým číslem. Sousedství jsou takto navzájem propojená a každá částice ovlivňuje všechny.



Obrázek 6. Topologie kruhu pro variantu sousedství algoritmu PSO

V případě výpočtu (20) se u sousedství nepočítá s nejlepší hodnotou populace *gBest* ale s nejlepším jedincem se sousedů *lBest*. Parametr χ se nazývá constriction faktor a má na rychlost podobné účinky jako parametr setrvačnosti. Doporučovaná hodnota χ je 0,729. [3]

- **Speciation PSO** je další varianta, která dokáže vytvořit různé typy částic, na které se dá populace rozdělit. Tyto skupiny mohou být buď morfologické - druhy vypadají podobně, ale liší se vzhledem, nebo biologické – druhy jsou skupiny jedinců, kteří se mohou uvnitř skupiny křížit a tím jsou odlišeni od ostatních jedinců. Během jednoho cyklu algoritmu jsou vybráni jedinci, kteří jsou ve své skupině nejlepší a tito jedinci posléze zbytek skupiny vedou. Tyto jedince označujeme jako *seeds* a v dalším cyklu už to mohou být úplně odlišní jedinci, než byli tento cyklus.
- **Niching PSO** je varianta která je založena na vytváření podskupinek. Nejdřív je vygenerována počáteční hejno a částice s nejlepší hodnotou účelové funkce vytvoří z nejbližších částic svoji skupinu, která danou oblast prozkoumá podrobněji. Ostatní částice jsou odkázány na sebe a snaží se najít lepší hodnotu. Takto se hejno neustále dělí na podskupiny, které prohledávají místa ve slibných oblastech ve skupinách.
- **INPSO** je varianta která používá více sousedství, která jsou na sobě nezávislá.
- **Hybrid PSO** kombinuje právě předcházející variantu INPSO s algoritmem GPEA. Výhodou INPSO je rychlost, ale na některé řešení nedokázal najít řešení. Naopak GPEA dokáže najít řešení vždy ale průběh je velmi dlouhý.
- **Dynamic Neigbourhood PSO** využívá také více nezávislých sousedství. V tomto sousedství ovšem nesetrvávají natrvalo, ale pokud se přiblíží k nějakému sousedství na vzdálenost, která dává možnost na přijetí částice do druhé skupiny je jí přijata.

5 CHAOTICKÉ SYSTÉMY

Pod slovem chaos si každý vybaví neurčitost, nepřehlednost, zmatek a nepořádek. Pro něj nejdůležitější vlastností je nemožnost předvídání a nenalezení žádného řádu v jeho chování. Když mluvíme o chaotickém chování skutečných systémů v přírodě, můžeme se setkat s termínem deterministický chaos. Je to soustava konzistentních struktur, které se chovají v souladu s přírodními zákony. Jinak řečeno jde o proces samo-organizace těchto složitých systémů. Název deterministický chaos proto vyznívá paradoxně, protože slučuje dva úplně neslučitelné termíny. Jde ovšem o to, že systém se jako celek vyvíjí zcela podle přírodních zákonů, tedy deterministicky. Ovšem z hlediska jedné určité struktury obsahuje systém prvky nahodilosti. Tento paradox můžeme nalézt v přírodních systémech všech možných forem, jako je chování kolonie mravenců, různé chemické reakce nebo Elliotova vlna na kapitálových trzích. Chaotické systémy můžeme rozdělit do dvou základních skupin.

- Diskrétní systémy
- Spojité systémy

Následující kapitoly popisují nejpoužívanější a nejznámějších příkladů chaotických systémů.

5.1 Diskrétní systémy

5.1.1 Logická rovnice

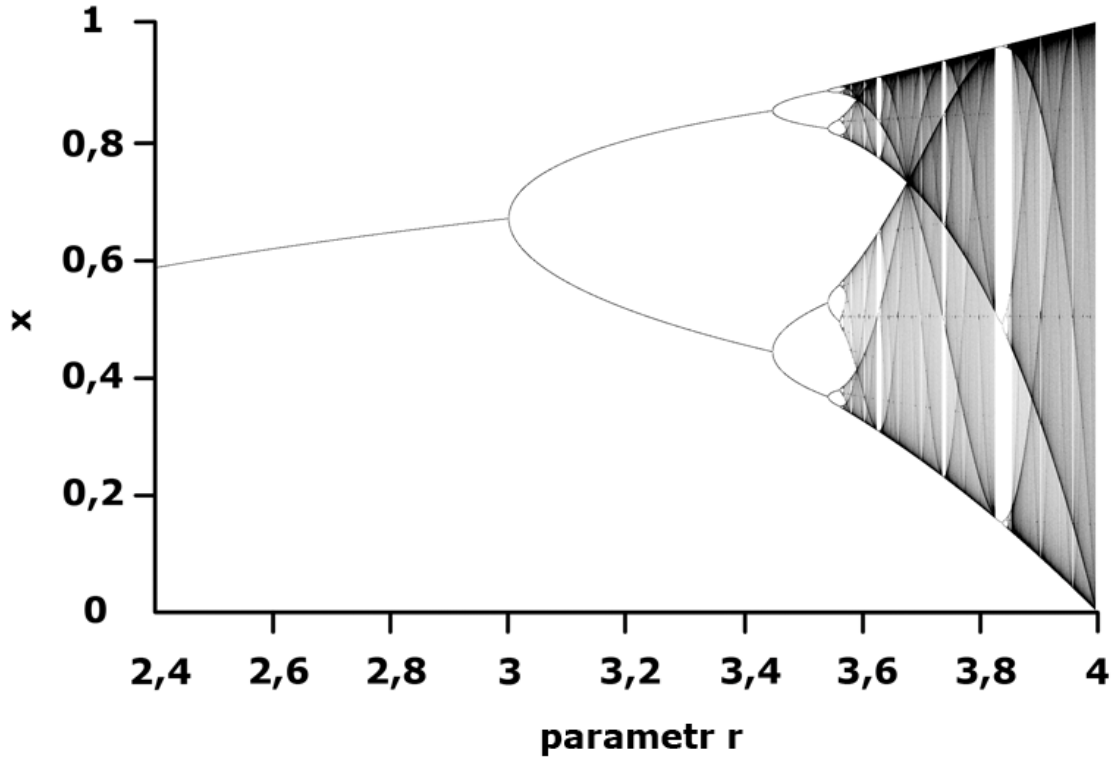
Tento chaotický systém vznikl jako popisný vzorec pro vývoj populace v čase. Mnoho zvířat totiž rodí svoje mláďata v určitém období, tak aby hned po narození měli dostatek potravy, kterou se živí. Ale protože v přírodě není vždy všechno ideální, nestačí nám klasické rovnice, ale musíme použít nelineární Diferenciální rovnice. Na Logické rovnici (Logické mapě) lze jednoduše dokázat, že složité chaotické chování může vznikat i v jednoduché nelineární dynamické rovnici. Je to vlastně jednodimenzionální diskrétní časový příklad. Tento chaotický systém byl představen a propagován Robertem Mayem. [9] Pierre François Verhulst jej zase představuje jako demografický model jako typický vztah dravce a kořisti. Logickou rovnici popisuje vztah (21)

$$x_{n+1} = rx_n(1 - x_n) \tag{21}$$

- x – parametr představující velikost populace

- n – parametr, který představuje rok, ve kterém se počítala velikost populace *x*
- r – parametr, kterým je kladné číslo představující kombinaci sazby pro reprodukci a hladovění

Chaotické chování lze vypozorovat změnou parametru *r,* pokud je parametr na hranici *r*=3,57 potom jsme na hranici začátku chaosu. Pokud je hodnota parametru *r* vyšší než 3,57 systém vykazuje chaotické chování. [9]



Obrázek 7. Bifurkační diagram Logické rovnice

Tento bifurkační diagram popisuje chování Logické rovnice, kdy na vodorovné ose se zobrazuje hodnota parametru *r* a na svislé ose hodnota vstupu *x* v možném dlouhodobém horizontu.

5.1.2 Henonova Mapa

Henonova Mapa je další příklad diskrétního dynamického systému, který byl vytvořen jako matematický model pro vyšetřování chaosu. Je to jeden z nejstudovanějších modelů dynamického systému a je to vlastně dvourozměrné vyjádření jednorozměrné kvadratické

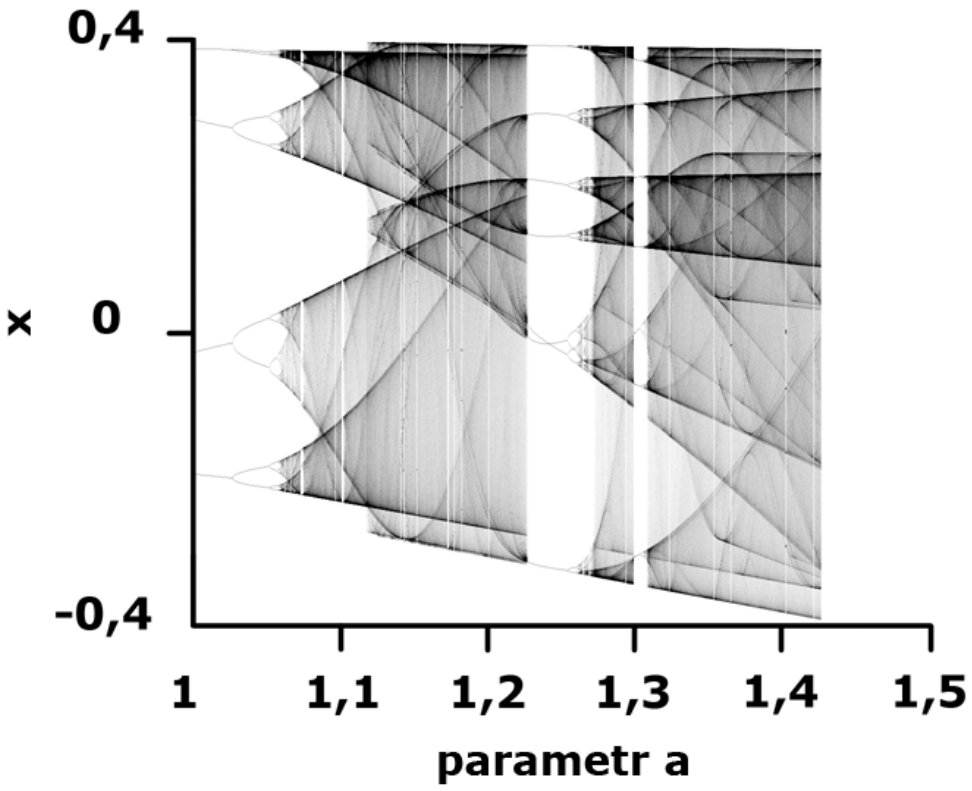
mapy. Ve skutečnosti jde o zjednodušení Poincaréovi mapy pro Lorenzův systém. Henonovu mapu popisuje následující matematické vyjádření (22) a (23).

$$x_{n+1} = 1 + y_n - ax_n^2 \tag{22}$$

$$y_{n+1} = bx_n \tag{23}$$

Mapa závisí na dvou parametrech *a* a *b*, které pro kanonickou Henonovu mapu mají hodnoty parametru *a* = 1,4 a *b* = 0,3. Pro tyto kanonické hodnoty Henonova mapa vykazuje chaotické chování. Pro ostatní hodnoty *a* a *b* může být chování chaotické, přerušované, nebo konverguje k pravidelné dráze. [9]

Henonova mapa bývá často označována termínem „podivné atraktory".



Obrázek 8. Bifurkační diagram Henonovy mapy

Jak můžeme vypozorovat z grafu, na svislé ose grafu je opět vynesen parametr *x* a na vodorovné parameter *a*. Tento graf představuje bifurkační diagram Henonovy mapy jako funkci jednoho kontrolního parametru při neměnné hodnotě druhého parametru.

5.2 Spojité systémy

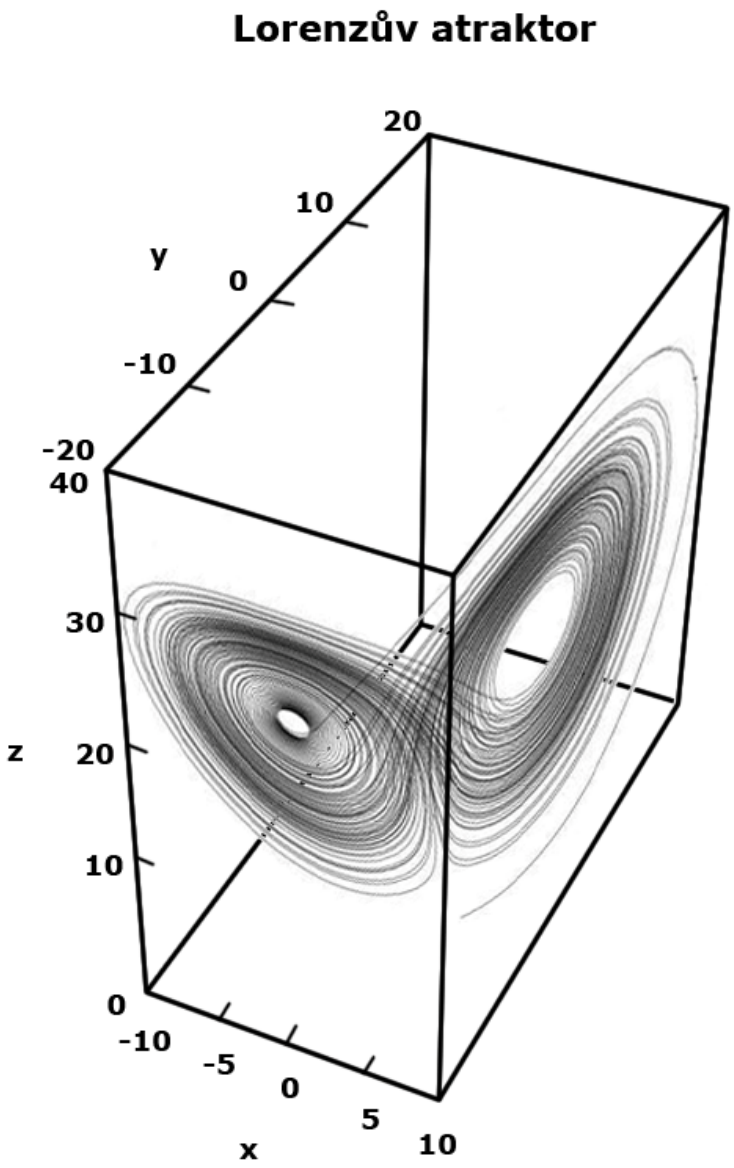
5.2.1 Lorenzův systém

Lorenzův systém byl představen Edwardem Lorenzem v roce 1963 jako tří-dimenzionální dynamický systém, který vykazuje chaotické chování. Tento systém vychází ze zjednodušené rovnice vynucené konvence v atmosféře. [9] Ze začátku se používal jen pro studii problému předvídání počasí. Později se osvědčil také jako popisný systém pro dynama, lasery a specifická vodní kola a dnes se používá jako jednoduchý model dynamiky zespodu zahřívaných tekutin v gravitačním poli. Lorenzův systém popisují následující rovnice (24), (25) a (26)

$$\frac{dx}{dt} = \sigma(y - x) \tag{24}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{25}$$

$$\frac{dy}{dt} = xy - \beta z \tag{26}$$



Obrázek 9. Lorenzův atraktor

5.2.2 Rösslerův systém

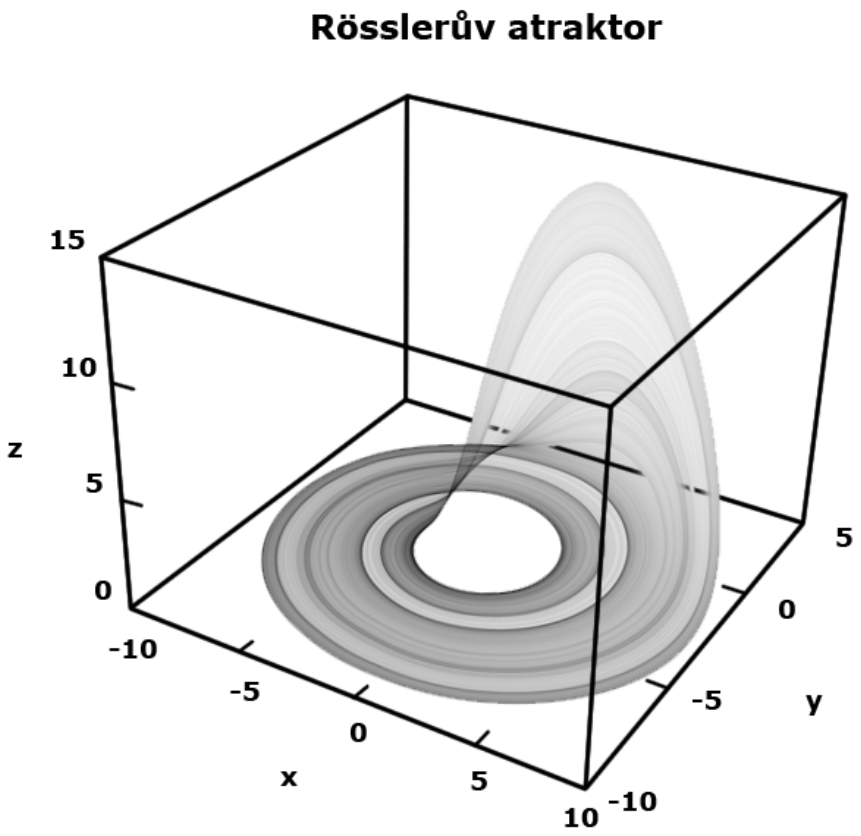
Tento systém byl navržen v roce 1976 jako ukázka velmi jednoduchého příkladu chaotického systému. Až později se zjistilo, že se tento atraktor dá využít pro modelování rovnováhy u chemických reakcí. Rösslerův systém je popsán jako systém tří nelineárních obyčejných diferenciálních rovnic definujících nepřetržitý dynamický systém, který vykazuje chaotickou dynamiku v souvislosti s fraktálními vlastnostmi atraktoru. Tento atraktor je velmi podobný Lorenzovu atraktoru, jen je jednodušší a dá se snadněji kvalitativně analyzovat. Tento systém je popsán třemi diferenciálními rovnicemi (27), (28) a (29).

$$\frac{dx}{dt} = -y - z \tag{27}$$

$$\frac{dy}{dt} = x + ay \tag{28}$$

$$\frac{dz}{dt} = b + z(x - c) \tag{29}$$

Chaotický atraktor s trajektorií rotující kolem pevného bodu studoval právě Otto Rössler. Ten používal svůj systém s hodnotami parametrů a = 0,2, b = 0,2, a c = 5,7. Pro dosažení zdvojnásobení periody atraktoru a chaotického chování můžeme zvýšit hodnotu a. Následující obrázek ukazuje Rösslerův atraktor.



Obrázek 10. Rösslerův atraktor

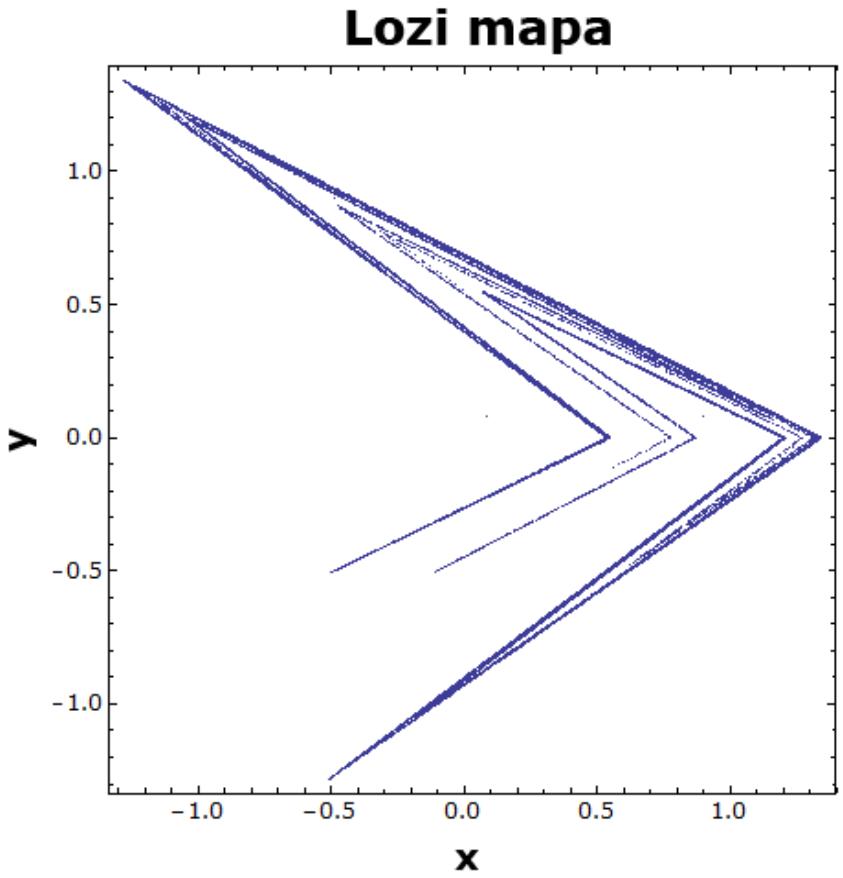
6 CHAOTICKÁ DIFERENCIÁLNÍ EVOLUCE

Pracuje na stejném principu jako klasická Diferenciální evoluce, která byla popsána v kapitole 3. Jediný rozdíl je v procesu křížení a mutace, kdy místo generátoru náhodných čísel, které se porovnávají s parametrem *CR*, se používá chaotický systém tzv. Lozi map. Lozi map je dvou-dimenzionální chaotický systém, který je podobný Henonove mapě. Matematické vyjádření Lozi mapy znázorňují následující rovnice.

$$x_{n+1} = 1 + y_n - a|x_n| \tag{30}$$

$$y_{n+1} = bx_n \tag{31}$$

Lozi mapa opět závisí na dvou parametrech *a* a *b*. Ty jsou nastaveny stejně jak u Henonovy mapy v kapitole 5.1.2.



Obrázek 11. atraktor Lozi mapy

II. PRAKTICKÁ ČÁST

7 OPTIMALIZACE ŘÍZENÍ CHAOSU

7.1 Návrh problému

Praktická část této práce je zaměřena na nalezení nejvhodnějších parametrů pro evoluční algoritmy a dále s jejich pomocí na odhad optimálního nastavení dostupných ovládacích parametrů pro Pyragasovu metodu řízení chaosu EDTAS pro dvě účelové funkce, a to konkrétně pro Targeting CF1 popsané v kapitole 7.2.1 a Targeting CF2, která je popsána v kapitole 7.2.2. Pyragasova metoda řízení chaosu se zpožděnou zpětnou vazbou byla vytvořena pro zkrácení času počáteční chaotické vyčkávací fáze. Zpětnou vazbou docílíme nejen efektu stabilizujícího, ale taky rychlejšího vedení k cíli. Pro úspěšné použití optimalizace nastavení parametrů pomocí evolučních algoritmů využíváme počet dostupných řídících parametrů EDTAS, které mohou být nastavovány jak pomocí znalostí, tak pomocí matematické analýzy. Pomocí optimalizace dosáhneme rychlejší a kvalitnější stabilizace na zvoleném žádaném stavu (UPO) a zlepšení chování systému. UPO je zkratka pro Unstable Periodic Orbit a představuje následující typy:

- p – 1 (ustálený orbit)
- p – 2 (stabilní periodický děj)
- p – 4 (příklady pro vyšší orbit)

Testovanými chaotickými systémy byli jedno-rozměrná Logická rovnice, která byla popsána v kapitole 5.1.1. a dvourozměrná Henonova mapa, která byla popsána v kapitole 5.1.2.. Zde je seznam požadovaných UPOs :

Logická rovnice s parametrem $r = 3{,}8$:

p – 1 (orbit) : $x_F = 0{,}73842$

p – 2 orbit : $x_1 = 0{,}3737,\ x_2 = 0{,}8894$

p – 4 orbit : $x_1 = 0{,}3038,\ x_2 = 0{,}8037,\ x_3 = -0{,}5995,\ x_4 = 0{,}9124$

Henonova mapa s parametry $a = 1.2$ a $b = 0.3$:

p – 1 (orbit) : $x_F = 0{,}8$

p – 2 orbit : $x_1 = -0{,}562414,\ x_2 = 1{,}26241$

p – 4 orbit : $x_1 = 0{,}139,\ x_2 = 1{,}4495,\ x_3 = -0{,}8595,\ x_4 = 0{,}8962$

Pro každou strategie Evolučního algoritmu byla simulace opakována 50 krát. Cílem simulace bylo zjistit robustnost použité metody a nalézt aktuální optimum. Jako metoda řízení je použita ETDAS.

Klasické řízení vykazuje určité problémy u optimalizace pomocí evolučních algoritmů pro stabilizace vyšších periodických orbitů. Pro Logickou rovnici použita metoda ETDAS o následujících vyjádření (32), (33) a (34). Optimalizovanými parametry byly K,F_{max} a R.

$$x_{n+1} = rx_n(1 - x_n) + F_n \tag{32}$$

$$F_n = K[(1 - R)S_{n-m} - x_n] \tag{33}$$

$$S_n = x_n + RS_{n-m} \tag{34}$$

Metoda řízení ETDAS byla použita i pro všechny simulace Henonovy mapy. Optimalizovanými parametry byly také v tomto případě K,F_{max} a R

$$x_{n+1} = a - x_n^2 + by_n + F_n \tag{35}$$

$$F_n = K[(1 - R)S_{n-m} - x_n] \tag{36}$$

$$S_n = x_n + RS_{n-m} \tag{37}$$

U dvou-dimenzionální Henonovy mapy jsou všechny výsledky zobrazovány pouze pro proměnou *x*, protože proměnná *y* je shodná s hodnotou proměnné *x*, jen je fázově posunutá.

Perturbace F_n v rovnici (33) a (36) mohou pro logickou rovnici při libovolně velkých hodnotách způsobovat divergování trajektorie mimo interval {0, 1}. Pro Henonovu mapu je tento rozptyl v intervalu {-1.5, 1.5}. Pro základní nastavení Evolučních algoritmů, nastavíme parametr F_n v intervalu $-F_{max}$, F_{max} a Evoluční algoritmy by potom měli najít hodnotu, která zamezí rozkmitu systému.

Pro všechny simulace byly použity čtyři strategie algoritmu SOMA, šest variant Diferenciální evoluce, šest variant Chaotické Diferenciální evoluce. Následující tabulky zobrazují použité strategie algoritmů. Sloupec index bude při prezentaci výsledků simulací reprezentovat strategii příslušného evolučního algoritmu.

Tabulka 4. Použité strategie algoritmu SOMA

Index	Strategie algoritmu SOMA
1	AllToOne
2	AllToRandom
3	AllToAll
4	AllToAllAdaptive

Tabulka 5. Použité varianty Diferenciální evoluce

Index	Varianty Diferenciální evoluce
5	DERand1Bin
6	DERand2Bin
7	DEBest2Bin
8	DELocalToBest
9	DERand1DIter
10	DEBest1JIter

Tabulka 6. Použité varianty chaotické Diferenciální evoluce

Index	Varianty Chaotické Diferenciální evoluce
11	DERand1Bin
12	DERand2Bin
13	DEBest2Bin
14	DELocalToBest
15	DERand1DIter
16	DEBest1JIter

7.2 Návrh účelové funkce

Pro stabilizaci p – 1 (orbit), p – 2 orbit a p – 4 orbit bylo vyvinuto a testováno několik typů účelových funkcí. Účelová funkce se obecně vytváří z aktuálního výstupu a jeho vzdálenosti k požadovanému stavu. Proto se hodnota účelové funkce bude blížit s nejlepším řešením hodnotě nula. Cílem všech simulací tedy bylo najít takové řešení, které vrátí hodnotu účelové funkce co nejbližší nule.

7.2.1 Targeting CF – CF Targ1

Nejdůležitějším úkolem Targeting CF bylo snížení průměrného počtu iterací potřebných pro úspěšnou stabilizaci. Penalizace se přidává na podobné úrovni jako nepenalizovaná hodnota účelové funkce, aby se předešlo problému spojeného s vrácením hodnoty 0 účelové funkce. Konkrétně tato varianta návrhu účelové funkce by měla předejít problému s definicí malé hodnoty konstanty *SC*, která se objevuje hlavně u stabilizace vyšších periodických orbitů. Hodnota *SC* se tedy počítá jako mocnina nepenalizované základní části účelové funkce

$$SC = 10^{EXPCF} \tag{38}$$

$$EXPCF = log_{10}\left(\sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t| + 10^{-15}\right) \tag{39}$$

Existují dvě základní strategie Targeting CF. V první verzi je vynásoben součet první nalezené minimální hodnoty mezi žádanou a skutečnou hodnotu výstupu systému a automaticky vypočtené *SC* počtem s iterací potřebných pro stabilizaci (*NI*).

$$CF_{TARG1} = NI\left(SC + penalization1 + \sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t|\right) \tag{40}$$

kde :

TS – cílový stav

AS – aktuální stav

τ_1 – první minimální hodnota rozdílu mezi TS a AS

τ_2 – konec optimalizovaného intervalu ($\tau_1 - \tau_s$)

penalization1 = 0 jestli $\tau_i - \tau_2 \geq \tau_s$

penalization1 = $10 * (\tau_i - \tau_2)$ jestli $\tau_i - \tau_2 < \tau_s$ (pozdější stabilizace)

Díky tomuhle řešení by měli Evoluční algoritmy nalézt požadovaný stav daleko rychleji.

7.2.2 Targeting CF – CF Targ2

U druhé strategie Targeting CF je jen malá odchylka od předchozího návrhu v kapitole 7.2.1.. Hodnota *SC* se zde počítá stejně jak v předchozím případě, ale jako jediná je násobena počtem iterací potřebných pro stabilizaci (*NI*). Matematický zápis tohoto návrhu je:

$$CF_{TARG1} = (NI.SC) + penalization1 + \sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t| \tag{41}$$

kde :

TS – cílový stav

AS – aktuální stav

τ_1 – první minimální hodnota rozdílu mezi TS a AS

τ_2 – konec optimalizovaného intervalu ($\tau_1 - \tau_s$)

penalization1 = 0 jestli $\tau_i - \tau_2 \geq \tau_s$

penalization1 = $10 * (\tau_i - \tau_2)$ jestli $\tau_i - \tau_2 < \tau_s$ (pozdější stabilizace)

$$SC = 10^{EXPCF} \tag{42}$$

$$EXPCF = log_{10}\left(\sum_{t=\tau 1}^{\tau 2} |TS_t - AS_t| + 10^{-15}\right) \tag{43}$$

8 NASTAVENÍ PARAMETRŮ EVOLUČNÍCH ALGORITMŮ

Ještě před spuštěním samotného testu bylo nutné nastavit řídící parametry evolučních algoritmů. Některé parametry byly nastaveny na pevnou hodnotu, u jiných se použily zkušební optimalizace pomocí evolučního algoritmu. Jako účelová funkce se zvolila nejtěžší možná kombinace a to dvojdimenzionální Henonova mapa s nevyššími p-orbit při použití Targeting CF 2.

8.1 Nastavení parametrů algoritmu SOMA

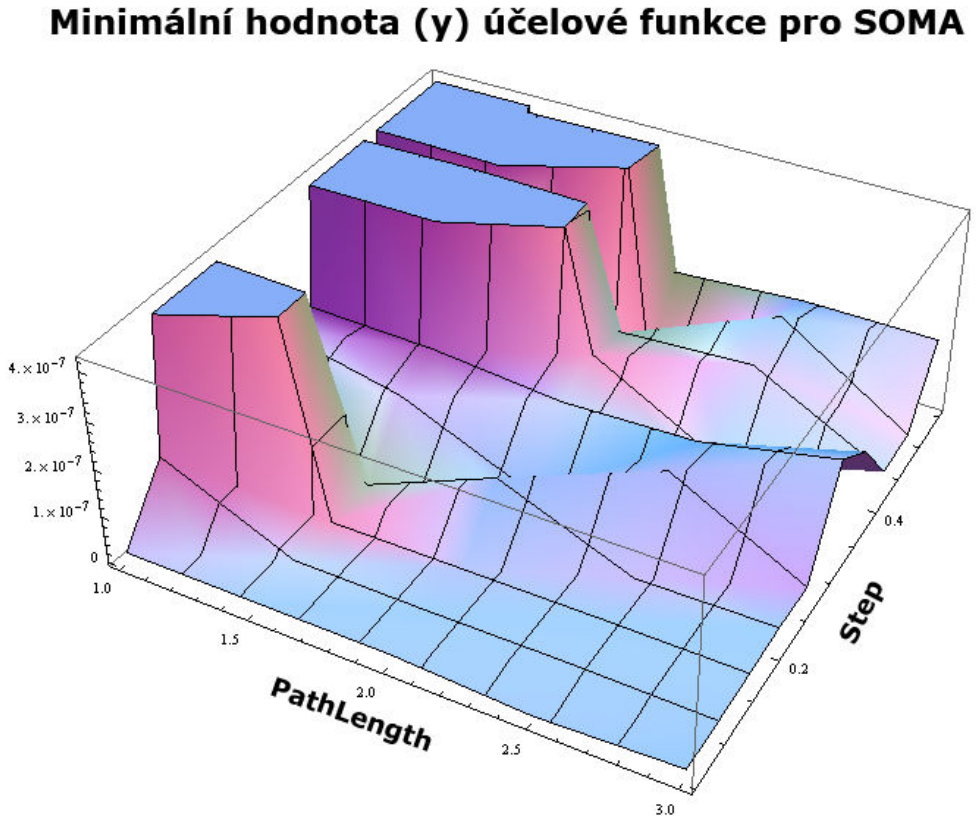
Pro zjištění parametru *Step* a *PathLength* byl použit algoritmus SOMA ve strategie SOMA AllToOne. Ostatní parametry byly nastaveny napevno na určitou hodnotu:

- D = 3
- PopSize = 25
- Migrace = 50
- PRT = 1
- MinDiv = -1

Parametr *Step* byl hledán po kroku 0,055 v intervalu hodnot 0,055 až 0,55, parametr *PathLength* po kroku 0,5 v intervalu hodnot 1 až 3. Pro každou kombinaci těchto dvou parametrů bylo vytvořeno padesát simulací. Výsledky byly ukládány do souboru, jehož název byl vytvořen z nastavených řídících parametrů a uvnitř bylo ukládáno padesát hodnot účelové funkce. Pro každou simulaci byla vyhodnocena minimální, průměrná, maximální a mediánová hodnota účelové funkce. Tyto hodnoty byly vyneseny do následující tabulky. Pro každou tabulku byl ještě vytvořen i 3D graf pro lepší znázornění průběhu hodnoty účelové funkce.

Tabulka 7. Minimální hodnoty účelové funkce pro 50 simulaci SOMA AllToOne

	PathLength					
Step		1	1.5	2	2.5	3
	0,055	$1{,}6375.10^{-8}$	$1{,}6174.10^{-8}$	$1{,}6403.10^{-8}$	**$2{,}0004.10^{-9}$**	$1{,}5809.10^{-8}$
	0,11	$1{,}4253.10^{-7}$	$1{,}8674.10^{-8}$	$1{,}7141.10^{-8}$	$1{,}8790.10^{-8}$	$1{,}5248.10^{-8}$
	0,165	$1{,}3959.10^{-6}$	$1{,}6116.10^{-8}$	$1{,}7979.10^{-8}$	$1{,}9001.10^{-8}$	$1{,}7565.10^{-8}$
	0,22	$1{,}4049.10^{-7}$	$1{,}6721.10^{-8}$	$1{,}3241.10^{-7}$	$1{,}7387.10^{-8}$	$2{,}1191.10^{-8}$
	0,275	$1{,}9525.10^{-7}$	$1{,}3747.10^{-7}$	$1{,}8721.10^{-8}$	$1{,}6648.10^{-7}$	$1{,}7552.10^{-8}$
	0,33	$1{,}7253.10^{-7}$	$1{,}6483.10^{-7}$	$1{,}3977.10^{-7}$	$1{,}4833.10^{-7}$	$2{,}0719.10^{-7}$
	0,385	$1{,}3375.10^{-6}$	$1{,}3169.10^{-6}$	$1{,}6928.10^{-7}$	$1{,}9643.10^{-8}$	$1{,}4310.10^{-7}$
	0,44	$1{,}4697.10^{-7}$	$2{,}2031.10^{-7}$	$1{,}4627.10^{-7}$	$1{,}6223.10^{-7}$	$1{,}8474.10^{-8}$
	0,495	$2{,}1040.10^{-6}$	$2{,}0578.10^{-6}$	$2{,}2845.10^{-8}$	$1{,}9865.10^{-7}$	$2{,}2507.10^{-8}$
	0,55	$1{,}8286.10^{-6}$	$1{,}8188.10^{-8}$	$1{,}4334.10^{-7}$	$1{,}6137.10^{-7}$	$1{,}6033.10^{-7}$

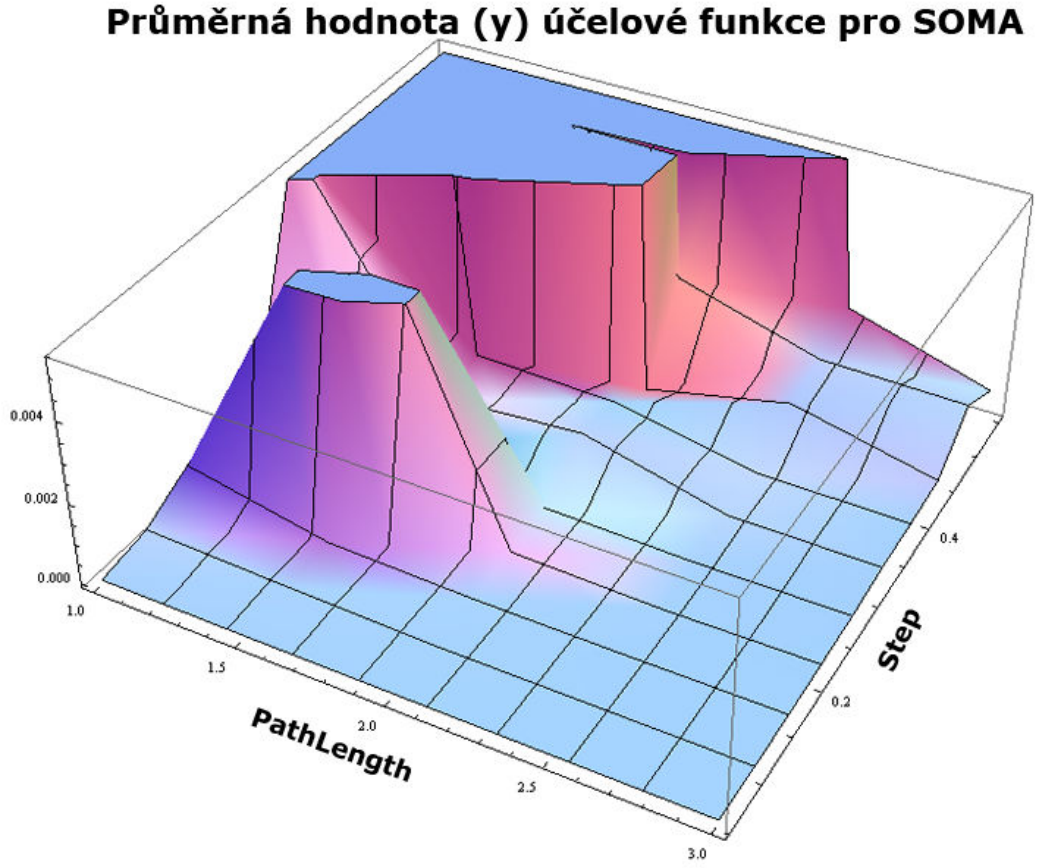


Obrázek 12. Zobrazení minimální hodnoty účelové funkce v 3D grafu

Z tabulky můžeme zjistit, že pro minimální hodnoty účelové funkce algoritmu SOMA je nejnižší hodnota rovna $2{,}0004.10^{-9}$, což odpovídá hodnotám parametrů *Step*=0,055 a *PathLength*=2,5.

Tabulka 8. Průměrné hodnoty účelové funkce pro 50 simulaci SOMA AllToOne

	PathLength					
Step		1	1.5	2	2.5	3
	0,055	$1{,}9866.10^{-5}$	$3{,}3086.10^{-6}$	$1{,}3730.10^{-6}$	$4{,}4417.10^{-7}$	$\mathbf{3{,}7336.10^{-7}}$
	0,11	$2{,}1903.10^{-4}$	$1{,}0313.10^{-4}$	$1{,}0950.10^{-5}$	$4{,}6567.10^{-6}$	$1{,}4167.10^{-6}$
	0,165	$8{,}3305.10^{-4}$	$1{,}2671.10^{-4}$	$2{,}8490.10^{-5}$	$1{,}1701.10^{-5}$	$1{,}4170.10^{-5}$
	0,22	$9{,}4534.10^{-4}$	$8{,}1073.10^{-3}$	$1{,}1916.10^{-4}$	$2{,}1805.10^{-5}$	$5{,}2354.10^{-5}$
	0,275	$2{,}1420.10^{-3}$	$4{,}6053.10^{-4}$	$1{,}6120.10^{-4}$	$8{,}2668.10^{-5}$	$1{,}0043.10^{-4}$
	0,33	$6{,}0944.10^{-3}$	$8{,}1415.10^{-4}$	$1{,}0990.10^{-3}$	$2{,}1798.10^{-4}$	$2{,}0550.10^{-4}$
	0,385	$3{,}3405.10^{-2}$	$1{,}0019.10^{-4}$	$9{,}1431.10^{-4}$	$2{,}5634.10^{-4}$	$2{,}4118.10^{-4}$
	0,44	$3{,}0866.10^{-2}$	$7{,}6507.10^{-2}$	$2{,}9592.10^{-4}$	$1{,}0964.10^{-3}$	$3{,}0813.10^{-4}$
	0,495	$4{,}9326.10^{-2}$	$6{,}2004.10^{-3}$	$2{,}3010.10^{-3}$	$1{,}2938.10^{-3}$	$1{,}2577.10^{-3}$
	0,55	$1{,}3329.10^{-1}$	$1{,}3965.10^{-2}$	$4{,}1085.10^{-2}$	$1{,}7223.10^{-3}$	$6{,}7305.10^{-4}$

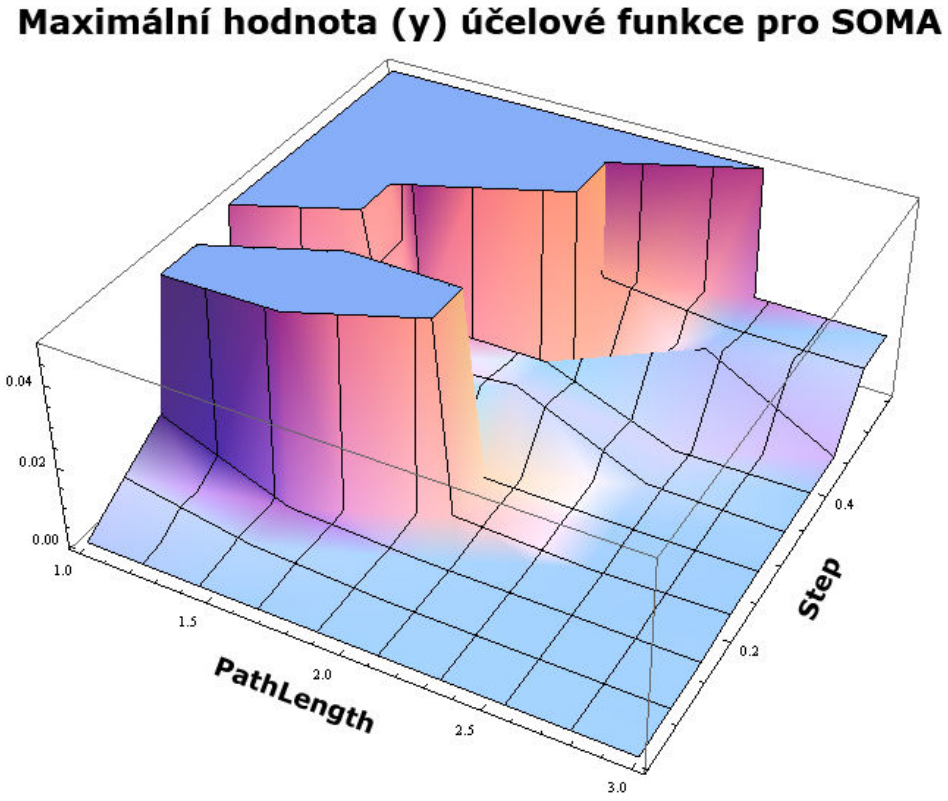


Obrázek 13. Zobrazení průměrné hodnoty účelové funkce v 3D grafu

Nejnižší hodnotou pro průměrnou hodnotu účelové funkce byla hodnota $3{,}7336.10^{-7}$ a té odpovídají hodnoty parametrů *Step*=0,055 a *PathLength*=3.

Tabulka 9. Maximální hodnoty účelové funkce pro 50 simulaci SOMA AllToOne

		PathLength				
Step		1	1.5	2	2.5	3
	0,055	$1,6660.10^{-4}$	$2,2786.10^{-4}$	$2,2932.10^{-5}$	**$1,9609.10^{-6}$**	$2,1664.10^{-6}$
	0,11	$7,2904.10^{-3}$	$1,6957.10^{-3}$	$1,4084.10^{-4}$	$2,2177.10^{-5}$	$1,4442.10^{-5}$
	0,165	$1,4211.10^{-2}$	$1,5417.10^{-3}$	$1,5772.10^{-4}$	$1,4029.10^{-4}$	$1,4275.10^{-4}$
	0,22	$1,4348.10^{-2}$	$3,7803.10^{-1}$	$1,5484.10^{-3}$	$2,1166.10^{-4}$	$1,2957.10^{-3}$
	0,275	$1,9656.10^{-2}$	$1,3450.10^{-2}$	$1,6316.10^{-3}$	$1,4218.10^{-3}$	$1,6542.10^{-3}$
	0,33	$2,0536.10^{-1}$	$1,3898.10^{-2}$	$1,6698.10^{-2}$	$2,1799.10^{-3}$	$1,6677.10^{-3}$
	0,385	$7,5535.10^{-1}$	$1,4241.10^{-2}$	$1,4096.10^{-2}$	$1,5040.10^{-3}$	$1,9454.10^{-3}$
	0,44	$8,2805.10^{-1}$	3,3921	$1,9045.10^{-3}$	$1,9511.10^{-2}$	$1,8181.10^{-3}$
	0,495	$7,5045.10^{-1}$	$8,8667.10^{-2}$	$1,9848.10^{-2}$	$1,6334.10^{-2}$	$1,6137.10^{-2}$
	0,55	3,8194	$3,6840.10^{-1}$	1,9834	$1,6154.10^{-2}$	$1,6058.10^{-2}$

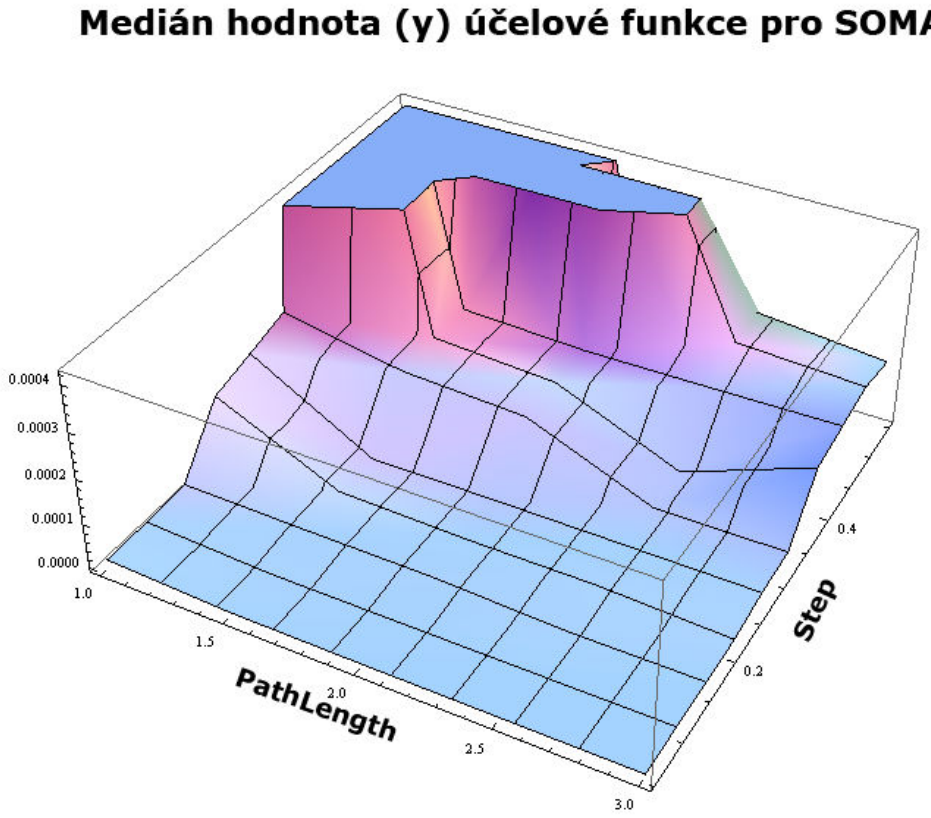


Obrázek 14. *Zobrazení maximální hodnoty účelové funkce v 3D grafu*

Nejnižší hodnotou pro maximální hodnotu účelové funkce byla hodnota $1,9609.10^{-6}$ a té odpovídají hodnoty parametrů *Step*=0,055 a *PathLength*=2,5.

Tabulka 10. Medián hodnot účelové funkce pro 50 simulaci SOMA AllToOne

	PathLength					
Step		1	1.5	2	2.5	3
	0,055	$1{,}3748.10^{-5}$	$1{,}5170.10^{-6}$	$1{,}6917.10^{-7}$	$1{,}4855.10^{-7}$	**$1{,}4212.10^{-7}$**
	0,11	$1{,}4934.10^{-5}$	$2{,}1328.10^{-6}$	$1{,}6477.10^{-6}$	$1{,}4951.10^{-6}$	$1{,}3723.10^{-6}$
	0,165	$1{,}9526.10^{-5}$	$1{,}5938.10^{-5}$	$1{,}3900.10^{-5}$	$1{,}7868.10^{-6}$	$1{,}7487.10^{-6}$
	0,22	$1{,}4644.10^{-4}$	$2{,}1137.10^{-5}$	$1{,}4316.10^{-5}$	$1{,}4247.10^{-5}$	$1{,}9079.10^{-6}$
	0,275	$1{,}7061.10^{-4}$	$1{,}7001.10^{-5}$	$1{,}8351.10^{-5}$	$1{,}7494.10^{-5}$	$1{,}6039.10^{-5}$
	0,33	$1{,}9583.10^{-4}$	$1{,}4600.10^{-4}$	$1{,}3247.10^{-4}$	$1{,}9671.10^{-5}$	$2{,}0378.10^{-5}$
	0,385	$1{,}3759.10^{-3}$	$1{,}5203.10^{-4}$	$1{,}3623.10^{-4}$	$2{,}1228.10^{-5}$	$1{,}2963.10^{-4}$
	0,44	$1{,}3124.10^{-3}$	$1{,}4857.10^{-4}$	$1{,}3804.10^{-4}$	$1{,}4068.10^{-4}$	$1{,}4496.10^{-4}$
	0,495	$1{,}6306.10^{-3}$	$7{,}5570.10^{-4}$	$7{,}3855.10^{-4}$	$1{,}7171.10^{-4}$	$1{,}5846.10^{-4}$
	0,55	$1{,}2640.10^{-2}$	$1{,}3078.10^{-3}$	$1{,}8731.10^{-4}$	$1{,}6909.10^{-4}$	$1{,}4268.10^{-4}$

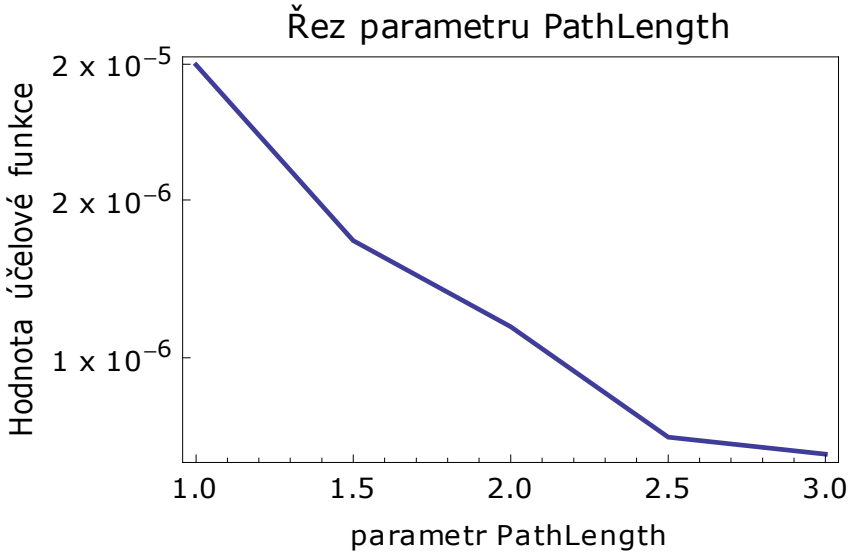


Obrázek 15. Zobrazení mediánu hodnoty účelové funkce v 3D grafu

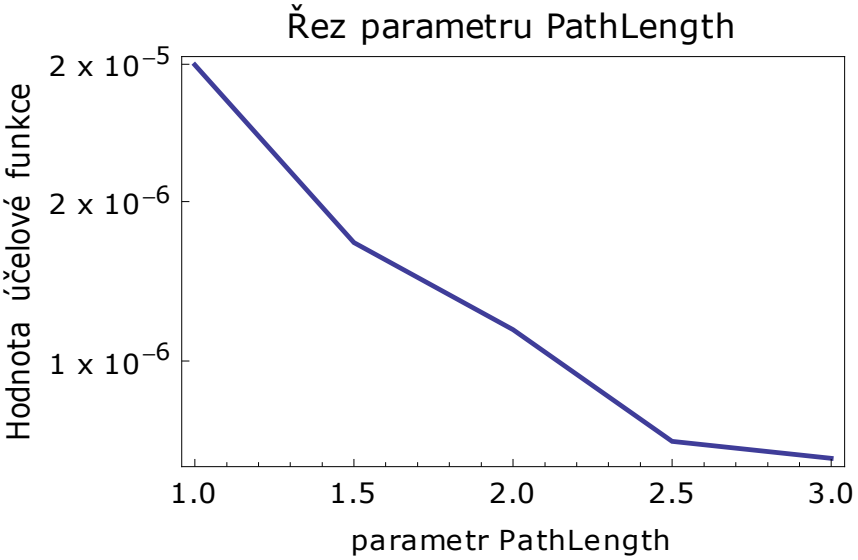
Nejnižší hodnotou pro medián hodnoty účelové funkce byla hodnota $1{,}4212.10^{-7}$ a té odpovídají hodnoty parametrů *Step*=0,055 a *PathLength*=3.

Pro medián i průměrnou hodnotu účelové funkce vyšla stejná kombinace parametrů *Step*=0,55 a *PathLength*=3. Maximální i minimální hodnoty účelové funkce se lišily od průměrné hodnoty parametrů jen o hodnotu 0,5 parametru *PathLength*. Mělo by se přiklonit k hodnotě průměru a mediánu hodnot účelové funkce, ale protože pro toto nastavení SOMA algoritmu vycházelo asi 65 000 ohodnocení účelové funkce pro jeden

běh algoritmu, s přihlédnutím na čas ohodnocení účelové funkce pro čtyři verze SOMA by to znamenalo nalezení výsledku v čase, který přesahuje termín odevzdání této práce. Proto byl pro simulace nastaven parametr *Step* s hodnotou 0,11 a *PathLength* s hodnotou 3. Pro tyto hodnoty byly vytvořeny následující řezy, které potvrzují výběr právě tohoto nastavení parametru *Step* a *PathLength* .



Obrázek 16. Řez parametru PathLength



Obrázek 17. Řez parametru Step

8.2 Nastavení parametrů Diferenciální evoluce

Stejně jako u předcházejícího nastavení parametrů pro algoritmus SOMA se i pro nastavení parametrů Diferenciální evoluce a chaotické Diferenciální evoluce použily zkušební optimalizace pomocí evolučního algoritmu. V tomto případě to byla Diferenciální evoluce

ve variantě DERand1Bin. Parametry, kterým se měnila hodnota, byly parametry *F* a *Cr*. Ostatní byly nastaveny na určitou hodnotu:

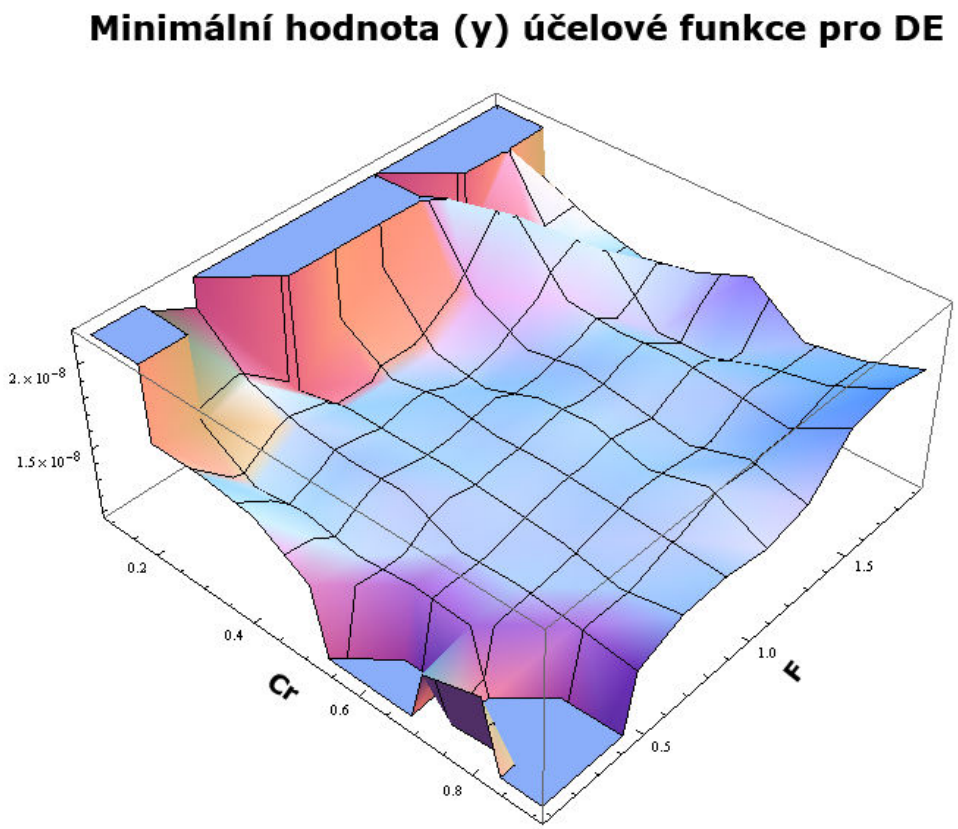
- D = 3
- NP = 25
- Generace = 400

Parametr *F* byl nastavován v intervalu hodnot 0,1 až 1,9 po kroku 0,2 a parametr *Cr* v intervalu hodnot 0,1 až 0,9 po kroku 0,1. Pro každou tuto kombinaci parametrů Diferenciální evoluce bylo vytvořeno 50 simulací. Celý soubor padesáti hodnot účelové funkce byl ukládán do souboru, ze kterého bylo potom vytvořeno pro každou kombinaci minimální, průměrná, maximální a medián hodnoty účelové funkce. Tyto výsledky byly vepsány do následujících tabulek. Každá tabulka byla ještě pro větší názornost doplněna 3D grafem.

Tabulka 11. Minimální hodnota účelové funkce pro 50 simulaci DE DERand1Bin

	Cr					
F		0,1	0,2	0,3	0,4	0,5
	0,1	$1{,}4539.10^{-7}$	$1{,}7236.10^{-8}$	$1{,}7269.10^{-8}$	$1{,}6827.10^{-8}$	$1{,}4831.10^{-8}$
	0,3	$2{,}1353.10^{-8}$	$1{,}6587.10^{-8}$	$1{,}4623.10^{-8}$	$1{,}5039.10^{-8}$	$1{,}4474.10^{-8}$
	0,5	$1{,}9972.10^{-8}$	$1{,}7290.10^{-8}$	$1{,}6428.10^{-8}$	$1{,}6174.10^{-8}$	$1{,}6301.10^{-8}$
	0,7	$1{,}8162.10^{-7}$	$1{,}4985.10^{-8}$	$1{,}5722.10^{-8}$	$1{,}5721.10^{-8}$	$1{,}5792.10^{-8}$
	0,9	$1{,}5218.10^{-7}$	$1{,}7654.10^{-8}$	$1{,}5710.10^{-8}$	$1{,}6353.10^{-8}$	$1{,}5911.10^{-8}$
	1,1	$1{,}5762.10^{-7}$	$1{,}9131.10^{-8}$	$1{,}5662.10^{-8}$	$1{,}5139.10^{-8}$	$1{,}6371.10^{-8}$
	1,3	$1{,}8160.10^{-8}$	$2{,}2066.10^{-8}$	$1{,}7444.10^{-8}$	$1{,}5827.10^{-8}$	$1{,}6436.10^{-8}$
	1,5	$1{,}5105.10^{-7}$	$1{,}7661.10^{-8}$	$2{,}0744.10^{-8}$	$1{,}7187.10^{-8}$	$1{,}6911.10^{-8}$
	1,7	$1{,}4611.10^{-7}$	$2{,}1998.10^{-8}$	$1{,}7242.10^{-8}$	$1{,}8615.10^{-8}$	$1{,}5286.10^{-8}$
	1,9	$2{,}0929.10^{-8}$	$1{,}8609.10^{-8}$	$1{,}7395.10^{-8}$	$1{,}6649.10^{-8}$	$1{,}7340.10^{-8}$

	Cr				
F		0,6	0,7	0,8	0,9
	0,1	$\mathbf{2{,}1013.10^{-11}}$	$1{,}3962.10^{-8}$	$1{,}5028.10^{-8}$	$1{,}5565.10^{-9}$
	0,3	$1{,}4831.10^{-8}$	$1{,}4634.10^{-8}$	$2{,}1712.10^{-9}$	$1{,}4538.10^{-9}$
	0,5	$1{,}4790.10^{-8}$	$1{,}4941.10^{-8}$	$1{,}4653.10^{-8}$	$1{,}4230.10^{-8}$
	0,7	$1{,}5567.10^{-8}$	$1{,}5321.10^{-8}$	$1{,}4653.10^{-8}$	$1{,}5327.10^{-8}$
	0,9	$1{,}6030.10^{-8}$	$1{,}5846.10^{-8}$	$1{,}5641.10^{-8}$	$1{,}5425.10^{-8}$
	1,1	$1{,}6631.10^{-8}$	$1{,}5754.10^{-8}$	$1{,}6669.10^{-8}$	$1{,}5005.10^{-8}$
	1,3	$1{,}5947.10^{-8}$	$1{,}5347.10^{-8}$	$1{,}6406.10^{-8}$	$1{,}5625.10^{-8}$
	1,5	$1{,}7145.10^{-8}$	$1{,}6855.10^{-8}$	$1{,}6411.10^{-8}$	$1{,}8213.10^{-8}$
	1,7	$1{,}5350.10^{-8}$	$1{,}6499.10^{-8}$	$1{,}7165.10^{-8}$	$1{,}8714.10^{-8}$
	1,9	$1{,}8662.10^{-8}$	$1{,}6329.10^{-8}$	$1{,}6902.10^{-8}$	$1{,}8116.10^{-8}$



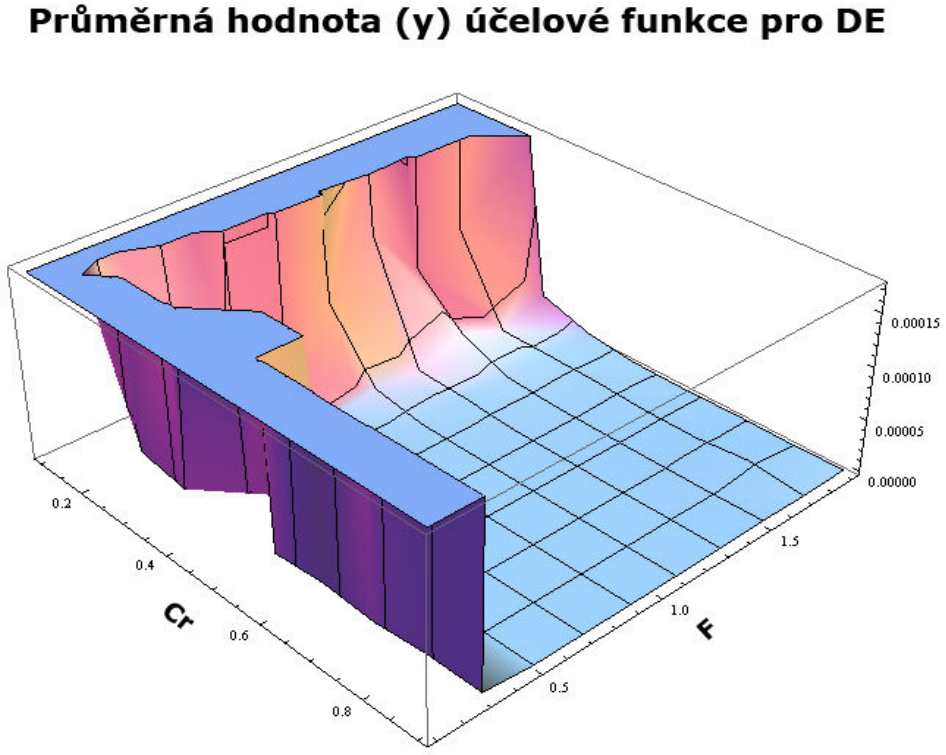
Obrázek 18. Zobrazení minimální hodnoty účelové funkce v 3D grafu

Nejnižší hodnotu ze souboru minimálních hodnot účelové funkce byla v předcházející tabulce hodnota $2{,}1013.10^{-11}$, což odpovídá kombinaci parametrů F=0,1 a Cr=0,6.

Tabulka 12. Průměrná hodnota účelové funkce pro 50 simulaci DE DERand1Bin

	Cr					
F		0,1	0,2	0,3	0,4	0,5
	0,1	$4{,}0080.10^{-4}$	$3{,}6052.10^{-3}$	$3{,}3337.10^{-2}$	$2{,}7865.10^{-3}$	$2{,}4156.10^{-2}$
	0,3	$2{,}7599.10^{-4}$	$9{,}4292.10^{-6}$	$1{,}6271.10^{-6}$	$1{,}2238.10^{-3}$	$2{,}3199.10^{-7}$
	0,5	$4{,}5987.10^{-4}$	$8{,}6425.10^{-6}$	$4{,}2430.10^{-7}$	$4{,}8715.10^{-7}$	$2{,}0841.10^{-7}$
	0,7	$1{,}3542.10^{-3}$	$1{,}4095.10^{-5}$	$1{,}4999.10^{-6}$	$3{,}3730.10^{-7}$	$1{,}5512.10^{-7}$
	0,9	$3{,}0267.10^{-3}$	$1{,}2898.10^{-5}$	$1{,}0882.10^{-6}$	$6{,}6097.10^{-7}$	$2{,}8025.10^{-7}$
	1,1	$7{,}8081.10^{-4}$	$7{,}0117.10^{-5}$	$9{,}6573.10^{-6}$	$1{,}3426.10^{-6}$	$7{,}3989.10^{-7}$
	1,3	$9{,}7531.10^{-4}$	$1{,}1702.10^{-4}$	$6{,}1527.10^{-6}$	$4{,}6605.10^{-6}$	$8{,}7219.10^{-7}$
	1,5	$6{,}4368.10^{-3}$	$6{,}5320.10^{-5}$	$2{,}3786.10^{-5}$	$4{,}2190.10^{-6}$	$1{,}2079.10^{-6}$
	1,7	$6{,}5168.10^{-3}$	$8{,}8077.10^{-5}$	$5{,}7876.10^{-6}$	$5{,}1155.10^{-6}$	$4{,}1747.10^{-6}$
	1,9	$2{,}5999.10^{-3}$	$4{,}0869.10^{-4}$	$2{,}2277.10^{-5}$	$7{,}3075.10^{-6}$	$1{,}2781.10^{-6}$

	Cr				
F		0,6	0,7	0,8	0,9
	0,1	$6{,}7646.10^{-3}$	$2{,}3145.10^{-2}$	$2{,}1108.10^{-2}$	$2{,}6985.10^{-2}$
	0,3	$3{,}7554.10^{-6}$	$1{,}4517.10^{-7}$	$4{,}5209.10^{-6}$	$4{,}4101.10^{-6}$
	0,5	$1{,}1058.10^{-7}$	$9{,}0280.10^{-8}$	$1{,}1109.10^{-7}$	$\mathbf{3{,}3079.10^{-8}}$
	0,7	$1{,}1487.10^{-7}$	$7{,}2790.10^{-8}$	$5{,}9534.10^{-8}$	$7{,}4918.10^{-8}$
	0,9	$1{,}6342.10^{-7}$	$1{,}1827.10^{-7}$	$1{,}1783.10^{-7}$	$1{,}2329.10^{-7}$
	1,1	$2{,}3678.10^{-7}$	$2{,}4058.10^{-7}$	$2{,}2079.10^{-7}$	$1{,}1928.10^{-7}$
	1,3	$5{,}9905.10^{-7}$	$3{,}1952.10^{-7}$	$5{,}8586.10^{-7}$	$3{,}4742.10^{-7}$
	1,5	$6{,}3700.10^{-7}$	$3{,}3879.10^{-7}$	$4{,}5195.10^{-7}$	$1{,}4039.10^{-6}$
	1,7	$1{,}4852.10^{-6}$	$1{,}2195.10^{-6}$	$6{,}2588.10^{-6}$	$7{,}2429.10^{-7}$
	1,9	$5{,}9966.10^{-7}$	$1{,}5125.10^{-6}$	$1{,}0178.10^{-6}$	$1{,}9879.10^{-6}$



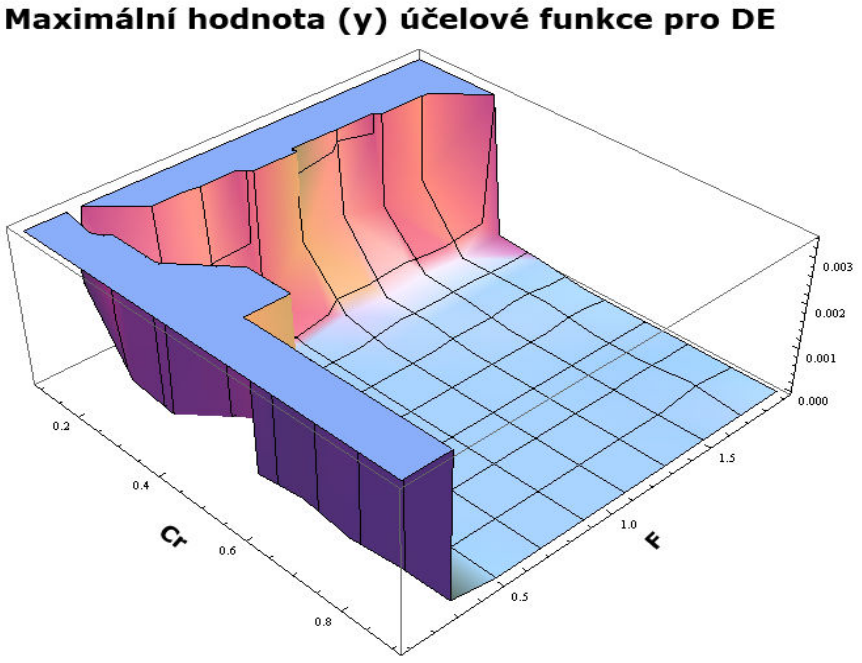
Obrázek 19. *Zobrazení průměrné hodnoty účelové funkce v 3D grafu*

Nejnižší hodnotou pro soubor průměrných hodnot účelové funkce byla hodnota $3{,}3079.10^{-8}$ a té odpovídají kombinace hodnot parametrů *F*=0,5 a *Cr*=0,9.

Tabulka 13. Maximální hodnota účelové funkce pro 50 simulaci DE DERand1Bin

	Cr					
F		0,1	0,2	0,3	0,4	0,5
	0,1	$1{,}3233.10^{-2}$	$1{,}7774.10^{-1}$	$9{,}7406.10^{-1}$	$1{,}3710.10^{-1}$	$7{,}4967.10^{-1}$
	0,3	$1{,}7582.10^{-3}$	$1{,}4433.10^{-4}$	$1{,}4483.10^{-5}$	$6{,}1162.10^{-2}$	$1{,}7647.10^{-6}$
	0,5	$1{,}2231.10^{-2}$	$1{,}5014.10^{-4}$	$2{,}1075.10^{-6}$	$1{,}4187.10^{-5}$	$1{,}5123.10^{-6}$
	0,7	$1{,}4601.10^{-2}$	$1{,}5240.10^{-4}$	$1{,}5285.10^{-5}$	$1{,}6978.10^{-6}$	$1{,}7710.10^{-6}$
	0,9	$8{,}1410.10^{-2}$	$1{,}9895.10^{-4}$	$1{,}4380.10^{-5}$	$1{,}4206.10^{-5}$	$1{,}5255.10^{-6}$
	1,1	$1{,}4563.10^{-2}$	$1{,}3596.10^{-3}$	$2{,}1014.10^{-4}$	$1{,}5306.10^{-5}$	$1{,}3214.10^{-5}$
	1,3	$1{,}6242.10^{-2}$	$1{,}5159.10^{-3}$	$1{,}4773.10^{-4}$	$1{,}5260.10^{-4}$	$1{,}8537.10^{-5}$
	1,5	$1{,}0470.10^{-1}$	$1{,}2993.10^{-3}$	$2{,}0902.10^{-4}$	$1{,}4475.10^{-4}$	$1{,}5344.10^{-5}$
	1,7	$1{,}0062.10^{-1}$	$1{,}4731.10^{-3}$	$1{,}2591.10^{-4}$	$1{,}5070.10^{-4}$	$1{,}6167.10^{-4}$
	1,9	$8{,}1827.10^{-2}$	$1{,}3865.10^{-2}$	$2{,}2018.10^{-4}$	$1{,}4405.10^{-4}$	$1{,}5020.10^{-5}$

	Cr				
F		0,6	0,7	0,8	0,9
	0,1	$3{,}2045.10^{-1}$	$6{,}6675.10^{-1}$	$6{,}8616.10^{-1}$	$8{,}3086.10^{-1}$
	0,3	$1{,}6676.10^{-4}$	$1{,}5461.10^{-6}$	$1{,}4056.10^{-4}$	$1{,}7559.10^{-4}$
	0,5	$1{,}4621.10^{-6}$	$1{,}8522.10^{-6}$	$1{,}7633.10^{-6}$	$1{,}6813.10^{-7}$
	0,7	$1{,}4666.10^{-6}$	**$1{,}6374.10^{-7}$**	$1{,}6893.10^{-7}$	$1{,}6937.10^{-6}$
	0,9	$1{,}7014.10^{-6}$	$1{,}3727.10^{-6}$	$1{,}4015.10^{-6}$	$1{,}5781.10^{-6}$
	1,1	$1{,}6415.10^{-6}$	$1{,}9601.10^{-6}$	$2{,}0163.10^{-6}$	$1{,}5447.10^{-6}$
	1,3	$1{,}4429.10^{-5}$	$1{,}4866.10^{-6}$	$1{,}3865.10^{-5}$	$1{,}6334.10^{-6}$
	1,5	$1{,}6861.10^{-5}$	$1{,}5813.10^{-6}$	$1{,}8620.10^{-6}$	$1{,}7160.10^{-5}$
	1,7	$1{,}8378.10^{-5}$	$1{,}6331.10^{-5}$	$1{,}4059.10^{-4}$	$1{,}7066.10^{-5}$
	1,9	$2{,}1004.10^{-6}$	$1{,}9344.10^{-5}$	$1{,}3797.10^{-5}$	$1{,}9441.10^{-5}$



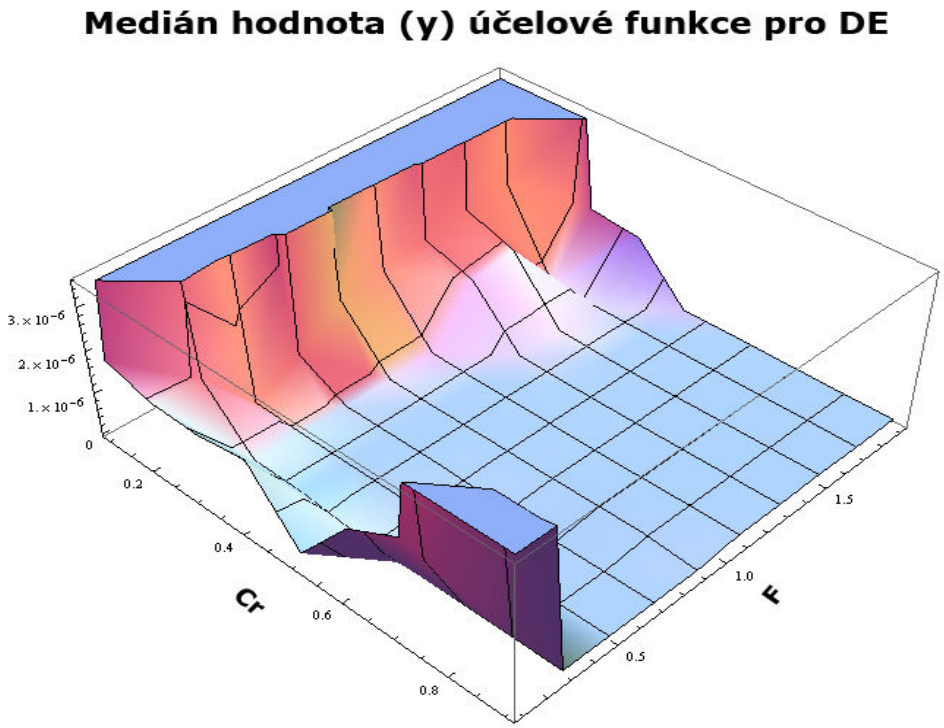
Obrázek 20. Zobrazení maximální hodnoty účelové funkce v 3D grafu

Nejnižší hodnotou pro soubor maximálních hodnot účelové funkce byla hodnota $1{,}6374.10^{-7}$ a té odpovídají kombinace hodnot parametrů F=0,7 a Cr=0,7.

Tabulka 14. Medián hodnot účelové funkce pro 50 simulaci DE DERand1Bin

	Cr					
F		0,1	0,2	0,3	0,4	0,5
	0,1	$1{,}8058.10^{-6}$	$1{,}4920.10^{-6}$	$1{,}4588.10^{-6}$	$1{,}6423.10^{-6}$	$1{,}8032.10^{-7}$
	0,3	$1{,}6861.10^{-5}$	$1{,}4166.10^{-6}$	$1{,}6847.10^{-7}$	$1{,}4818.10^{-7}$	$2{,}0228.10^{-8}$
	0,5	$1{,}9717.10^{-5}$	$1{,}9242.10^{-7}$	$1{,}5154.10^{-7}$	$1{,}4058.10^{-7}$	$1{,}3957.10^{-7}$
	0,7	$1{,}5166.10^{-4}$	$7{,}5496.10^{-7}$	$1{,}5753.10^{-7}$	$1{,}4707.10^{-7}$	$1{,}4232.10^{-7}$
	0,9	$2{,}0685.10^{-5}$	$1{,}4737.10^{-6}$	$1{,}8120.10^{-7}$	$1{,}5485.10^{-7}$	$1{,}5977.10^{-7}$
	1,1	$2{,}2119.10^{-5}$	$1{,}6060.10^{-6}$	$2{,}1852.10^{-7}$	$1{,}8388.10^{-7}$	$1{,}7310.10^{-7}$
	1,3	$7{,}0617.10^{-5}$	$1{,}7920.10^{-6}$	$1{,}4293.10^{-6}$	$1{,}9628.10^{-7}$	$1{,}6517.10^{-7}$
	1,5	$1{,}7951.10^{-4}$	$1{,}9740.10^{-6}$	$1{,}5923.10^{-6}$	$2{,}2041.10^{-7}$	$1{,}7628.10^{-7}$
	1,7	$1{,}5785.10^{-4}$	$2{,}1168.10^{-6}$	$2{,}1431.10^{-7}$	$2{,}0336.10^{-7}$	$1{,}9257.10^{-7}$
	1,9	$1{,}5091.10^{-4}$	$1{,}3180.10^{-5}$	$1{,}5064.10^{-6}$	$1{,}3898.10^{-6}$	$1{,}9233.10^{-7}$

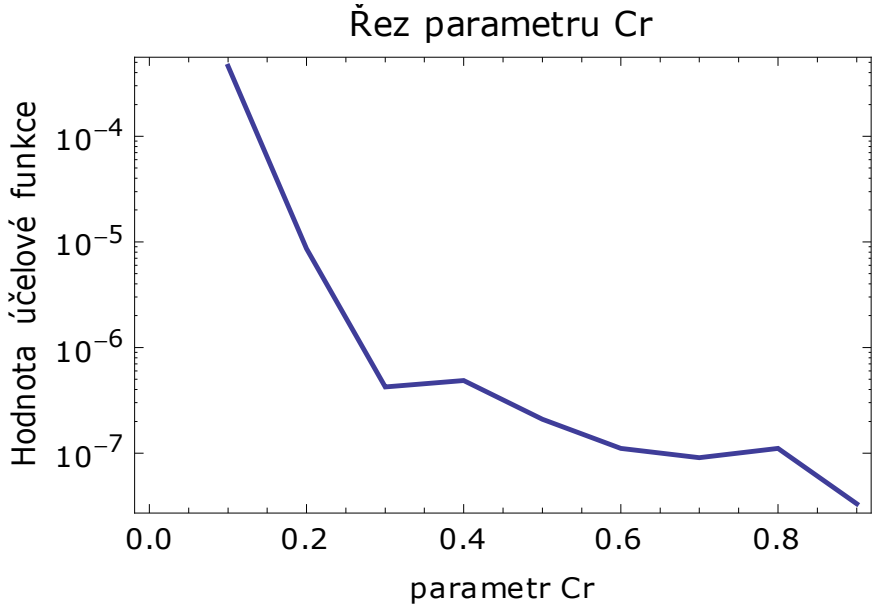
	Cr				
F		0,6	0,7	0,8	0,9
	0,1	$1{,}6077.10^{-6}$	$2{,}2454.10^{-6}$	$1{,}4160.10^{-5}$	$1{,}6683.10^{-5}$
	0,3	$1{,}7968.10^{-8}$	$1{,}9066.10^{-8}$	$1{,}9227.10^{-8}$	$1{,}9939.10^{-8}$
	0,5	$1{,}9921.10^{-8}$	$1{,}8020.10^{-8}$	$1{,}7570.10^{-8}$	$\mathbf{1{,}6556.10^{-8}}$
	0,7	$7{,}8968.10^{-8}$	$1{,}9473.10^{-8}$	$1{,}9832.10^{-8}$	$1{,}8739.10^{-8}$
	0,9	$1{,}4242.10^{-7}$	$1{,}4051.10^{-7}$	$1{,}3949.10^{-7}$	$2{,}0525.10^{-8}$
	1,1	$1{,}4813.10^{-7}$	$1{,}4120.10^{-7}$	$1{,}4295.10^{-7}$	$1{,}3697.10^{-7}$
	1,3	$1{,}4933.10^{-7}$	$1{,}4874.10^{-7}$	$1{,}4476.10^{-7}$	$1{,}4343.10^{-7}$
	1,5	$1{,}5622.10^{-7}$	$1{,}5912.10^{-7}$	$1{,}6130.10^{-7}$	$1{,}6610.10^{-7}$
	1,7	$1{,}7708.10^{-7}$	$1{,}6759.10^{-7}$	$1{,}6818.10^{-7}$	$1{,}5064.10^{-7}$
	1,9	$1{,}9196.10^{-7}$	$1{,}7782.10^{-7}$	$1{,}6449.10^{-7}$	$1{,}6883.10^{-7}$



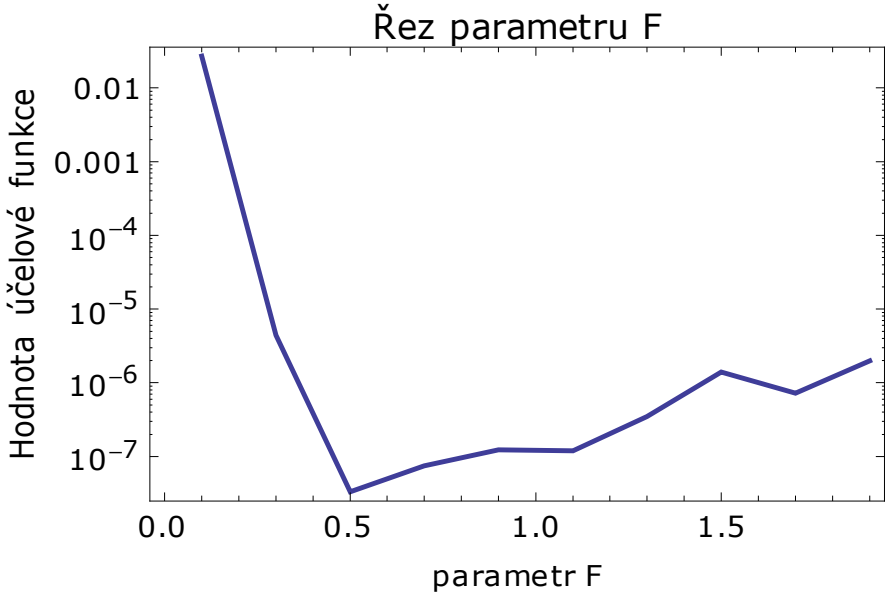
Obrázek 21. Zobrazení mediánu hodnoty účelové funkce v 3D grafu

Nejnižší hodnotu ze souboru mediánu hodnot účelové funkce byla hodnota $1{,}6556.10^{-8}$, což odpovídá kombinaci parametrů F=0,5 a Cr=0,9.

Parametry F a Cr vyšly naprosto shodně pro průměr a medián. U minimální hodnoty jde vidět, že tam jedna hodnota spadla příliš nízko. Hodnota účelové funkce je totiž při kombinaci parametrů F=0,1 *a* Cr=0,6 u ostatních souborů hodnot daleko větší. I z pohledu na 3D graf minimálních hodnot jde vidět, že plocha se značně odlišuje od zbylých grafů. Právě proto jsem se opět přiklonil stejně jako u parametrů algoritmu SOMA k hodnotám mediánu a průměru hodnot, které vyšly shodně pro kombinaci parametrů F=0,5 a Cr=0,9. Pro tuto kombinaci vybraných parametrů byly vytvořeny řezy, abych dokázal správnost své volby.



Obrázek 22. Řez parametru Cr



Obrázek 23. Řez parametru F

9 VÝSLEDKY SIMULACE

Po optimalizaci parametrů evolučních algoritmů, které byly provedeny v předcházející kapitole, je možno přejít k samotné optimalizaci parametrů pro metodu řízení chaosu. Evoluční algoritmy byly použity pro optimalizaci řízení dvou chaotických systémů. Jednorozměrné Logické rovnice a dvourozměrné Henonovy mapy. Oba systémy byly podrobeny vybraným UPOs. A to konkrétně p-1 fixnímu bodu a p-2 a p-4 příkladům pro vyšší orbit. Jako účelové funkce byly použity již výše popsané Targeting CF – CF Targ1 a CF Targ2. Dohromady tedy bylo vytvořeno dvanáct simulací, a každá z nich byla opakována 50 krát, pro vytvoření rozsáhlého podkladu pro statistické vyhodnocení. Tyto výsledky byly vloženy do tabulek a pro pochopení popisků tabulky byl vytvořen následující seznam zkratek:

- ***K*** - parametr pro ETDAS metodu řízení chaosu pro Logickou rovnici a pro Henonovu mapu. Jeho hodnota se hledá v rozsahu <-2, 2>
- $\boldsymbol{F_{max}}$ - parametr pro ETDAS metodu řízení chaosu pro Logickou rovnici a pro Henonovu mapu. Jeho hodnota se hledá v rozsahu <0,01, 0,9>
- ***R*** - parametr pro ETDAS metodu řízení chaosu pro Logickou rovnici a pro Henonovu mapu. Jeho hodnota se hledá v rozsahu <0.01, 0.9>
- ***Min CF Val*** - minimální hodnota účelové funkce
- ***Avg CF Val*** - průměrná hodnota účelové funkce
- ***Med CF Val*** - medián hodnota účelové funkce

Aby se mohlo pokračovat v popisování zkratek v tabulkách, musí se ještě vysvětlit funkce *akceptovatelné hodnoty*, která v algoritmu hledá první vyhovující jedince. To je podobné omezení, jak parametr *MinDiv* u algoritmu SOMA, akorát v tomto případě nezajistí ukončení cyklu algoritmu, ale nechá ho dopočítat až do konce všech padesáti cyklů. S touto *akceptovatelnou hodnotou* se porovnávají hodnoty účelové funkce. Pokud je hodnota menší, než *akceptovatelná hodnota*, algoritmus pokračuje dál, ale hodnota jedince se uloží do souboru Stop. Jsou tam tedy uloženi všichni jedinci, jejichž hodnota účelové funkce byla nižší než *akceptovatelná hodnota*. Hodnota této proměnné je ovšem pro každý případ jiná, jak to ukazuje následující *tabulka 15*. Hodnoty v tabulce jsou založeny na předcházejících zkušenostech při aplikaci EA na optimalizaci řízení chaosu. [9]

Tabulka 15. Akceptovatelná hodnota pro všechny typy chaotických systémů

Logická rovnice	1-p CF1	2-p CF1	4-p CF1	1-p CF2	2-p CF2	4-p CF2
hodnota omezení	10^{-14}	0,01	0,1	10^{-14}	0,0001	0,001
Henonova mapa	1-p CF1	2-p CF1	4-p CF1	1-p CF2	2-p CF2	4-p CF2
hodnota omezení	10^{-14}	0,1	0,01	10^{-14}	0,001	0,0001

- ***Avg CFE*** - Průměrný počet ohodnocení účelové funkce, než došlo k nalezení prvního dobrého jedince, tedy hodnoty účelové funkce nižší než je stanovená *akceptovatelná hodnota*. Tato hodnota se ukládá do souboru s koncovkou stop
- ***Min CF stop*** - minimální hodnota účelové funkce, která byla uložena do souboru s koncovkou stop, tedy nejmenší hodnota účelové funkce z prvních vyhovujících jedinců.
- ***Avg CF stop*** - průměrná hodnota účelové funkce, které byla vypočítána z jedinců uložených do souboru s koncovkou stop, tedy průměr hodnot účelové funkce z prvních vyhovujících jedinců.
- ***%*** - procentuální vyjádření, kolik z těch 50 cyklů vyhovovalo podmínce, že hodnota účelové funkce je menší, než *akceptovatelná hodnota* resp. jaké procento hodnot je uloženo v souboru s koncovkou stop

Kromě souboru s koncovkou stop program ukládal hodnoty do souboru s koncovkou full. Už podle názvu můžeme předvídat, že se zde ukládaly nejlepší hodnoty z posledních populací ze všech padesáti cyklů bez ohledu na *akceptovatelnou hodnotu*.

9.1 Logická rovnice

Logická rovnice je jednodimenzionální diskrétní chaotický systém, na kterém byla testována výkonnost evolučních algoritmů. Pro každou kombinaci použité účelové funkce a žádaného stavu (UPO) byl vytvořen program v prostředí *Mathematica 7.0* který pro algoritmus SOMA, Diferenciální evoluci a chaotickou Diferenciální evoluci prováděl testování na účelových funkcích CF Targ1 a CF Targ2

9.1.1 CF Targ1

Výsledky, které byly získány pomocí evolučních algoritmů, byly vyneseny do následujících tabulek. Každý evoluční algoritmus, jehož strategii reprezentuje číslo popsané v *tabulka 4, tabulka 5* a *tabulka 6*, pracoval pro p-1 orbit, p-2 a p-4 orbit.

p-1 orbit

Tabulka 16. Nejlepší řešení řízení logické rovnice, p-1 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	**-1,1425**	**0,5353**	**0,6345**	**$1{,}6.10^{-15}$**	**$1{,}762.10^{-15}$**
2	-1,1586	0,5488	0,6436	$1{,}7.10^{-15}$	$1{,}894.10^{-15}$
3	-1,1486	0,5369	0,6380	$1{,}7.10^{-15}$	$1{,}908.10^{-15}$
4	-1,1568	0,5485	0,6427	$1{,}6.10^{-15}$	$1{,}868.10^{-15}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	**$1{,}7.10^{-15}$**	**180,46**	**$2{,}1.10^{-15}$**	**$5{,}7544.10^{-15}$**	**100**
2	$1{,}9.10^{-15}$	324,64	$2{,}4.10^{-15}$	$5{,}726.10^{-15}$	100
3	$1{,}9.10^{-15}$	553,58	$2{,}1.10^{-15}$	$5{,}411.10^{-15}$	100
4	$1{,}8.10^{-15}$	287,18	$2{,}8.10^{-15}$	$5{,}5261.10^{-15}$	100

Tabulka 17. Nejlepší řešení řízení logické rovnice, p-1 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	**-1,1553**	**0,5349**	**0,6418**	**$1{,}4.10^{-15}$**	**$1{,}416.10^{-15}$**
6	-1,1574	0,5328	0,6430	$1{,}4.10^{-15}$	$1{,}46.10^{-15}$
7	-1,1565	0,5318	0,6425	$1{,}4.10^{-15}$	$1{,}448.10^{-15}$
8	-1,1574	0,5355	0,6430	$1{,}4.10^{-15}$	$1{,}534.10^{-15}$
9	-1,1569	0,5350	0,6427	$1{,}5.10^{-15}$	$1{,}56.10^{-15}$
10	-1,1557	0,5337	0,6421	$1{,}5.10^{-15}$	$1{,}574.10^{-15}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	**$1{,}4.10^{-15}$**	**57,88**	**$2{,}5.10^{-15}$**	**$5{,}0417.10^{-15}$**	**100**
6	$1{,}5.10^{-15}$	46,18	$2{,}4.10^{-15}$	$5{,}1281.10^{-15}$	100
7	$1{,}4.10^{-15}$	40,04	$2{,}3.10^{-15}$	$5{,}5993.10^{-15}$	100
8	$1{,}5.10^{-15}$	53,34	$2{,}2.10^{-15}$	$5{,}5497.10^{-15}$	100
9	$1{,}6.10^{-15}$	50,86	$2{,}4.10^{-15}$	$5{,}6370.10^{-15}$	100
10	$1{,}6.10^{-15}$	57,8	$2{,}4.10^{-15}$	$5{,}3461.10^{-15}$	100

Tabulka 18. Nejlepší řešení řízení logické rovnice p-1orbit, CF Targ1,Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	**-1,1572**	**0,5296**	**0,6429**	**$1{,}4.10^{-15}$**	**$1{,}518.10^{-15}$**
12	-1,157	0,5355	0,6428	$1{,}4.10^{-15}$	$1{,}484.10^{-15}$
13	-1,1564	0,5348	0,6424	$1{,}4.10^{-15}$	$1{,}494.10^{-15}$
14	-1,1577	0,5337	0,6432	$1{,}8.10^{-15}$	$2{,}216.10^{-15}$
15	-1,1577	0,5331	0,6431	$1{,}5.10^{-15}$	$1{,}58.10^{-15}$
16	-1,1579	0,5339	0,6433	$1{,}5.10^{-15}$	$1{,}61.10^{-15}$

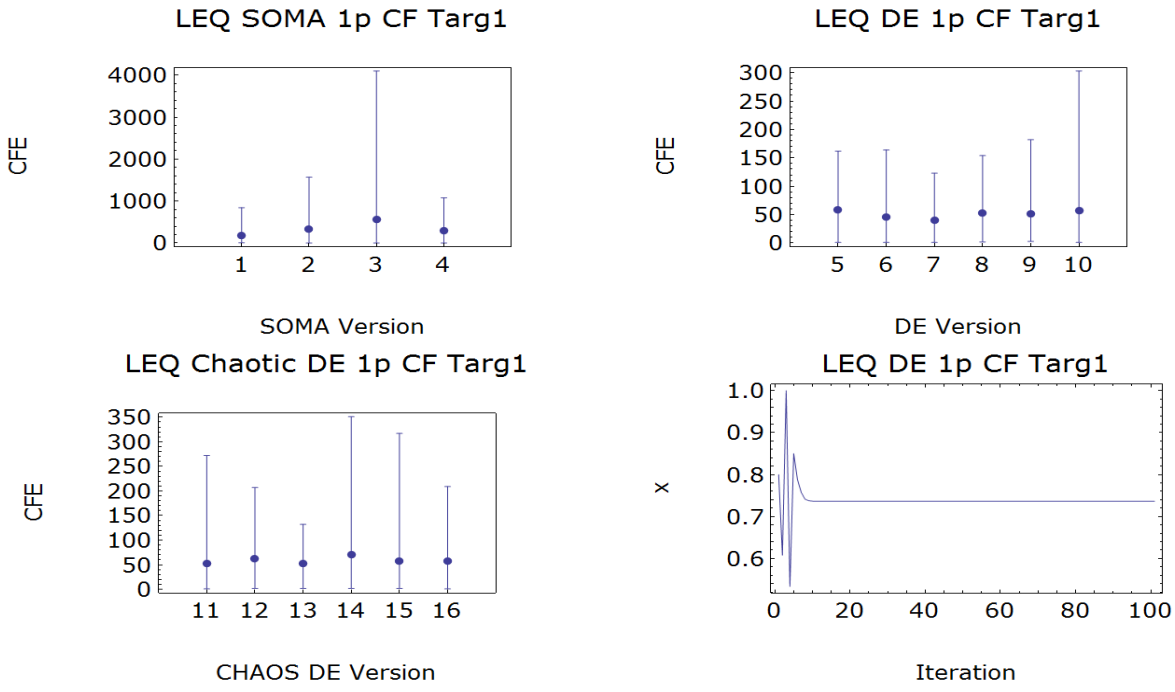
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	**$1{,}5.10^{-15}$**	**53,06**	**$2{,}8.10^{-15}$**	**$5{,}6369.10^{-15}$**	**100**
12	$1{,}5.10^{-15}$	62,04	$2{,}3.10^{-15}$	$5{,}2020.10^{-15}$	100
13	$1{,}5.10^{-15}$	52,32	$2{,}8.10^{-15}$	$5{,}0840.10^{-15}$	100
14	$2{,}2.10^{-15}$	70,76	$2{,}7.10^{-15}$	$5{,}4473.10^{-15}$	100
15	$1{,}6.10^{-15}$	57,24	$2{,}9.10^{-15}$	$5{,}2554.10^{-15}$	100
16	$1{,}6.10^{-15}$	57,3	$3{,}1.10^{-15}$	$5{,}6281.10^{-15}$	100

Zvýrazněný řádek strategie AllToOne algoritmu SOMA v *tabulce 16* ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}6.10^{-15}$ pro nastavení Logické rovnice, p-1 orbit, CF Targ1.

Zvýrazněný řádek varianty DERand1Bin Diferenciální evoluce v *tabulka 17* ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}4.10^{-15}$ pro nastavení Logické rovnice, p-1 orbit, CF Targ1.

Zvýrazněný řádek varianty DERand1Bin Chaotické Diferenciální evoluce v *tabulce 18* ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}4.10^{-15}$ pro nastavení Logické rovnice, p-1 orbit, CF Targ1.

Pro řízení stabilizace Logické rovnice, p-1 orbit, CF Targ1 se jako nejlepší evoluční algoritmus pro řízení ukazovala Diferenciální evoluce ve variantě DERand1Bin. Měla sice minimální hodnotu účelové funkce stejnou jako Chaotická Diferenciální evoluce, nicméně průměrnou hodnotu účelové funkce měla klasická Diferenciální evoluce o jednu desetinu nižší. Proto byla zvolena právě tato varianta pro zobrazení simulace stabilizace chaotického systému na *obrázku 24* dole vpravo



Obrázek 24. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro logickou rovnici, p-1 orbit, CF Targ1

p-2 orbit

Tabulka 19. Nejlepší řešení logické rovnice, p-2 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	0,5560	0,2123	0,5081	$7{,}1624.10^{-13}$	$3{,}7098.10^{-3}$
2	0,5470	0,0679	0,4722	$8{,}7897.10^{-13}$	$1{,}8936.10^{-2}$
3	0,5571	0,0974	0,5080	$1{,}5156.10^{-12}$	$4{,}8295.10^{-3}$
4	**0,5435**	**0,0343**	**0,4813**	$\mathbf{5{,}7915.10^{-13}}$	$\mathbf{1{,}6863.10^{-2}}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$2{,}0988.10^{-4}$	14518,3	$7{,}1624.10^{-13}$	0,0044	94
2	$3{,}4571.10^{-4}$	16193,7	$1{,}3098.10^{-8}$	0,0034	82
3	$6{,}0557.10^{-5}$	16354,66	$1{,}5156.10^{-12}$	0,0029	90
4	$\mathbf{3{,}7346.10^{-4}}$	**15057,9**	$\mathbf{1{,}253.10^{-5}}$	**0,0048**	**88**

Tabulka 20. Nejlepší řešení logické rovnice, p-2 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	0,4404	0,1999	0,2625	$2{,}3126.10^{-13}$	$9{,}1311.10^{-2}$
6	0,5487	0,0876	0,4837	$5{,}4144.10^{-13}$	$2{,}1346.10^{-6}$
7	**0,5286**	**0,0422**	**0,4511**	$\mathbf{2{,}1537.10^{-13}}$	$\mathbf{1{,}0772.10^{-1}}$
8	0,5284	0,0423	0,4518	$4{,}0045.10^{-13}$	$1{,}1348.10^{-5}$
9	0,5240	0,1376	0,4464	$6{,}1918.10^{-13}$	$2{,}3805.10^{-7}$
10	0,5496	0,0363	0,4747	$3{,}8397.10^{-13}$	$1{,}2875.10^{-7}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$1{,}0074.10^{-5}$	2878,26	$1{,}0465.10^{-5}$	0,0045	98
6	$4{,}8887.10^{-9}$	3561,78	$6{,}6534.10^{-7}$	0,0032	100
7	$\mathbf{7{,}5219.10^{-13}}$	**3147,28**	$\mathbf{1{,}6758.10^{-12}}$	**0,0041**	**98**
8	$6{,}2647.10^{-9}$	1924,38	$5{,}2043.10^{-12}$	0,0036	100
9	$4{,}4234.10^{-9}$	3509,12	$1{,}0502.10^{-12}$	0,0032	100
10	$6{,}6913.10^{-13}$	2642,76	$1{,}0999.10^{-12}$	0,0032	100

Tabulka 21. Nejlepší řešení logické rovnice, p-2 orbit, CF Targ1, Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	0,5626	0,0335	0,4912	$9{,}5114.10^{-13}$	$2{,}1378.10^{-3}$
12	0,5550	0,2143	0,4841	$4{,}0990.10^{-13}$	$1{,}2434.10^{-6}$
13	**0,5383**	**0,0413**	**0,4654**	$\mathbf{2{,}2949.10^{-13}}$	$\mathbf{8{,}6714.10^{-2}}$
14	0,5532	0,0885	0,4941	$8{,}4517.10^{-13}$	$5{,}4230.10^{-4}$
15	0,5596	0,0321	0,4918	$5{,}1094.10^{-13}$	$1{,}0665.10^{-6}$
16	0,5461	0,1499	0,4839	$4{,}2540.10^{-13}$	$1{,}0778.10^{-6}$

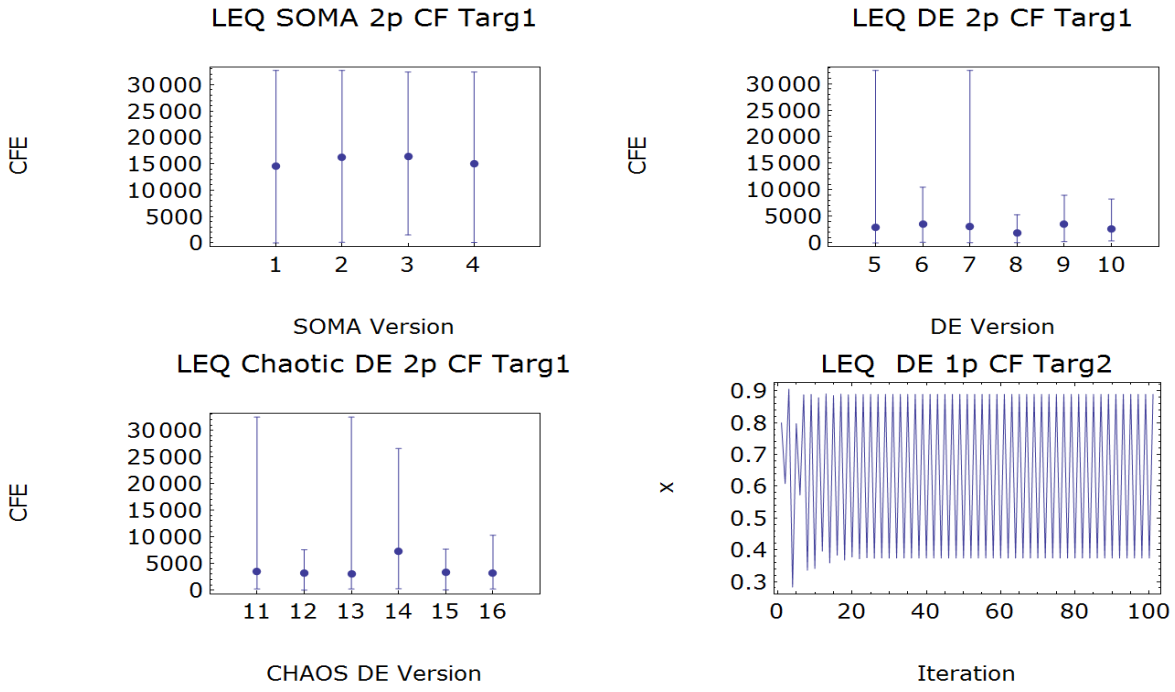
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$1{,}0989.10^{-4}$	3568,02	$9{,}5114.10^{-13}$	$3{,}6483.10^{-3}$	96
12	$2{,}2441.10^{-8}$	3223,9	$9{,}2214.10^{-13}$	$3{,}1392.10^{-3}$	100
13	$\mathbf{9{,}1156.10^{-13}}$	**3062,44**	$\mathbf{1{,}9205.10^{-9}}$	$\mathbf{2{,}6531.10^{-3}}$	**98**
14	$6{,}0546.10^{-5}$	7255,44	$1{,}6307.10^{-12}$	$2{,}4939.10^{-3}$	100
15	$5{,}0140.10^{-9}$	3420,62	$5{,}1094.10^{-13}$	$2{,}6766.10^{-3}$	100
16	$7{,}9321.10^{-13}$	3144,72	$1{,}8012.10^{-8}$	$2{,}6715.10^{-3}$	100

Nejlepší strategií algoritmu SOMA pro nastavení logické rovnice, p-2 orbit, CF Targ1 je AllToAllAdaptive. Minimální hodnota účelové funkce je *Min CF Val* = 5,7915.10^{-13}. To ukazuje *tabulka 19*.

Zvýrazněný řádek varianta DEBest2Bin Diferenciální evoluce ukazuje v *tabulce 20* minimální hodnotu účelové funkce *Min CF Val* = 1,6758.10^{-12} pro nastavení logické rovnice, p-2 orbit, CF Targ1.

Nejlepší variantou Chaotické Diferenciální evoluce pro nastavení logické rovnice, p-2 orbit, CF Targ1 je DEBest2Bin. Minimální hodnota účelové funkce je *Min CF Val* = 2,2949.10^{-13}. To ukazuje *tabulka 21*.

Pro nastavení simulace Logické rovnice, p-2 orbit, CF Targ1 se jako nejlepší evoluční algoritmus pro řízení ukazovala Diferenciální evoluce ve variantě DEBest2Bin. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 25* dole vpravo



Obrázek 25. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro logickou rovnici, p-2 orbit, CF Targ1

4-p orbit

Tabulka 22. Nejlepší řešení logické rovnice, p-4 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	-0,5223	0,3087	0,6121	$5{,}7356.10^{-3}$	1,3939
2	**-0,5222**	**0,3039**	**0,6121**	$\mathbf{5{,}7108.10^{-3}}$	**1,5609**
3	-0,5237	0,3145	0,6129	$6{,}1381.10^{-3}$	1,3961
4	-0,5232	0,1795	0,6141	$5{,}8947.10^{-3}$	1,5239

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$5{,}3926.10^{-1}$	27722	$7{,}7181.10^{-3}$	$4{,}2661.10^{-2}$	28
2	$\mathbf{9{,}1424.10^{-1}}$	**29934,94**	$\mathbf{6{,}5754.10^{-3}}$	$\mathbf{3{,}6847.10^{-2}}$	**18**
3	$5{,}9670.10^{-1}$	29218,6	$6{,}1829.10^{-3}$	$3{,}6956.10^{-2}$	28
4	1,1547	28445,9	$6{,}6158.10^{-3}$	$4{,}7338.10^{-2}$	24

Tabulka 23. Nejlepší řešení logické rovnice, p-4 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	-0,5229	0,1694	0,6138	$4{,}0608.10^{-3}$	1,6173
6	-0,5227	0,1128	0,6141	$5{,}4822.10^{-3}$	$5{,}9500.10^{-1}$
7	-0,5212	0,1698	0,6101	$4{,}2244.10^{-3}$	1,2666
8	**-0,4220**	**0,2136**	**0,3903**	$\mathbf{2{,}3220.10^{-3}}$	$\mathbf{8{,}0397.10^{-1}}$
9	-0,5178	0,7272	0,5979	$5{,}8864.10^{-3}$	$7{,}2142.10^{-1}$
10	-0,4210	0,3384	0,3915	$2{,}4618.10^{-3}$	$9{,}6076.10^{-1}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$4{,}7802.10^{-2}$	19834,72	$6{,}1159.10^{-3}$	$4{,}1859.10^{-2}$	56
6	$4{,}4176.10^{-2}$	22703,54	$6{,}4014.10^{-3}$	$3{,}6603.10^{-2}$	62
7	$5{,}6748.10^{-2}$	16775,74	$6{,}1717.10^{-3}$	$3{,}3455.10^{-2}$	62
8	$\mathbf{5{,}7095.10^{-3}}$	**12190,06**	$\mathbf{5{,}8146.10^{-3}}$	$\mathbf{4{,}2440.10^{-2}}$	**80**
9	$1{,}4487.10^{-1}$	25155,42	$6{,}0938.10^{-3}$	$3{,}3707.10^{-2}$	44
10	$5{,}8961.10^{-3}$	20041,28	$6{,}0132.10^{-3}$	$4{,}7331.10^{-2}$	64

Tabulka 24. Nejlepší řešení logické rovnice, p-4 orbit, CF Targ1, Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	-0,5227	0,1690	0,6133	$4{,}0540.10^{-3}$	1,8535
12	-0,5201	0,1972	0,6086	$5{,}5587.10^{-3}$	$3{,}2901.10^{-1}$
13	**-0,5210**	**0,1700**	**0,6039**	$\mathbf{3{,}9320.10^{-3}}$	$\mathbf{7{,}8891.10^{-1}}$
14	-0,5219	0,3925	0,6120	$5{,}9849.10^{-3}$	$5{,}0476.10^{-1}$
15	-0,5194	0,8438	0,5990	$5{,}9539.10^{-3}$	$6{,}0416.10^{-1}$
16	-0,5247	0,1693	0,6151	$4{,}2468.10^{-3}$	$3{,}6650.10^{-1}$

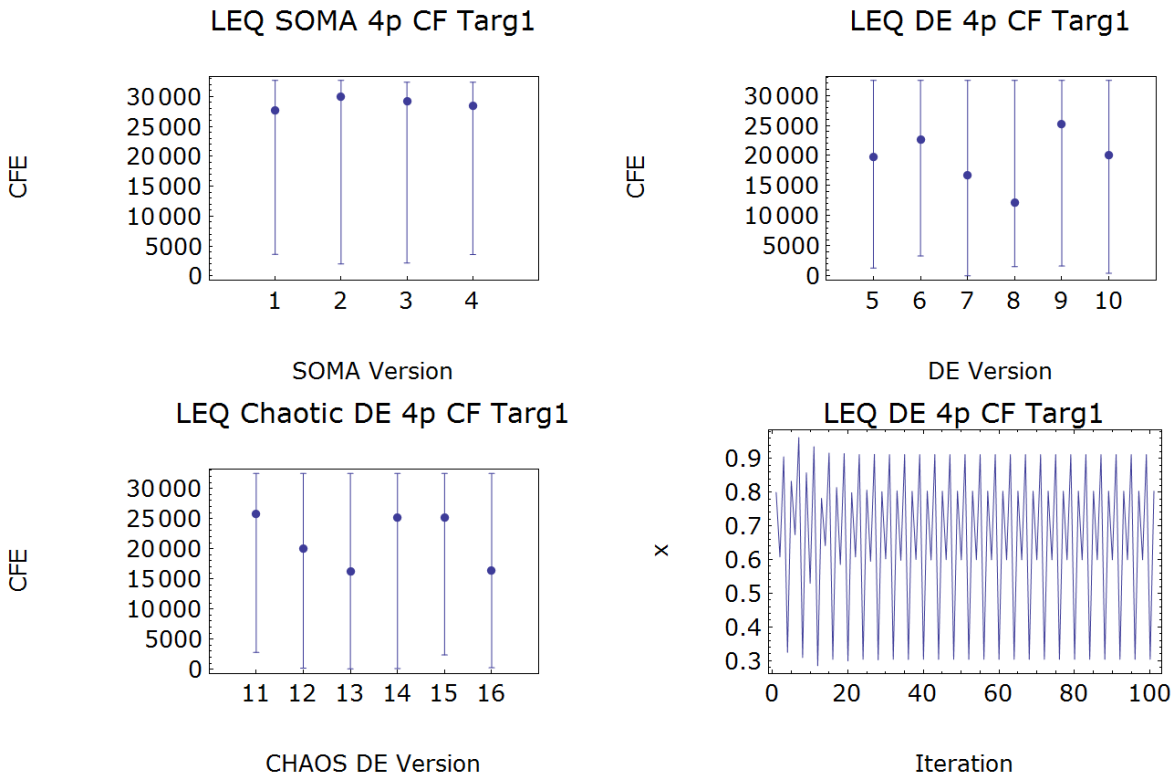
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$8{,}9533.10^{-1}$	25757,36	$9{,}7488.10^{-3}$	$5{,}6491.10^{-2}$	26
12	$1{,}1034.10^{-2}$	20080,42	$5{,}7992.10^{-3}$	$3{,}1660.10^{-2}$	76
13	$\mathbf{5{,}7055.10^{-3}}$	**16230,98**	$\mathbf{5{,}5530.10^{-3}}$	$\mathbf{2{,}9727.10^{-2}}$	**72**
14	$2{,}4162.10^{-1}$	25158,22	$5{,}9849.10^{-3}$	$3{,}3554.10^{-2}$	38
15	$1{,}2081.10^{-1}$	25180,66	$6{,}2755.10^{-3}$	$3{,}6170.10^{-2}$	44
16	$6{,}1746.10^{-3}$	16355,7	$5{,}8233.10^{-3}$	$3{,}3888.10^{-2}$	80

Zvýrazněný řádek strategie AllToRandom algoritmu SOMA ukazuje v *tabulce 22* minimální hodnotu účelové funkce *Min CF Val* = 5,7108.10^{-3} pro nastavení Logické rovnice, p-4 orbit, CF Targ1.

Nejlepší variantou Diferenciální evoluce pro nastavení logické rovnice, p-4 orbit, CF Targ1 je DELocalToBest. Minimální hodnota účelové funkce je *Min CF Val*= 2,3220.10^{-3}. To ukazuje *tabulka 23*.

Zvýrazněný řádek varianta DEBest2Bin Chaotické Diferenciální evoluce v *tabulce 24* ukazuje minimální hodnotu účelové funkce *Min CF Val* = 3,9320.10^{-3} pro nastavení Logické rovnice, p-4 orbit, CF Targ1.

Pro nastavení simulace Logické rovnice, p-4 orbit, CF Targ1 se jako nejlepší evoluční algoritmus pro řízení Chaotického systému ukazovala opět Diferenciální evoluce ve variantě DELocalToBest. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 26* dole vpravo



Obrázek 26. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro logickou rovnici, p-4 orbit, CF Targ1

9.1.2 CF Targ2

Stejně jako u předchozí účelové funkce byly vyhodnoceny výsledky z optimalizace pro simulaci řízení chaotických systému a výsledky byly zobrazeny v následujících tabulkách. Každý evoluční algoritmus pracoval pro p-1 orbit, p-2 a p-4 orbit.

p-1 orbit

Tabulka 25. Nejlepší řešení logické rovnice, p-1 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	**-1,1540**	**0,5352**	**0,6411**	**$1{,}5.10^{-15}$**	**$1{,}726.10^{-15}$**
2	-1,1500	0,5332	0,6387	$1{,}7.10^{-15}$	$1{,}86.10^{-15}$
3	-1,1433	0,6583	0,6351	$1{,}7.10^{-15}$	$1{,}8484.10^{-15}$
4	-1,1492	0,5587	0,6384	$1{,}6.10^{-15}$	$1{,}7904.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	**$1{,}7.10^{-15}$**	**14**	**$3{,}7.10^{-15}$**	**$6{,}7159.10^{-15}$**	**100**
2	$1{,}9.10^{-15}$	18,28	$4{,}0102.10^{-15}$	$7{,}1751.10^{-15}$	100
3	$1{,}8.10^{-15}$	97,64	$3{,}9102.10^{-15}$	$7{,}1597.10^{-15}$	100
4	$1{,}8.10^{-15}$	57,92	$3{,}2.10^{-15}$	$6{,}6568.10^{-15}$	100

Nejlepší verzí algoritmu SOMA pro nastavení logické rovnice, p-1 orbit, CF Targ2 je AllToOne. Minimální hodnota účelové funkce je *Min CF Val* = $1{,}5.10^{-15}$.

Tabulka 26. Nejlepší řešení logické rovnice, p-1 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	**-1,1575**	**0,5327**	**0,6430**	**$1{,}4.10^{-15}$**	**$1{,}436.10^{-15}$**
6	-1,1568	0,5344	0,6427	$1{,}4.10^{-15}$	$1{,}47.10^{-15}$
7	-1,1574	0,5343	0,6430	$1{,}4.10^{-15}$	$1{,}456.10^{-15}$
8	-1,1567	0,5353	0,6426	$1{,}4.10^{-15}$	$1{,}484.10^{-15}$
9	-1,1572	0,5322	0,6429	$1{,}5.10^{-15}$	$1{,}54.10^{-15}$
10	-1,1580	0,5338	0,6433	$1{,}5.10^{-15}$	$1{,}562.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	**$1{,}4.10^{-15}$**	**12,66**	**$3{,}5.10^{-15}$**	**$6{,}9601.10^{-15}$**	**100**
6	$1{,}5.10^{-15}$	11,14	3.10^{-15}	$6{,}7489.10^{-15}$	100
7	$1{,}4.10^{-15}$	14,3	3.10^{-15}	$7{,}3254.10^{-15}$	100
8	$1{,}5.10^{-15}$	11,14	$2{,}9.10^{-15}$	$6{,}8578.10^{-15}$	100
9	$1{,}5.10^{-15}$	11,36	$3{,}7102.10^{-15}$	$6{,}4722.10^{-15}$	100
10	$1{,}6.10^{-15}$	16,96	$2{,}8.10^{-15}$	$6{,}7309.10^{-15}$	100

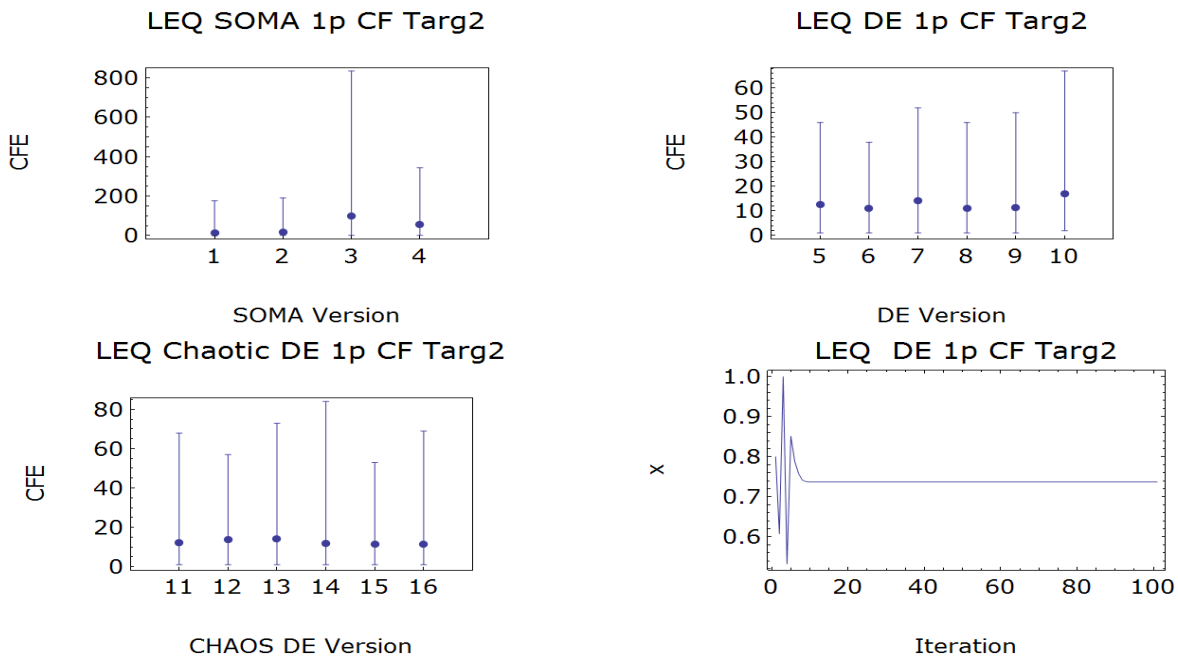
Zvýrazněný řádek varianty DERand1Bin Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}4.10^{-15}$ pro nastavení Logické rovnice, p-1 orbit, CF Targ2.

Tabulka 27. Nejlepší řešení logické rovnice, p-1 orbit, CF Targ1,Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	**-1,1545**	**0,5336**	**0,6414**	**$1{,}4.10^{-15}$**	**$1{,}468.10^{-15}$**
12	-1,1560	0,5345	0,6422	$1{,}4.10^{-15}$	$1{,}488.10^{-15}$
13	-1,1573	0,5338	0,6429	$1{,}4.10^{-15}$	$1{,}464.10^{-15}$
14	-1,1572	0,5345	0,6429	$1{,}8.10^{-15}$	$2{,}2564.10^{-15}$
15	-1,1568	0,5347	0,6427	$1{,}5.10^{-15}$	$1{,}578.10^{-15}$
16	-1,1571	0,5355	0,6428	$1{,}4.10^{-15}$	$1{,}586.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	**$1{,}5.10^{-15}$**	**12,1**	**$3{,}8102.10^{-15}$**	**$6{,}8334.10^{-15}$**	**100**
12	$1{,}5.10^{-15}$	13,92	$3{,}8102.10^{-15}$	$6{,}5166.10^{-15}$	100
13	$1{,}5.10^{-15}$	14,06	$2{,}7.10^{-15}$	$6{,}8009.10^{-15}$	100
14	$2{,}25.10^{-15}$	11,82	$3{,}5.10^{-15}$	$7{,}1526.10^{-15}$	100
15	$1{,}6.10^{-15}$	11,36	$3{,}2.10^{-15}$	$6{,}6698.10^{-15}$	100
16	$1{,}6.10^{-15}$	11,42	$3{,}4992.10^{-15}$	$6{,}4753.10^{-15}$	100

Nejlepší variantou Chaotické Diferenciální evoluce pro nastavení logické rovnice, p-1 orbit, CF Targ2 je DERand1Bin. Minimální hodnota účelové funkce je *Min CF Val* = $1{,}4.10^{-15}$.

Pro nastavení simulace Logické rovnice, p-1 orbit, CF Targ2 se jako nejlepší evoluční algoritmus pro řízení ukazovala Diferenciální evoluce ve variantě DERand1Bin. Tato varianta byla zvolena pro graf simulace stabilizace chaotického systému na *obrázku 27* dole vpravo.



Obrázek 27. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro logickou rovnici, p-1 orbit, CF Targ2

p-2 orbit

Tabulka 28. Nejlepší řešení logické rovnice, p-2 orbit, CF Targ2, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	0,5525	0,0868	0,4939	$2{,}0172.10^{-14}$	$1{,}9026.10^{-5}$
2	0,5533	0,0661	0,5016	$1{,}8395.10^{-14}$	$3{,}0771.10^{-5}$
3	**0,5475**	**0,1375**	**0,4887**	$\mathbf{1{,}7685.10^{-14}}$	$\mathbf{2{,}3422.10^{-5}}$
4	0,5327	0,1806	0,4582	$1{,}8662.10^{-14}$	$2{,}6637.10^{-5}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$1{,}5385.10^{-6}$	16687,4	$1{,}7911.10^{-8}$	$1{,}3966.10^{-5}$	90
2	$2{,}0377.10^{-6}$	20398,8	$1{,}8395.10^{-14}$	$1{,}1464.10^{-5}$	88
3	$\mathbf{1{,}6984.10^{-6}}$	**17798,9**	$\mathbf{1{,}9728.10^{-14}}$	$\mathbf{1{,}0366.10^{-5}}$	**88**
4	$1{,}4596.10^{-6}$	16548,9	$1{,}5439.10^{-8}$	$1{,}4918.10^{-5}$	86

Tabulka 29. Nejlepší řešení logické rovnice, p-2 orbit, CF Targ2, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	0,5611	0,0804	0,4977	$1{,}6952.10^{-14}$	$6{,}5909.10^{-5}$
6	0,5494	0,0363	0,4767	$1{,}6497.10^{-14}$	$7{,}6480.10^{-9}$
7	**0,5234**	**0,0434**	**0,4451**	$\mathbf{1{,}3865.10^{-14}}$	$\mathbf{5{,}5070.10^{-3}}$
8	0,5447	0,0365	0,4758	$1{,}5286.10^{-14}$	$3{,}8642.10^{-7}$
9	0,5637	0,0341	0,4851	$1{,}7130.10^{-14}$	$1{,}7003.10^{-8}$
10	0,5355	0,0386	0,4616	$1{,}4465.10^{-14}$	$3{,}2017.10^{-11}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$1{,}4620.10^{-7}$	4498,46	$1{,}7147.10^{-13}$	$1{,}3548.10^{-5}$	94
6	$1{,}8195.10^{-10}$	4004,52	$2{,}0350.10^{-14}$	$9{,}4606.10^{-6}$	100
7	$\mathbf{1{,}7735.10^{-14}}$	**2804,14**	$\mathbf{2{,}1238.10^{-14}}$	$\mathbf{1{,}2392.10^{-5}}$	**98**
8	$1{,}7072.10^{-10}$	1890,4	$1{,}4071.10^{-13}$	$1{,}1140.10^{-5}$	100
9	$7{,}4664.10^{-11}$	4974,46	$1{,}9761.10^{-14}$	$1{,}0277.10^{-5}$	100
10	$1{,}7130.10^{-14}$	2393,24	$1{,}9216.10^{-14}$	$8{,}9651.10^{-6}$	100

Tabulka 30. Nejlepší řešení logické rovnice, p-2 orbit, CF Targ2, Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	0,5479	0,0206	0,4923	$1{,}5997.10^{-14}$	$1{,}8622.10^{-2}$
12	0,5456	0,0834	0,4839	$1{,}6675.10^{-14}$	$6{,}6479.10^{-9}$
13	**0,5391**	**0,0393**	**0,4613**	$\mathbf{1{,}4665.10^{-14}}$	$\mathbf{4{,}1711.10^{-7}}$
14	0,5687	0,0672	0,4979	$1{,}8018.10^{-14}$	$3{,}1420\text{E}.10^{-5}$
15	0,5427	0,0384	0,4709	$1{,}6186.10^{-14}$	$1{,}9267.10^{-8}$
16	0,5341	0,0386	0,4620	$1{,}5053.10^{-14}$	$4{,}5145.10^{-9}$

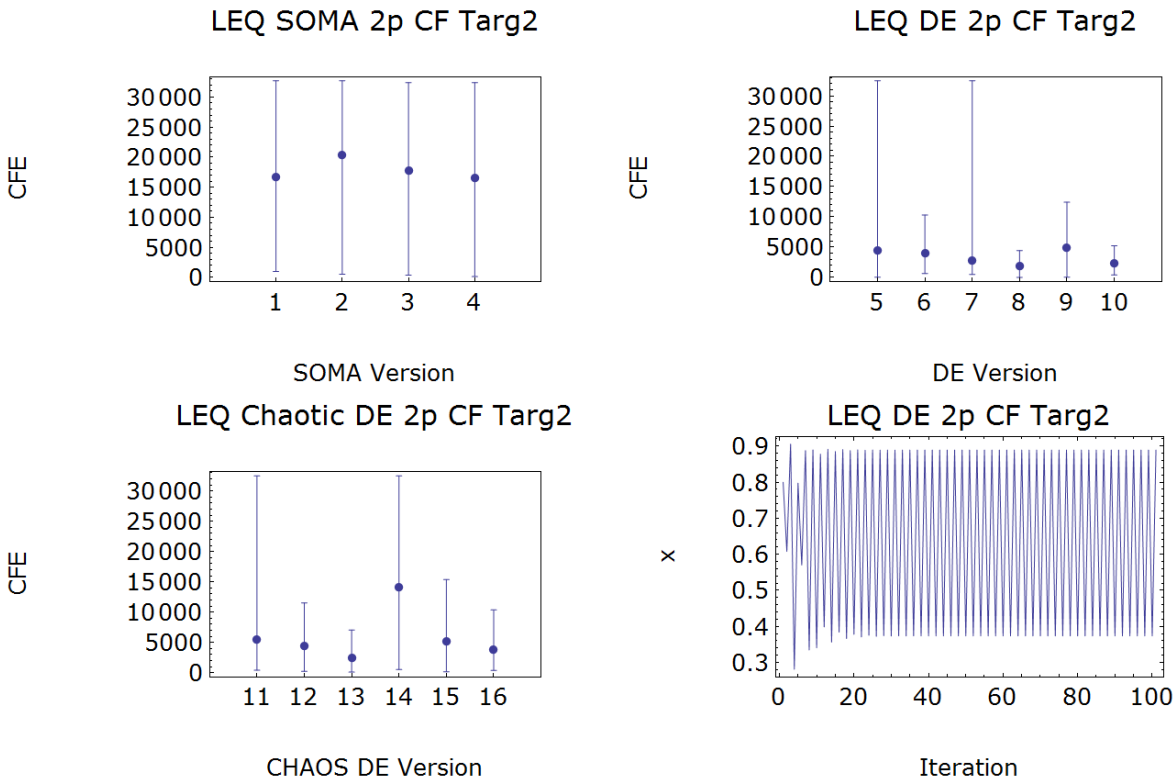
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$1{,}6204.10^{-6}$	5433,38	$1{,}9772.10^{-14}$	$1{,}3440.10^{-5}$	94
12	$1{,}8351.10^{-10}$	4372,8	$1{,}3671.10^{-13}$	$1{,}0525.10^{-5}$	100
13	$\mathbf{1{,}7623.10^{-10}}$	**2489,52**	$\mathbf{1{,}3674.10^{-8}}$	$\mathbf{1{,}1619.10^{-5}}$	**100**
14	$1{,}3824.10^{-6}$	14111,6	$2{,}1260.10^{-14}$	$9{,}3701.10^{-5}$	80
15	$1{,}4589.10^{-10}$	5150,16	$2{,}0550.10^{-14}$	$7{,}3505.10^{-5}$	100
16	$1{,}9428.10^{-14}$	3768,84	$1{,}3743.10^{-13}$	$1{,}0791.10^{-5}$	100

Zvýrazněný řádek strategie AllToAll algoritmu SOMA v *tabulce 28* ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}7685.10^{-14}$ pro nastavení Logické rovnice, p-2 orbit, CF Targ2.

Nejlepší variantou Diferenciální evoluce pro nastavení logické rovnice, p-2 orbit, CF Targ2 je DEBest2Bin. Minimální hodnota účelové funkce je *Min CF Val* = $1{,}3865.10^{-14}$. To ukazuje *tabulka 29*.

Zvýrazněný řádek varianta DEBest2Bin Chaotické Diferenciální evoluce ukazuje v *tabulce 30* minimální hodnotu účelové funkce *Min CF Val* = $1{,}4665.10^{-14}$ pro nastavení Logické rovnice, p-2 orbit, CF Targ2.

Pro nastavení simulace Logické rovnice, p-2 orbit, CF Targ2 se jako nejlepší evoluční algoritmus pro řízení chaotického systému ukazovala Diferenciální evoluce ve variantě DEBest2Bin. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 28* dole vpravo



Obrázek 28. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro logickou rovnici, p-2 orbit, CF Targ2

p-4 orbit

Tabulka 31. Nejlepší řešení logické rovnice, p-4 orbit, CF Targ2, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	-0,5233	0,4478	0,6138	$1{,}9322.10^{-4}$	0,00398
2	-0,5250	0,8974	0,6143	$1{,}9521.10^{-4}$	0,01394
3	**-0,5353**	**0,1685**	**0,6198**	$\mathbf{1{,}8519.10^{-4}}$	**0,03114**
4	-0,5212	0,1967	0,6075	$1{,}9365.10^{-4}$	0,01488

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$2{,}1720.10^{-4}$	26371,26	$1{,}9937.10^{-4}$	$2{,}1880.10^{-4}$	56
2	$2{,}1870.10^{-3}$	29540,92	$1{,}9707.10^{-4}$	$2{,}1477.10^{-4}$	28
3	$\mathbf{2{,}2037.10^{-3}}$	**28837,3**	$\mathbf{1{,}9919.10^{-4}}$	$\mathbf{2{,}1245.10^{-4}}$	**26**
4	$2{,}0331.10^{-3}$	28399,18	$1{,}9767.10^{-4}$	$2{,}2096.10^{-4}$	38

Tabulka 32. Nejlepší řešení logické rovnice, p-4 orbit, CF Targ2, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	-0,5240	0,1695	0,6124	$1{,}5390.10^{-4}$	$9{,}8389.10^{-3}$
6	-0,5326	0,1683	0,6248	$1{,}6407.10^{-4}$	$1{,}9272.10^{-4}$
7	**-0,5255**	**0,1697**	**0,6082**	$\mathbf{1{,}5221.10^{-4}}$	$\mathbf{4{,}0409.10^{-2}}$
8	-0,5243	0,1690	0,6128	$1{,}5419.10^{-4}$	$3{,}8446.10^{-2}$
9	-0,5243	0,1894	0,6150	$1{,}8248.10^{-4}$	$5{,}9944.10^{-4}$
10	-0,5226	0,1695	0,6133	$1{,}5429.10^{-4}$	$1{,}8201.10^{-4}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$1{,}9315.10^{-4}$	10284	$1{,}9507.10^{-4}$	$2{,}1997.10^{-4}$	86
6	$1{,}9288.10^{-4}$	11162,1	$1{,}9171.10^{-4}$	$2{,}1629.10^{-4}$	100
7	$\mathbf{1{,}8559.10^{-4}}$	**8232,98**	$\mathbf{1{,}9686.10^{-4}}$	$\mathbf{2{,}1817.10^{-4}}$	**88**
8	$1{,}9315.10^{-4}$	8447,1	$1{,}9520.10^{-4}$	$2{,}2304.10^{-4}$	90
9	$1{,}9795.10^{-4}$	16536,8	$1{,}9791.10^{-4}$	$2{,}1940.10^{-4}$	98
10	$1{,}8448.10^{-4}$	9103	$1{,}8949.10^{-4}$	$2{,}1505.10^{-4}$	100

Tabulka 33. Nejlepší řešení logické rovnice, p-4 orbit, CF Targ2, Chaotické DE.

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	-0,5264	0,0381	0,6190	$1{,}7619.10^{-4}$	$7{,}3417.10^{-2}$
12	-0,5233	0,1690	0,6131	$1{,}6809.10^{-4}$	$2{,}2956.10^{-4}$
13	**-0,5206**	**0,1700**	**0,6028**	$\mathbf{1{,}4997.10^{-4}}$	$\mathbf{2{,}4025.10^{-3}}$
14	-0,5230	0,8808	0,6154	$2{,}0679.10^{-4}$	$1{,}3176.10^{-2}$
15	-0,5263	0,1817	0,6148	$1{,}6900.10^{-4}$	$3{,}6204.10^{-3}$
16	-0,5247	0,1694	0,6134	$1{,}5869.10^{-4}$	$4{,}3113.10^{-3}$

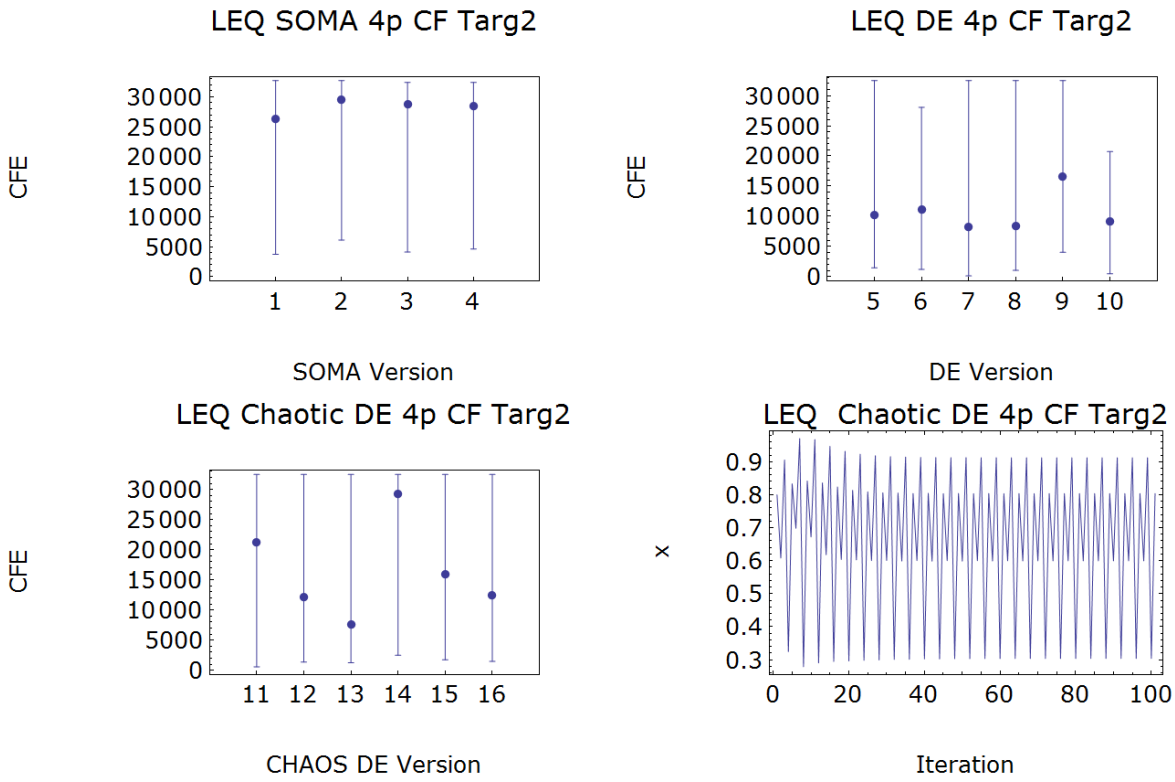
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$1{,}9214.10^{-3}$	21181,4	$1{,}9925.10^{-4}$	$2{,}2419.10^{-4}$	42
12	$1{,}9335.10^{-4}$	12099,7	$1{,}9908.10^{-4}$	$2{,}2126.10^{-4}$	98
13	$\mathbf{1{,}8863.10^{-4}}$	**7662,82**	$\mathbf{1{,}9909.10^{-4}}$	$\mathbf{2{,}2006.10^{-4}}$	**98**
14	$2{,}2186.10^{-3}$	29306,1	$2{,}0679.10^{-4}$	$2{,}2380.10^{-4}$	20
15	$1{,}9992.10^{-4}$	15875,3	$1{,}9926.10^{-4}$	$2{,}1825.10^{-4}$	90
16	$1{,}9152.10^{-4}$	12480,8	$1{,}9263.10^{-4}$	$2{,}1605.10^{-4}$	96

Nejlepší strategií SOMA pro nastavení logické rovnice, p-4 orbit, CF Targ2 je AllToAll. Minimální hodnota účelové funkce je *Min CF Val* = 1,8519.10^{-4}. Tu zobrazuje *tabulka 31*.

Zvýrazněný řádek varianty DEBest2Bin Diferenciální evoluce ukazuje v *tabulce 32* minimální hodnotu účelové funkce *Min CF Val* = 1,5221.10^{-4} pro nastavení Logické rovnice, p-4 orbit, CF Targ2.

Nejlepší variantou Chaotické Diferenciální evoluce pro nastavení logické rovnice, p-4 orbit, CF Targ2 je DEBest2Bin. Minimální hodnota účelové funkce je *Min CF Val* = 1,4997.10^{-4}. To ukazuje *tabulka 33*.

Pro nastavení simulace Logické rovnice, p-4 orbit, CF Targ2 se jako nejlepší evoluční algoritmus pro řízení chaotického systému ukazovala Diferenciální evoluce ve variantě DEBest2Bin. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 25* dole vpravo



Obrázek 29. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro logickou rovnici, p-4 orbit, CF Targ2

9.2 Henonova mapa

Henonova mapa je dvojdimenzionální diskrétní chaotický systém, na kterém byla testována výkonnost evolučních algoritmu. Pro každou kombinaci byl vytvořen program

v prostředí *Mathematica 7.0* který pro algoritmus SOMA, Diferenciální evoluci a chaotickou Diferenciální evoluci prováděl testování na účelových funkcích CF Targ1 a CF Targ2

9.2.1 CF Targ1

Výsledky, které nám podal program, byly vyneseny do následujících tabulek. Každý evoluční algoritmus pracoval pro p-1 orbit, p-2 a p-4 orbit. Strategie evolučních algoritmů jsou shodné s označením sloupce index v *tabulce 4, tabulce 5* a *tabulce 6.*

p-1 orbit

Tabulka 34. Nejlepší řešení Henonovy mapy, p-1 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	**-0,8601**	**0,2265**	**0,2169**	**$3{,}8.10^{-15}$**	**$4{,}562.10^{-15}$**
2	-0,8585	0,1835	0,2074	$3{,}8.10^{-15}$	$4{,}366.10^{-15}$
3	-0,8297	0,2619	0,2010	$3{,}9.10^{-15}$	$4{,}424.10^{-15}$
4	-0,8565	0,2636	0,2207	$3{,}8.10^{-15}$	$4{,}366.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	**$4{,}4.10^{-15}$**	**2138,14**	**$4{,}9.10^{-15}$**	**$6{,}928.10^{-15}$**	**100**
2	$4{,}3.10^{-15}$	2791,06	4.10^{-15}	$6{,}838.10^{-15}$	100
3	$4{,}45.10^{-15}$	4467,28	$4{,}9.10^{-15}$	$6{,}73.10^{-15}$	100
4	$4{,}4.10^{-15}$	1720,78	$4{,}2.10^{-15}$	$6{,}338.10^{-15}$	100

Nejlepší strategií algoritmu SOMA pro nastavení Henonovy mapy, p-1 orbit, CF Targ1 je AllToOne. Minimální hodnota účelové funkce je *Min CF Val* = $3{,}8.10^{-15}$.

Tabulka 35. Nejlepší řešení Henonovy mapy, p-1 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	**-0,8439**	**0,2792**	**0,2091**	**$3{,}7.10^{-15}$**	**$4{,}004.10^{-15}$**
6	-0,8387	0,2683	0,2084	$3{,}7.10^{-15}$	$3{,}866.10^{-15}$
7	-0,8247	0,2606	0,2016	$3{,}7.10^{-15}$	$4{,}102.10^{-15}$
8	-0,8348	0,2597	0,2017	$3{,}7.10^{-15}$	$3{,}93.10^{-15}$
9	-0,8390	0,2664	0,2054	$3{,}8.10^{-15}$	$4{,}14.10^{-15}$
10	-0,8296	0,2628	0,2039	$3{,}7.10^{-15}$	$4{,}06.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	**4.10^{-15}**	**371,7**	**$4{,}2.10^{-15}$**	**$6{,}274.10^{-15}$**	**100**
6	$3{,}9.10^{-15}$	389,96	$4{,}6.10^{-15}$	$6{,}698.10^{-15}$	100
7	$3{,}8.10^{-15}$	294,98	$4{,}8.10^{-15}$	$6{,}796.10^{-15}$	100
8	$3{,}9.10^{-15}$	340,72	$4{,}3.10^{-15}$	$6{,}254.10^{-15}$	100
9	$4{,}1.10^{-15}$	467,58	5.10^{-15}	$6{,}738.10^{-15}$	100
10	$4{,}1.10^{-15}$	469,88	$4{,}6.10^{-15}$	$6{,}754.10^{-15}$	100

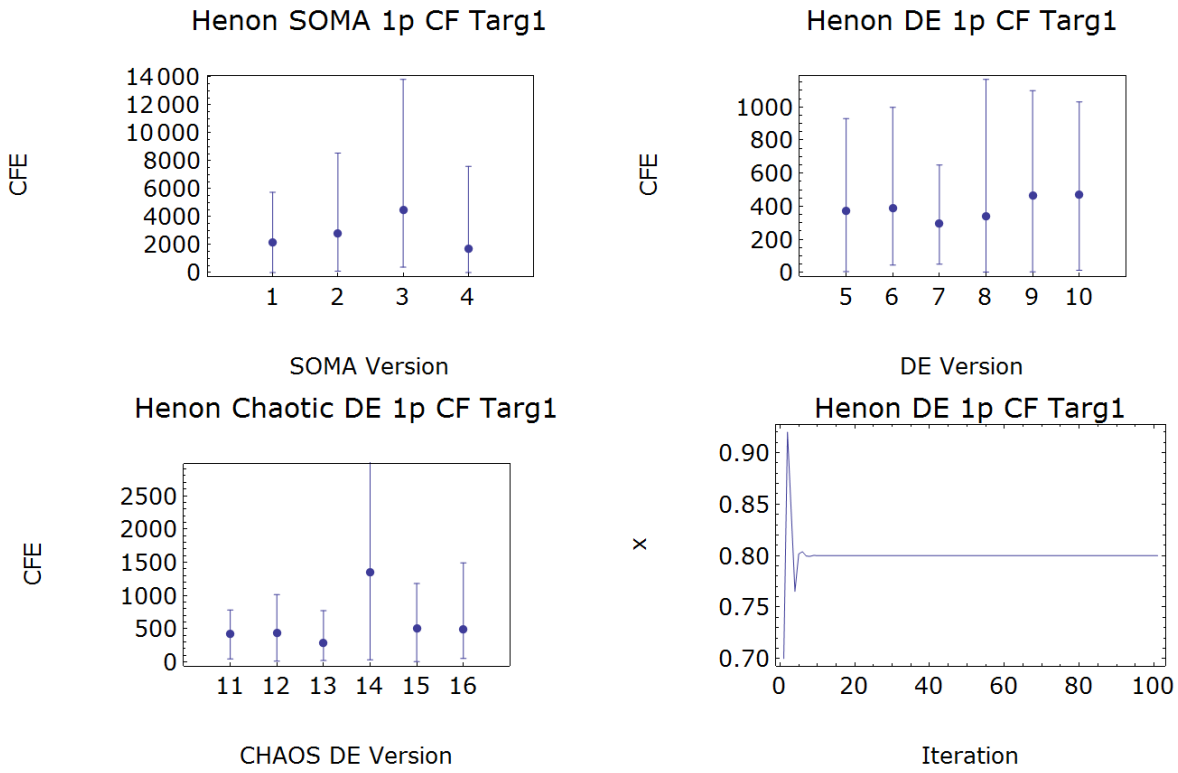
Zvýrazněný řádek varianty DERand1Bin Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $3{,}7.10^{-15}$ pro nastavení Henonovy mapy, p-1 orbit, CF Targ1.

Tabulka 36. Nejlepší řešení Henonovy mapy, p-1 orbit, CF Targ1,Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	**-0,8328**	**0,2613**	**0,2012**	**$3{,}7.10^{-15}$**	**$4{,}32.10^{-15}$**
12	-0,8353	0,2647	0,2030	$3{,}7.10^{-15}$	$3{,}89.10^{-15}$
13	-0,8408	0,2323	0,2002	$3{,}7.10^{-15}$	$3{,}97.10^{-15}$
14	-0,8249	0,2599	0,2010	$4{,}2.10^{-15}$	$5{,}01.10^{-15}$
15	-0,8342	0,2649	0,2069	$3{,}9.10^{-15}$	$4{,}18.10^{-15}$
16	-0,8407	0,2733	0,2080	$3{,}8.10^{-15}$	$4{,}06.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	**$4{,}25.10^{-15}$**	**428,26**	**$4{,}4.10^{-15}$**	**$6{,}434.10^{-15}$**	**100**
12	$3{,}9.10^{-15}$	436,84	$4{,}5.10^{-15}$	$6{,}596.10^{-15}$	100
13	$3{,}8.10^{-15}$	292,48	$4{,}6.10^{-15}$	$6{,}566.10^{-15}$	100
14	$4{,}9.10^{-15}$	1354,92	$4{,}3.10^{-15}$	$6{,}83.10^{-15}$	100
15	$4{,}2.10^{-15}$	511,66	$5{,}3.10^{-15}$	$6{,}894.10^{-15}$	100
16	4.10^{-15}	496,88	$4{,}8.10^{-15}$	$6{,}754.10^{-15}$	100

Nejlepší variantou Chaotické Diferenciální evoluce pro nastavení Henonovy mapy, p-1 orbit, CF Targ1 je DERand1Bin. Minimální hodnota účelové funkce je *Min CF Val* = $3{,}7.10^{-15}$.

Pro nastavení simulace Henonovy mapy, p-1 orbit, CF Targ1 se jako nejlepší evoluční algoritmus pro řízení ukazovala Diferenciální evoluce ve variantě DERand1Bin. Má sice minimální hodnotu účelové funkce stejnou jako Chaotická Diferenciální evoluce, nicméně průměrnou hodnotu účelové funkce má klasická Diferenciální evoluce o tři desetiny nižší. Proto byla zvolena právě tato varianta pro zobrazení simulace stabilizace chaotického systému na *obrázku 30* dole vpravo



Obrázek 30. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro Henonovu mapu, p-1 orbit, CF Targ1

p-2 orbit

Tabulka 37. Nejlepší řešení Henonovy mapy, p-2 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	0,4311	0,1901	0,3406	$6{,}8195.10^{-9}$	$8{,}5436.10^{-3}$
2	0,6162	0,5328	0,5030	$9{,}8272.10^{-8}$	$1{,}2922.10^{-2}$
3	**0,4821**	**0,1598**	**0,4842**	$\mathbf{7{,}9238.10^{-13}}$	$\mathbf{8{,}5434.10^{-3}}$
4	0,5193	0,5431	0,5094	$1{,}5011.10^{-12}$	$6{,}3652.10^{-3}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$2{,}9835.10^{-3}$	8681,52	$7{,}5598.10^{-8}$	$3{,}8297.10^{-2}$	98
2	$3{,}6095.10^{-3}$	11429,5	$3{,}6038.10^{-5}$	$4{,}7424.10^{-2}$	98
3	$\mathbf{7{,}7064.10^{-4}}$	**10140,1**	$\mathbf{1{,}1355.10^{-12}}$	$\mathbf{2{,}9445.10^{-2}}$	**98**
4	$9{,}6539.10^{-4}$	8203,32	$1{,}0117.10^{-7}$	$4{,}1216.10^{-2}$	100

Zvýrazněný řádek strategie AllToAll algoritmu SOMA ukazuje minimální hodnotu účelové funkce *Min CF Val* = $3{,}7.10^{-15}$ pro nastavení Henonovy mapy, p-2 orbit, CF Targ1.

Tabulka 38. Nejlepší řešení Henonovy mapy, p-2 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	0,4077	0,7757	0,3140	$4{,}8569.10^{-9}$	$2{,}6964.10^{-3}$
6	0,4812	0,1581	0,4767	$9{,}6384.10^{-13}$	$1{,}6492.10^{-3}$
7	0,4850	0,1569	0,4872	$5{,}7588.10^{-13}$	$4{,}6563.10^{-1}$
8	**0,4918**	**0,1596**	**0,4838**	**$4{,}8176.10^{-13}$**	**$2{,}4279.10^{-3}$**
9	0,4924	0,1603	0,4868	$6{,}0966.10^{-13}$	$1{,}4455.10^{-3}$
10	0,5312	0,5316	0,5042	$1{,}1388.10^{-13}$	$1{,}3701.10^{-3}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$2{,}2248.10^{-3}$	1424,08	$1{,}0322.10^{-7}$	$4{,}5179.10^{-2}$	100
6	$2{,}2468.10^{-3}$	1796,86	$1{,}0040.10^{-5}$	$4{,}3933.10^{-2}$	100
7	$1{,}7157.10^{-3}$	1772,22	$1{,}0584.10^{-4}$	$5{,}0795.10^{-2}$	98
8	**$2{,}1327.10^{-3}$**	**921,16**	**$1{,}2483.10^{-3}$**	**$4{,}5119.10^{-2}$**	**100**
9	$1{,}9448.10^{-3}$	2636,74	$1{,}7031.10^{-7}$	$4{,}2163.10^{-2}$	100
10	$1{,}8274.10^{-3}$	2340,38	$4{,}5847.10^{-5}$	$4{,}0254.10^{-2}$	100

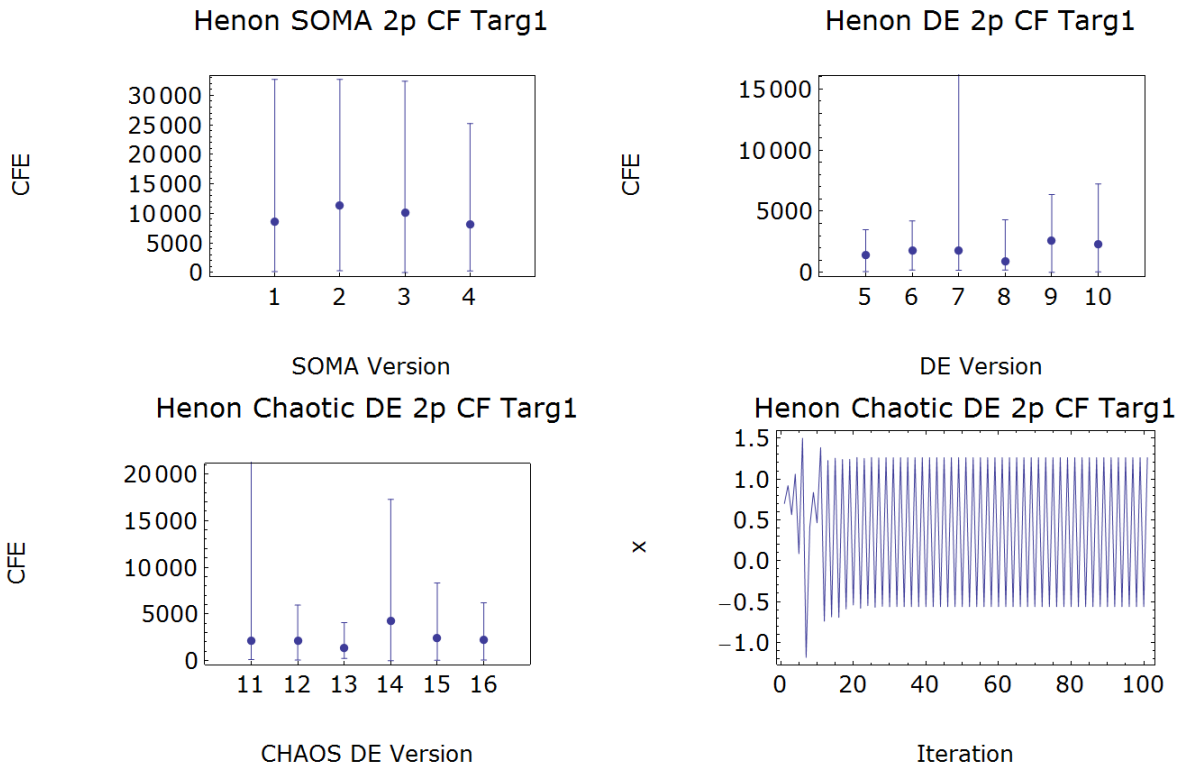
Nejlepší variantou Diferenciální evoluce pro nastavení Henonovy mapy, p-2 orbit, CF Targ1 je DELocalToBest. Minimální hodnota účelové funkce je *Min CF Val*= $4{,}8176.10^{-13}$.

Tabulka 39. Nejlepší řešení Henonovy mapy, p-2 orbit, CF Targ1,Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	**0,4715**	**0,5815**	**0,3472**	**$4{,}7548.10^{-13}$**	**$3{,}6039.10^{-1}$**
12	0,5280	0,5116	0,5117	$2{,}9588.10^{-12}$	$1{,}7068.10^{-3}$
13	0,4958	0,1155	0,4994	$7{,}3755.10^{-13}$	$1{,}403.10^{-3}$
14	0,5189	0,5354	0,5016	$1{,}6455.10^{-12}$	$5{,}0802.10^{-4}$
15	0,5162	0,5288	0,5017	$8{,}0681.10^{-13}$	$7{,}7735.10^{-4}$
16	0,4948	0,1576	0,4891	$8{,}7897.10^{-13}$	$1{,}6480.10^{-3}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	**$2{,}9650.10^{-3}$**	**2095,78**	**$8{,}4281.10^{-6}$**	**$4{,}8799.10^{-2}$**	**98**
12	$2{,}1681.10^{-3}$	2096,48	$6{,}1465.10^{-10}$	$3{,}7407.10^{-2}$	100
13	$1{,}7157.10^{-3}$	1316,88	$3{,}1603.10^{-5}$	$4{,}1620.10^{-2}$	100
14	$9{,}8610.10^{-6}$	4217,02	$2{,}6977.10^{-7}$	$2{,}7152.10^{-2}$	100
15	$5{,}6023.10^{-6}$	2436,94	$1{,}9808.10^{-7}$	$3{,}4457.10^{-2}$	100
16	$2{,}0293.10^{-3}$	2276,5	$5{,}4878.10^{-7}$	$3{,}8351.10^{-2}$	100

Zvýrazněný řádek varianty DERand1Bin chaotické Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $4{,}7548.10^{-13}$ pro nastavení Henonovy mapy, p-2 orbit, CF Targ1.

Pro nastavení simulace Henonovy mapy, p-2 orbit, CF Targ1 se jako nejlepší evoluční algoritmus pro řízení chaotického systému ukazovala Chaotická Diferenciální evoluce ve variantě DEBest2Bin. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 31* dole vpravo.



Obrázek 31. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro Henonovu mapu, p-2 orbit, CF Targ1

p-4 orbit

Tabulka 40. Nejlepší řešení Henonovy mapy, p-4 orbit, CF Targ1, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	-0,3732	0,11267	0,41413	$1{,}0707.10^{-6}$	0,00021
2	-0,3772	0,48292	0,43312	$5{,}6925.10^{-7}$	0,00019
3	-0,3809	0,26188	0,42637	$1{,}1864.10^{-6}$	0,00015
4	**-0,3787**	**0,11325**	**0,42878**	**$4{,}8612.10^{-7}$**	**0,0001**
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$4{,}72887.10^{-5}$	6375,78	$8{,}51555.10^{-6}$	0,00473	100
2	$5{,}99968.10^{-5}$	6203,78	$3{,}8499.10^{-6}$	0,00381	100
3	$5{,}3851.10^{-5}$	7344,04	$1{,}52209.10^{-6}$	0,00348	100
4	**$2{,}61284.10^{-5}$**	**5804,38**	**$4{,}27272.10^{-6}$**	**0,00347**	**100**

Nejlepší strategií algoritmu SOMA pro nastavení Henonovy mapy, p-4 orbit, CF Targ1 je AllToAllAdaptive. Minimální hodnota účelové funkce je *Min CF Val* = $4{,}8612.10^{-7}$.

Tabulka 41. Nejlepší řešení Henonovy mapy, p-4 orbit, CF Targ1, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	**-0,3575**	**0,2485**	**0,3762**	**$2{,}2566.10^{-8}$**	**$3{,}6169.10^{-5}$**
6	-0,3633	0,2239	0,3974	$1{,}8631.10^{-7}$	$4{,}1887.10^{-7}$
7	-0,3599	0,2278	0,3900	$5{,}3321.10^{-8}$	$3{,}9073.10^{-7}$
8	-0,3633	0,2221	0,3983	$8{,}4963.10^{-8}$	$8{,}0852.10^{-6}$
9	-0,3682	0,2203	0,4068	$2{,}5629.10^{-7}$	$4{,}6370.10^{-7}$
10	-0,3683	0,2167	0,4089	$1{,}5919.10^{-7}$	$4{,}1056.10^{-7}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	**$1{,}1296.10^{-6}$**	**865,18**	**$8{,}5494.10^{-6}$**	**$3{,}2404.10^{-3}$**	**100**
6	$4{,}6047.10^{-7}$	1229,08	$6{,}6147.10^{-6}$	$4{,}4752.10^{-3}$	100
7	$4{,}6047.10^{-7}$	776,72	$3{,}8166.10^{-5}$	$3{,}9660.10^{-3}$	100
8	$4{,}6047.10^{-7}$	624,7	$2{,}7268.10^{-6}$	$3{,}4744.10^{-3}$	100
9	$4{,}6048.10^{-7}$	1847,48	$5{,}4512.10^{-6}$	$3{,}5270.10^{-3}$	100
10	$4{,}6047.10^{-7}$	1396,5	$7{,}2317.10^{-6}$	$4{,}4540.10^{-3}$	100

Zvýrazněný řádek varianta DERand1Bin Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $2{,}2566.10^{-8}$ pro nastavení Henonovy mapy, p-4 orbit, CF Targ1.

Tabulka 42. Nejlepší řešení Henonovy mapy, p-4 orbit, CF Targ1,Chaotické DE

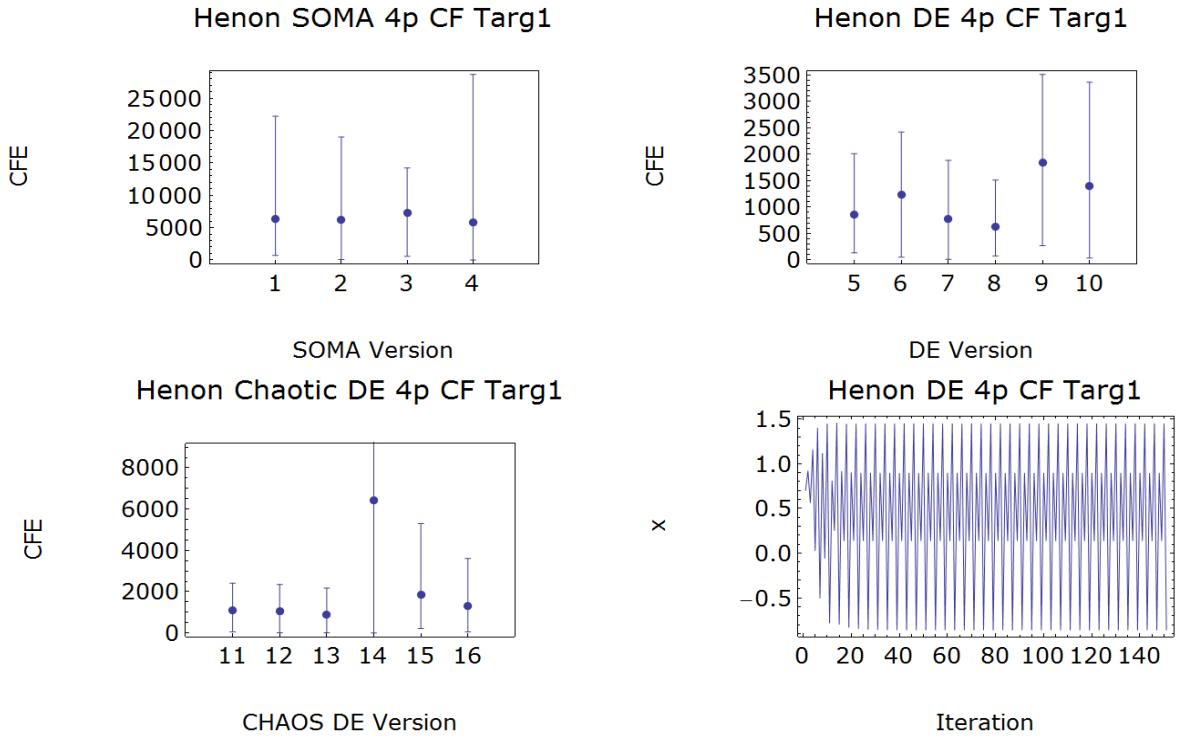
EA	K	Fmax	R	Min CF Val	Avg CF Val
11	-0,3689	0,2125	0,4121	$1{,}8979.10^{-7}$	$1{,}1813.10^{-4}$
12	-0,3698	0,2285	0,4063	$2{,}8025.10^{-7}$	$1{,}7099.10^{-6}$
13	**-0,3640**	**0,2649**	**0,4015**	**$1{,}0067.10^{-7}$**	**$5{,}9999.10^{-7}$**
14	-0,3786	0,2581	0,4182	$5{,}7809.10^{-6}$	$1{,}1358.10^{-3}$
15	-0,3667	0,2237	0,4031	$2{,}3089.10^{-7}$	$8{,}0344.10^{-7}$
16	-0,3653	0,3064	0,4037	$1{,}9863.10^{-7}$	$4{,}9204.10^{-7}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$2{,}2695.10^{-7}$	1105,1	$1{,}7890.10^{-5}$	$3{,}9466.10^{-3}$	100
12	$4{,}6047.10^{-7}$	1064	$7{,}7474.10^{-7}$	$3{,}2026.10^{-3}$	100
13	**$4{,}6047.10^{-7}$**	**879,74**	**$8{,}9662.10^{-7}$**	**$3{,}3426.10^{-3}$**	**100**
14	$9{,}8096.10^{-5}$	6406,34	$5{,}7809.10^{-6}$	$3{,}3102.10^{-3}$	98
15	$5{,}7636.10^{-7}$	1838,42	$1{,}4706.10^{-6}$	$3{,}6945.10^{-3}$	100
16	$4{,}6047.10^{-7}$	1321,24	$2{,}2262.10^{-6}$	$2{,}8241.10^{-3}$	100

Nejlepší variantou Chaotické Diferenciální evoluce pro nastavení Henonovy mapy, p-4 orbit, CF Targ1 je DEBest2Bin. Minimální hodnota účelové funkce je *Min CF Val* =$1{,}0067.10^{-7}$.

Pro nastavení simulace Henonovy mapy, p-4 orbit, CF Targ1 se jako nejlepší evoluční algoritmus pro řízení chaotického systému ukazovala Diferenciální evoluce ve variantě

DERand1Bin. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 32* dole vpravo.



Obrázek 32. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro Henonovu mapu, p-4 orbit, CF Targ1

9.2.2 CF Targ2

Výsledky, které nám podal program, byly vyneseny do následujících tabulek. Každý evoluční algoritmus pracoval pro p-1 orbit, p-2 a p-4 orbit.

p-1 orbit

Tabulka 43. Henonovy mapy, p-1 orbit, CF Targ2, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	-0,7062	0,8357	0,1001	$3{,}9992.10^{-15}$	$4{,}2129.10^{-15}$
2	-0,7157	0,1919	0,1063	$3{,}8.10^{-15}$	$4{,}2178.10^{-15}$
3	-0,6884	0,1921	0,0889	$3{,}8.10^{-15}$	$4{,}2079.10^{-15}$
4	**-0,7219**	**0,1863**	**0,1099**	$\mathbf{3{,}7.10^{-15}}$	$\mathbf{4{,}1404.10^{-15}}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$4{,}1992.10^{-15}$	57	$5{,}7433.10^{-15}$	$7{,}8855.10^{-15}$	100
2	$4{,}2.10^{-15}$	85,78	$5{,}7764.10^{-15}$	$8{,}0781.10^{-15}$	100
3	$4{,}2051.10^{-15}$	239,46	$6{,}1212.10^{-15}$	$8{,}0924.10^{-15}$	100
4	$\mathbf{4{,}1.10^{-15}}$	**111,32**	$\mathbf{5{,}6992.10^{-15}}$	$\mathbf{7{,}9272.10^{-15}}$	**100**

Zvýrazněný řádek strategie AllToAllAdaptive algoritmu SOMA ukazuje minimální hodnotu účelové funkce *Min CF Val* = $3{,}7.10^{-15}$ pro nastavení Henonovy mapy, p-1 orbit, CF Targ2.

Tabulka 44. Nejlepší řešení Henonovy mapy, p-1 orbit, CF Targ2, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	**-0,8407**	**0,2730**	**0,2076**	**$3{,}7.10^{-15}$**	**$3{,}9855.10^{-15}$**
6	-0,8347	0,2592	0,2012	$3{,}8992.10^{-15}$	$3{,}9595.10^{-15}$
7	-0,8238	0,2605	0,2036	$3{,}7992.10^{-15}$	$3{,}9074.10^{-15}$
8	-0,8433	0,2256	0,2002	$3{,}7.10^{-15}$	$3{,}9376.10^{-15}$
9	-0,8356	0,2583	0,2008	$3{,}7992.10^{-15}$	$3{,}9937.10^{-15}$
10	-0,7068	0,3016	0,1005	$3{,}8.10^{-15}$	$4{,}0035.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	**$3{,}9992.10^{-15}$**	**20,62**	**$5{,}5.10^{-15}$**	**$8{,}0815.10^{-15}$**	**100**
6	$3{,}9992.10^{-15}$	17,44	$5{,}3653.10^{-15}$	$8{,}1084.10^{-15}$	100
7	$3{,}8992.10^{-15}$	18,58	$5{,}7764.10^{-15}$	$7{,}7885.10^{-15}$	100
8	$3{,}8996.10^{-15}$	19,42	$6{,}1764.10^{-15}$	$8{,}2505.10^{-15}$	100
9	$3{,}9992.10^{-15}$	17,98	$5{,}5220.10^{-15}$	$7{,}8769.10^{-15}$	100
10	$3{,}9992.10^{-15}$	23,36	$4{,}9661.10^{-15}$	$8{,}0526.10^{-15}$	100

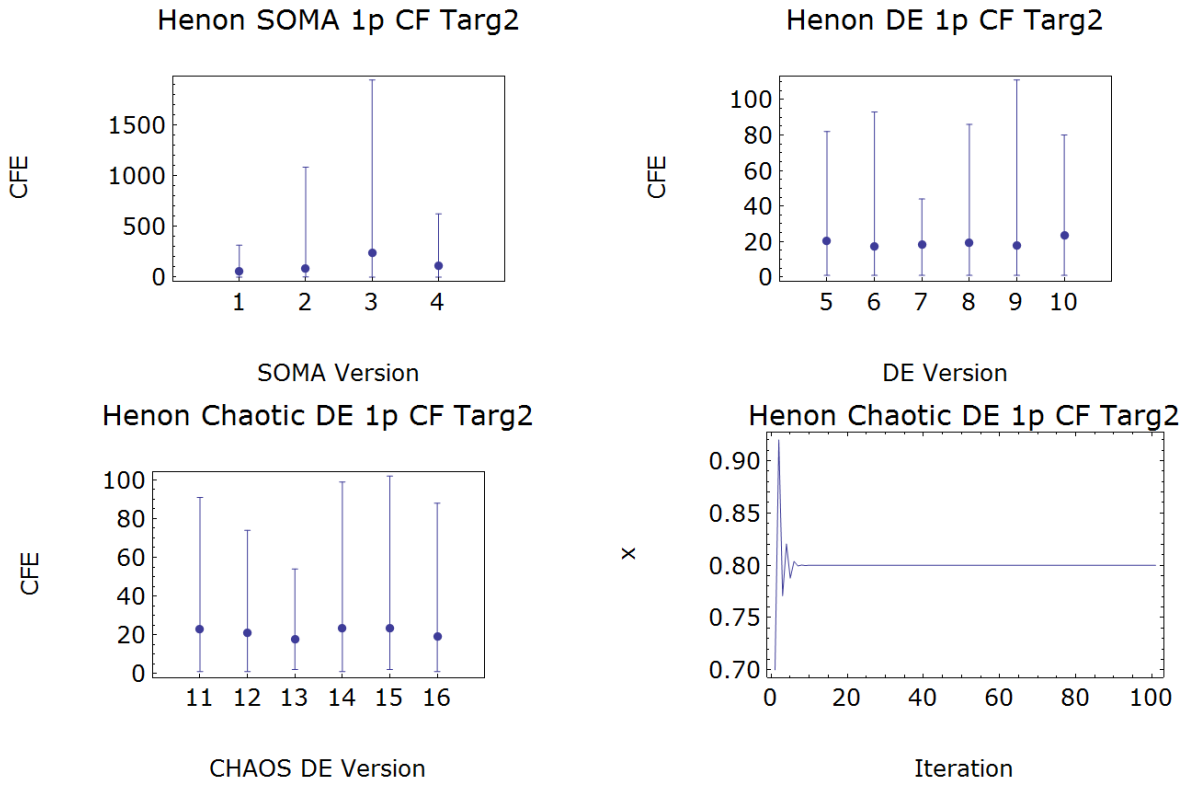
Zvýrazněný řádek varianty DERand1Bin Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $3{,}7.10^{-15}$ pro nastavení Henonovy mapy, p-1 orbit, CF Targ2.

Tabulka 45. Nejlepší řešení Henonovy mapy, p-1 orbit, CF Targ2, Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	-0,8432	0,2736	0,2106	$3{,}8.10^{-15}$	$4{,}0326.10^{-15}$
12	-0,8586	0,2047	0,2123	$3{,}8.10^{-15}$	$3{,}9735.10^{-15}$
13	**-0,8585**	**0,2072**	**0,2113**	**$3{,}7.10^{-15}$**	**$3{,}8969.10^{-15}$**
14	-0,8550	0,2654	0,2210	$4{,}2.10^{-15}$	$4{,}7395.10^{-15}$
15	-0,8562	0,2635	0,2185	$3{,}8.10^{-15}$	$3{,}9945.10^{-15}$
16	-0,8484	0,1505	0,2014	$3{,}8.10^{-15}$	$4{,}0403.10^{-15}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$4{,}0492.10^{-15}$	22,74	$5{,}2764.10^{-15}$	$8{,}1556.10^{-15}$	100
12	$3{,}9992.10^{-15}$	21,22	$5{,}4874.10^{-15}$	$8{,}1020.10^{-15}$	100
13	**$3{,}8992.10^{-15}$**	**17,88**	**$5{,}8.10^{-15}$**	**$7{,}9744.10^{-15}$**	**100**
14	$4{,}7823.10^{-15}$	23,24	$5{,}4764.10^{-15}$	$7{,}7959.10^{-15}$	100
15	$3{,}9996.10^{-15}$	23,32	$5{,}5212.10^{-15}$	$7{,}8656.10^{-15}$	100
16	$4{,}0102.10^{-15}$	18,92	$5{,}9433.10^{-15}$	$8{,}2031.10^{-15}$	100

Zvýrazněný řádek varianty DEBest2Bin Chaotické Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $3{,}7.10^{-15}$ pro nastavení Henonovy mapy, p-1 orbit, CF Targ2.

Pro nastavení simulace Henonovy mapy, p-1 orbit, CF Targ2 se jako nejlepší evoluční algoritmus pro řízení ukazovala Chaotická Diferenciální evoluce ve variantě DEBest2Bin. Měla sice minimální hodnotu účelové funkce stejnou jako všechny ostatní evoluční algoritmy, nicméně průměrnou hodnotu účelové funkce měla chaotická Diferenciální evoluce o jednu desetinu nižší. Proto byla zvolena právě tato varianta pro zobrazení simulace stabilizace chaotického systému dole *obrázku 33* vpravo



Obrázek 33. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro Henonovu mapu, p-1 orbit, CF Targ2

p-2 orbit

Tabulka 46. Nejlepší řešení Henonovy mapy, p-2 orbit, CF Targ2, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	0,5447	0,6747	0,5276	$2{,}0216.10^{-13}$	$8{,}4708.10^{-5}$
2	0,5067	0,1776	0,4210	$1{,}4463.10^{-8}$	$2{,}9055.10^{-4}$
3	0,5440	0,6120	0,4646	$1{,}5586.10^{-11}$	$9{,}6191.10^{-5}$
4	**0,5175**	**0,5396**	**0,4919**	$\mathbf{1{,}4277.10^{-13}}$	$\mathbf{1{,}3918.10^{-4}}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	$8{,}3725.10^{-5}$	11691,3	$1{,}5997.10^{-8}$	$1{,}5311.10^{-4}$	100
2	$1{,}7557.10^{-5}$	18960,2	$1{,}4463.10^{-8}$	$1{,}2463.10^{-4}$	84
3	$2{,}1557.10^{-6}$	17159,1	$1{,}7974.10^{-11}$	$1{,}1941.10^{-4}$	96
4	$\mathbf{8{,}3677.10^{-6}}$	**13646,7**	$\mathbf{1{,}774.10^{-13}}$	$\mathbf{1{,}2647.10^{-4}}$	**94**

Zvýrazněný řádek strategie AllToAllAdaptive algoritmu SOMA ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}4277.10^{-13}$ pro nastavení Henonovy mapy, p-2 orbit, CF Targ2.

Tabulka 47. Nejlepší řešení Henonovy mapy, p-2 orbit, CF Targ2, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	0,4314	0,8360	0,3626	$1{,}9992.10^{-11}$	$1{,}3385.10^{-4}$
6	0,5143	0,5322	0,5098	$1{,}4810.10^{-13}$	$1{,}1222.10^{-4}$
7	**0,4913**	**0,1591**	**0,4807**	**$1{,}6797.10^{-14}$**	**$8{,}7393.10^{-5}$**
8	0,5162	0,5357	0,5002	$1{,}8107.10^{-14}$	$1{,}1480.10^{-4}$
9	0,5157	0,5328	0,5090	$1{,}4288.10^{-13}$	$6{,}5399.10^{-5}$
10	0,5208	0,5469	0,5022	$2{,}1105.10^{-14}$	$9{,}0273.10^{-5}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$1{,}5013.10^{-4}$	1962,38	$1{,}8275.10^{-9}$	$1{,}8832.10^{-4}$	100
6	$1{,}4654.10^{-4}$	3034,9	$1{,}9534.10^{-8}$	$1{,}6866.10^{-4}$	100
7	**$1{,}4310.10^{-4}$**	**1844,32**	**$1{,}7331.10^{-9}$**	**$1{,}5992.10^{-4}$**	**100**
8	$1{,}4829.10^{-4}$	1407,14	$2{,}1808.10^{-10}$	$1{,}6691.10^{-4}$	100
9	$8{,}0918.10^{-8}$	4402,3	$1{,}6203.10^{-8}$	$1{,}4451.10^{-4}$	100
10	$1{,}4382.10^{-4}$	3279,82	$2{,}0031.10^{-8}$	$1{,}4460.10^{-4}$	100

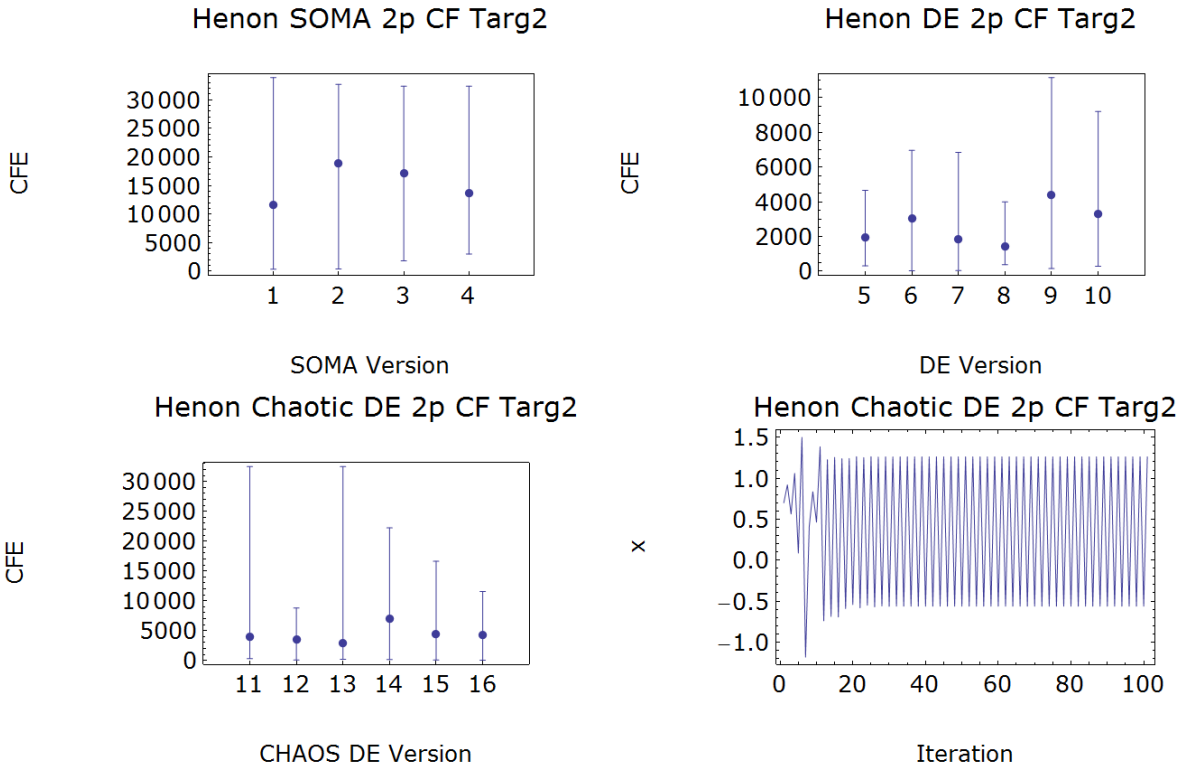
Nejlepší verzí Diferenciální evoluce pro nastavení Henonovy mapy, p-2 orbit, CF Targ2 je DELocalToBest. Minimální hodnota účelové funkce je *Min CF Val* = $1{,}6797.10^{-14}$.

Tabulka 48. Nejlepší řešení Henonovy mapy, p-2 orbit, CF Targ2,Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	0,4519	0,5509	0,3237	$1{,}3558.10^{-7}$	$2{,}1815.10^{-5}$
12	0,5299	0,5333	0,5066	$1{,}4687.10^{-13}$	$9{,}8191.10^{-5}$
13	**0,4838**	**0,1568**	**0,4643**	**$1{,}6908.10^{-14}$**	**$1{,}1962.10^{-5}$**
14	0,4938	0,1608	0,4965	$1{,}3799.10^{-13}$	$2{,}2529.10^{-5}$
15	0,5225	0,5368	0,4979	$2{,}0661.10^{-14}$	$4{,}8057.10^{-5}$
16	0,4884	0,1589	0,4840	$1{,}7108.10^{-14}$	$8{,}5643.10^{-5}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	$1{,}623.10^{-4}$	3909,44	$1{,}7385.10^{-7}$	$1{,}8493.10^{-4}$	94
12	$1{,}4599.10^{-4}$	3514,44	$1{,}5587.10^{-9}$	$1{,}7368.10^{-4}$	100
13	**$1{,}431.10^{-4}$**	**2839,86**	**$1{,}6202.10^{-6}$**	**$1{,}6566.10^{-4}$**	**98**
14	$1{,}4763.10^{-6}$	7015,24	$1{,}5080.10^{-8}$	$9{,}7040.10^{-5}$	100
15	$1{,}8340.10^{-8}$	4369,04	$1{,}5652.10^{-10}$	$1{,}3675.10^{-4}$	100
16	$1{,}4493.10^{-4}$	4309,72	$1{,}9162.10^{-8}$	$1{,}6542.10^{-4}$	100

Zvýrazněný řádek varianta DELocalToBest Chaotické Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1{,}6908.10^{-14}$ pro nastavení Henonovy mapy, p-2 orbit, CF Targ2.

Pro nastavení simulace Henonovy mapy, p-2 orbit, CF Targ2 se jako nejlepší evoluční algoritmus pro řízení chaotického systému ukazovala Diferenciální evoluce ve variantě DELocalToBest. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 34* dole vpravo



Obrázek 34. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro Henonovu mapu, p-2 orbit, CF Targ2

p-4 orbit

Tabulka 49. Nejlepší řešení Henonovy mapy, p-4 orbit, CF Targ2, SOMA

EA	K	Fmax	R	Min CF Val	Avg CF Val
1	**-0,3710**	**0,1075**	**0,4160**	**$1{,}4707.10^{-8}$**	**$3{,}1759.10^{-6}$**
2	-0,3762	0,2326	0,4140	$1{,}8579.10^{-8}$	$1{,}4324.10^{-5}$
3	-0,3754	0,1229	0,4260	$1{,}6504.10^{-8}$	$7{,}0706.10^{-6}$
4	-0,3729	0,2669	0,3922	$1{,}4984.10^{-8}$	$5{,}9369.10^{-6}$

EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
1	**$1{,}4223.10^{-6}$**	**8585,18**	**$1{,}725.10^{-7}$**	**$1{,}5626.10^{-5}$**	**100**
2	$1{,}9496.10^{-6}$	13785,8	$1{,}8579.10^{-8}$	$1{,}5779.10^{-5}$	94
3	$1{,}5151.10^{-6}$	14848,4	$1{,}6504.10^{-8}$	$1{,}3148.10^{-5}$	98
4	$1{,}4572.10^{-6}$	9159,12	$1{,}8039.10^{-7}$	$1{,}5230.10^{-5}$	98

Nejlepší strategií algoritmu SOMA pro nastavení Henonovy mapy, p-4 orbit, CF Targ2 je AllToAllAdaptive. Minimální hodnota účelové funkce je *Min CF Val* = $1{,}4707.10^{-8}$.

Tabulka 50. Nejlepší řešení Henonovy mapy, p-4 orbit, CF Targ2, DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
5	-0,3730	0,2172	0,4165	$1,484.10^{-8}$	$1,9103.10^{-6}$
6	-0,3636	0,2284	0,3961	$1,4055.10^{-8}$	$1,5936.10^{-8}$
7	-0,3612	0,2258	0,3930	$1,7718.10^{-9}$	$1,5002.10^{-8}$
8	**-0,3585**	**0,2502**	**0,3771**	**$1,4666.10^{-9}$**	**$1,5770.10^{-6}$**
9	-0,3684	0,2026	0,4113	$1,4910.10^{-8}$	$1,8787.10^{-8}$
10	-0,3699	0,1071	0,4150	$1,4327.10^{-8}$	$1,5977.10^{-8}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
5	$1,3921.10^{-7}$	1538,1	$1,49.10^{-7}$	$1,60.10^{-5}$	100
6	$1,5902.10^{-8}$	1488,58	$1,52.10^{-7}$	$1,28.10^{-5}$	100
7	$1,6369.10^{-8}$	1059,82	$1,86.10^{-7}$	$1,09.10^{-5}$	100
8	**$1,6369.10^{-8}$**	**886,74**	**$1,72.10^{-7}$**	**$1,51.10^{-5}$**	**100**
9	$1,6382.10^{-8}$	2424,98	$1,86.10^{-8}$	$1,33.10^{-5}$	100
10	$1,5739.10^{-8}$	2017,98	$1,37.10^{-7}$	$1,17.10^{-5}$	100

Zvýrazněný řádek varianty DELocalToBest Diferenciální evoluce ukazuje minimální hodnotu účelové funkce *Min CF Val* = $1,4666.10^{-9}$ pro nastavení Henonovy mapy, p-4 orbit, CF Targ2.

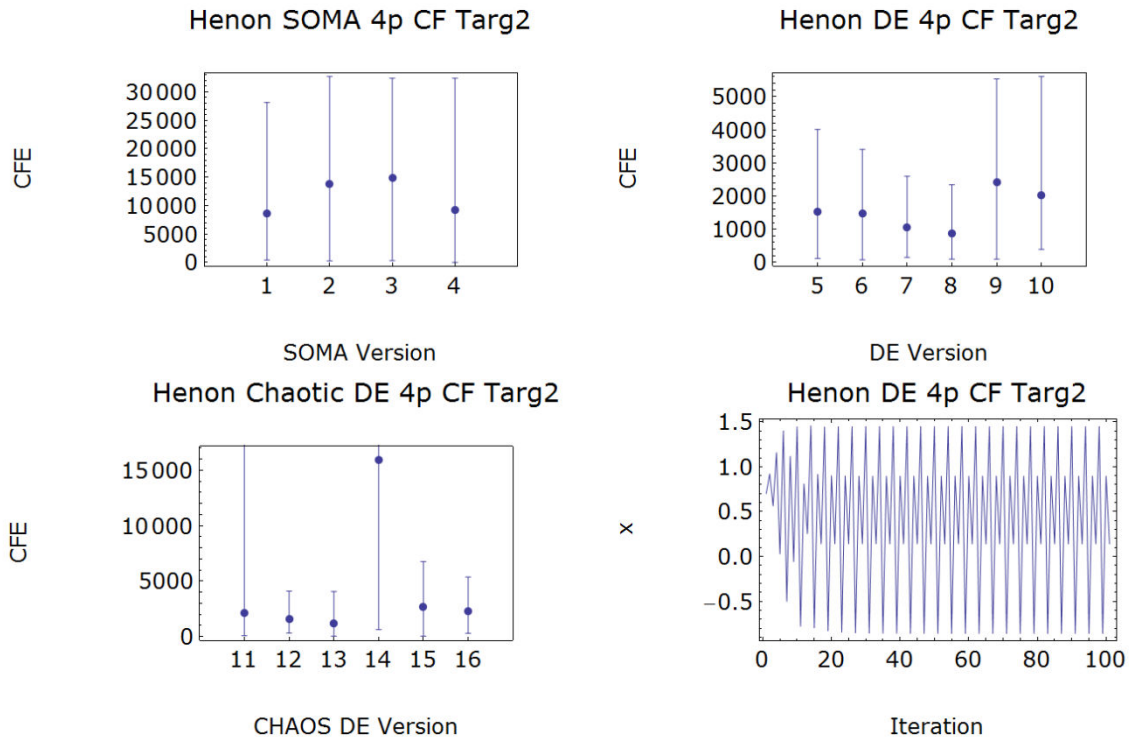
Tabulka 51. Nejlepší řešení Henonovy mapy, p-4 orbit, CF Targ2,Chaotické DE

EA	K	Fmax	R	Min CF Val	Avg CF Val
11	**-0,3549**	**0,2448**	**0,3735**	**$1,9697.10^{-9}$**	**$6,7581.10^{-5}$**
12	-0,3699	0,2158	0,4117	$1,4590.10^{-8}$	$5,0515.10^{-8}$
13	-0,3655	0,2232	0,4014	$1,3925.10^{-8}$	$1,8980.10^{-8}$
14	-0,3810	0,5825	0,4411	$1,9286.10^{-8}$	$3,5547.10^{-5}$
15	-0,3659	0,2025	0,4057	$1,4319.10^{-8}$	$3,0186.10^{-8}$
16	-0,3651	0,2233	0,4008	$1,4053.10^{-8}$	$1,9380.10^{-8}$
EA	Med CF Val	Avg CFE	Min CF Stop	Avg CF Stop	%
11	**$1,4424.10^{-8}$**	**2089,98**	**$1,9739.10^{-7}$**	**$1,4155.10^{-5}$**	**98**
12	$1,6370.10^{-8}$	1611,94	$1,9233.10^{-8}$	$1,4059.10^{-5}$	100
13	$1,6369.10^{-8}$	1208,22	$1,5189.10^{-7}$	$1,3807.10^{-5}$	100
14	$2,0928.10^{-6}$	15958,5	$1,9286.10^{-8}$	$1,1171.10^{-5}$	78
15	$1,6921.10^{-8}$	2698,36	$1,3757.10^{-7}$	$1,2977.10^{-5}$	100
16	$1,6472.10^{-8}$	2282,4	$1,3917.10^{-7}$	$1,1577.10^{-5}$	100

Nejlepší variantou Chaotické Diferenciální evoluce pro nastavení Henonovy mapy, p-4 orbit, CF Targ2 je DERand1Bin. Minimální hodnota účelové funkce je *Min CF Val* =$1,9697.10^{-9}$.

Pro nastavení simulace Henonovy mapy, p-4 orbit, CF Targ2 se jako nejlepší evoluční algoritmus pro řízení chaotického systému ukazovala Diferenciální evoluce ve variantě

DELocalToBest. Tato varianta byla také použita pro zobrazení simulace stabilizace chaotického systému na *obrázku 35* dole vpravo



Obrázek 35. Grafické znázornění úspěšností jednotlivých evolučních algoritmů pro Henonovu mapu, p-4 orbit, CF Targ2

10 ZÁVĚREČNÉ POROVNÁNÍ

Zde byl vytvořen přehledný souhrn výsledků z kapitoly 9. Pro porovnání výkonnosti evolučních algoritmů byly použity tyto hodnoty:

- minimální hodnoty účelové funkce - *Min CF Val*
- minimální hodnota prvního jedince, který je nižší než akceptovatelná hodnota *Min CF Stop*
- průměrná hodnota účelové funkce - *Avg CF*
- počet ohodnocení účelové funkce před prvním jedincem, který je nižší než akceptovatelná hodnota - *Avg CFE*
- počet ohodnocení účelové funkce před prvním jedincem, je nižší než akceptovatelná hodnota, ale neobsazené pozice v souboru Stop nejsou penalizovány na nejvyšší hodnotu - *nekorigovaná Avg CFE*
- procentuální vyjádření, kolik z těch 50 opakovaných optimalizací našlo vůbec nějaké optimální řešení, tedy takové, které bylo nižší než první akceptovatelné - % (poměr velikosti souboru s koncovkou stop a full) - *%*

Každá z šesti tabulek obsahuje výše zmíněné hodnoty a výsledky, které dosáhly jednotlivé evoluční algoritmy. Nejlepší výsledek pro dané nastavení parametrů je zvýrazněn.

10.1 Porovnání výkonnosti EA pro Logickou rovnici

Tabulka 52. Porovnání EA pomocí minimální hodnoty účelové funkce

Min CF Val			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	$1{,}6.10^{-15}$	**$1{,}4.10^{-15}$**	**$1{,}4.10^{-15}$**
CF Targ1 p-2 orbit	$5{,}7915.10^{-13}$	**$2{,}1537.10^{-13}$**	$2{,}2949.10^{-13}$
CF Targ1 p-4 orbit	$5{,}7108.10^{-3}$	**$2{,}322.10^{-3}$**	$3{,}923.10^{-3}$
CF Targ2 p-1 orbit	$1{,}5.10^{-15}$	**$1{,}4.10^{-15}$**	**$1{,}4.10^{-15}$**
CF Targ2 p-2 orbit	$1{,}7685.10^{-14}$	**$1{,}3865.10^{-14}$**	$1{,}4665.10^{-14}$
CF Targ2 p-4 orbit	$1{,}8519.10^{-4}$	$1{,}5221.10^{-4}$	**$1{,}4997.10^{-4}$**

Tabulka 53. Porovnání EA minimální hodnotou prvního akceptovatelného řešení

Min CF Stop			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	**$2{,}1.10^{-15}$**	$2{,}5.10^{-15}$	$2{,}8.10^{-15}$
CF Targ1 p-2 orbit	$1{,}253.10^{-5}$	**$1{,}6758.10^{-12}$**	$1{,}9205.10^{-9}$
CF Targ1 p-4 orbit	$6{,}5754.10^{-3}$	$5{,}8146.10^{-3}$	**$5{,}553.10^{-3}$**
CF Targ2 p-1 orbit	$3{,}7.10^{-15}$	**$3{,}5.10^{-15}$**	$3{,}8102.10^{-15}$
CF Targ2 p-2 orbit	$1{,}9728.10^{-14}$	$2{,}1238.10^{-14}$	**$1{,}3674.10^{-8}$**
CF Targ2 p-4 orbit	$1{,}9919.10^{-4}$	**$1{,}9686.10^{-4}$**	$1{,}9909.10^{-4}$

Tabulka 54. Porovnání EA pomocí průměrné hodnoty účelové funkce

Avg CF			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	$1{,}858.10^{-15}$	**$1{,}4987.10^{-15}$**	$1{,}6503.10^{-15}$
CF Targ1 p-2 orbit	**$1{,}1084.10^{-2}$**	$3{,}3174.10^{-2}$	$3{,}3174.10^{-2}$
CF Targ1 p-4 orbit	1,4686	0,994178	**0,7411**
CF Targ2 p-1 orbit	$1{,}8062.10^{-15}$	**$1{,}4913.10^{-15}$**	$1{,}64.10^{-15}$
CF Targ2 p-2 orbit	**$9{,}4199.10^{-5}$**	$9{,}2889.10^{-4}$	$3{,}109.10^{-3}$
CF Targ2 p-4 orbit	$1{,}5986.10^{-2}$	**$1{,}494.10^{-2}$**	$1{,}6192.10^{-2}$

Tabulka 55. Porovnání EA pomocí počtu ohodnocení účelové funkce

Avg CFE			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	337	**52**	59
CF Targ1 p-2 orbit	15532	**2944**	3946
CF Targ1 p-4 orbit	28831	**19451**	21461
CF Targ2 p-1 orbit	47	**13**	**13**
CF Targ2 p-2 orbit	17859	**3428**	5888
CF Targ2 p-4 orbit	28287	**10628**	16435

Tabulka 56. Porovnání EA pomocí počtu nekorigovaného ohodnocení účelové funkce

Nekorigovaná Avg CFE			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	337	**52**	59
CF Targ1 p-2 orbit	13317	**2746**	3658
CF Targ1 p-4 orbit	17317	**11224**	12787
CF Targ2 p-1 orbit	47	**13**	**13**
CF Targ2 p-2 orbit	15856	**3035**	4683
CF Targ2 p-4 orbit	21028	**9149**	10790

Tabulka 57. Procentuální vyjádření nalezení první akceptovatelné hodnoty

%			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	**100%**	**100%**	**100%**
CF Targ1 p-2 orbit	88,5%	**99,3%**	99%
CF Targ1 p-4 orbit	24,5%	**61,3%**	56%
CF Targ2 p-1 orbit	**100%**	**100%**	**100%**
CF Targ2 p-2 orbit	88%	**98,7%**	95,7%
CF Targ2 p-4 orbit	37%	**93,7%**	74%

V *tabulce 52* vycházejí nejlepší výsledky Diferenciální evoluci. U obou p-1 orbitu vychází stejné hodnoty i pro chaotickou Diferenciální evoluci. Jen v posledním případě CF Targ2 a p-4 orbit byl nalezen lepší výsledek a to pro chaotickou Diferenciální evolucí.

Další tabulka již tak jednoznačná není. Minimální hodnoty ze souboru Stop jsou nejnižší v třech případech z šesti u Diferenciální evoluce.

Průměrná hodnota účelové funkce v *tabulce 54* je nejnižší pro Diferenciální evoluci v obou případech p-1 orbitu a navíc pro p-4 orbit, CF targ2. SOMA byla nejúspěšnější pro oba p-2 orbit a chaotická Diferenciální evoluce byla úspěšná jen pro p-4 orbit, CF Targ1.

Nejmenšího počtu ohodnocení účelové funkce dosáhla pro všechny kombinace parametrů Diferenciální evoluce. To je zobrazeno v *tabulce 55*.

I pro nekorigovanou hodnotu počtu ohodnocení účelové funkce podala nejnižší hodnoty v *tabulce 56* Diferenciální evoluci pro všechny parametry.

Tabulka 57 ukazuje, že pro p-1 orbit jsou všechny evoluční algoritmy stoprocentní. V ostatních případech je opět nejlepší Diferenciální evoluce.

10.2 Porovnání výkonnosti EA pro Henonovu mapu

Tabulka 58. Porovnání EA pomocí minimální hodnoty účelové funkce

Min CF Val			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	$3{,}8.10^{-15}$	$\mathbf{3{,}7.10^{-15}}$	$\mathbf{3{,}7.10^{-15}}$
CF Targ1 p-2 orbit	$7{,}9238.10^{-13}$	$4{,}8176.10^{-13}$	$\mathbf{4{,}7548.10^{-13}}$
CF Targ1 p-4 orbit	$4{,}8612.10^{-7}$	$\mathbf{2{,}2566.10^{-8}}$	$1{,}0067.10^{-7}$
CF Targ2 p-1 orbit	$\mathbf{3{,}7.10^{-15}}$	$\mathbf{3{,}7.10^{-15}}$	$\mathbf{3{,}7.10^{-15}}$
CF Targ2 p-2 orbit	$1{,}4277.10^{-13}$	$\mathbf{1{,}6797.10^{-14}}$	$1{,}6908.10^{-14}$
CF Targ2 p-4 orbit	$1{,}4707.10^{-8}$	$\mathbf{1{,}4666.10^{-9}}$	$1{,}9697.10^{-9}$

Tabulka 59. Minimální hodnota prvního akceptovatelného řešení

Min CF Stop			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	$4{,}9.10^{-15}$	**$4{,}2.10^{-15}$**	$4{,}4.10^{-15}$
CF Targ1 p-2 orbit	$2{,}9445.10^{-2}$	$1{,}2483.10^{-3}$	**$8{,}4281.10^{-6}$**
CF Targ1 p-4 orbit	$4{,}2727.10^{-6}$	$8{,}5494.10^{-6}$	**$8{,}9662.10^{-7}$**
CF Targ2 p-1 orbit	$5{,}6992.10^{-15}$	**$5{,}5.10^{-15}$**	$5{,}8.10^{-15}$
CF Targ2 p-2 orbit	**$1{,}774.10^{-13}$**	$1{,}7331.10^{-9}$	$1{,}6202.10^{-6}$
CF Targ2 p-4 orbit	$1{,}725.10^{-7}$	**$1{,}72.10^{-7}$**	$1{,}9739.10^{-7}$

Tabulka 60. Porovnání EA pomocí průměrné hodnoty účelové funkce

Avg CF			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	$4{,}4295.10^{-15}$	**$4{,}017.10^{-15}$**	$4{,}2353.10^{-15}$
CF Targ1 p-2 orbit	**$9{,}0934.10^{-3}$**	$7{,}92.10^{-2}$	$6{,}1071.10^{-2}$
CF Targ1 p-4 orbit	$1{,}6391.10^{-4}$	**$7{,}656.10^{-6}$**	$2{,}0958.10^{-4}$
CF Targ2 p-1 orbit	$4{,}1948.10^{-15}$	**$3{,}96454.10^{-15}$**	$4{,}1128.10^{-15}$
CF Targ2 p-2 orbit	$1{,}5266.10^{-4}$	**$1{,}0065.10^{-4}$**	$2{,}072.10^{-3}$
CF Targ2 p-4 orbit	$7{,}6268.10^{-6}$	**$5{,}9215.10^{-7}$**	$7{,}070.10^{-6}$

Tabulka 61. Porovnání EA pomocí počtu ohodnocení účelové funkce

Avg CFE			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	2780	**390**	587
CF Targ1 p-2 orbit	9614	**1816**	2407
CF Targ1 p-4 orbit	6432	**1124**	2103
CF Targ2 p-1 orbit	124	**20**	22
CF Targ2 p-2 orbit	15365	**2656**	4327
CF Targ2 p-4 orbit	11595	**1570**	4309

Tabulka 62. Porovnání EA pomocí počtu nekorigovaného ohodnocení účelové funkce

Nekorigovaná Avg CFE			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	2780	**390**	587
CF Targ1 p-2 orbit	9264	**1713**	2306
CF Targ1 p-4 orbit	6432	**1124**	2001
CF Targ2 p-1 orbit	124	**20**	22
CF Targ2 p-2 orbit	14167	**2656**	3946
CF Targ2 p-4 orbit	11057	**1570**	3134

Tabulka 63. Procentuální vyjádření nalezení první akceptovatelné hodnoty

%			
Parametry	SOMA	DE	Chaotická DE
CF Targ1 p-1 orbit	**100%**	**100%**	**100%**
CF Targ1 p-2 orbit	98,5%	**99,6%**	99,7%
CF Targ1 p-4 orbit	**100%**	**100%**	99,7%
CF Targ2 p-1 orbit	**100%**	**100%**	**100%**
CF Targ2 p-2 orbit	93,5%	**100%**	98,7%
CF Targ2 p-4 orbit	97,5%	**100%**	96%

Výsledky vynesené v *tabulce 58* ukazují, že z hlediska minimální hodnoty účelové funkce byla pro Henonovu mapu nejúspěšnější Diferenciální evoluce. Jediným nezdarem se jí stalo nastavení p-2 orbit a CF Targ1 kde byla o jednu desetinu úspěšnější chaotická Diferenciální evoluce.

Tabulka 59 zobrazující minimální hodnotu prvního akceptovatelného jedince stejně jako pro Logickou rovnici nerozhodla ve prospěch ani jednoho evolučního algoritmu. Z šesti nastavení měla opět Diferenciální evoluce tři minimální hodnoty. Chaotická Diferenciální evoluce byla nejúspěšnější pro CF Targ1, p-2 a p-4 orbit.

Na rozdíl od předcházejícího případu, porovnání průměrné hodnoty účelové funkce (*Tabulka 60*) bylo daleko jednoznačnější. Diferenciální evoluce zde byla nejúspěšnější. Jedinou porážku utrpěla pro nastavení parametrů p-2 orbit a CF Targ2 od algoritmu SOMA.

Tabulka 61 zobrazuje průměrný počet ohodnocení. Na rozdíl od Logické rovnice je pro Henonovu mapu nejúspěšnějším algoritmem Diferenciální evoluce.

Nekorigovaný počet ohodnocení účelové funkce je zobrazen v *tabulce 62*. Stejně jako v předchozím případě i tady byla nejúspěšnější Diferenciální evoluce.

Procentuální nalezení první akceptovatelné hodnoty dopadlo opět nejlépe pro Diferenciální evoluci. Algoritmus SOMA byl stoprocentní ve třech případech. Chaotická Diferenciální evoluce byla stoprocentní u obou p-1 orbitu. Nicméně je nutné podotknout, že žádná hodnota neklesla pod 93%

ZÁVĚR

Hlavním úkolem této práce bylo nalezení optimálního nastavení vybraných evolučních algoritmů a jejich následné použití na problém optimalizace řízení deterministického chaosu. Šlo konkrétně o tyto evoluční algoritmy:

- SOMA
- Diferenciální evoluce
- Chaotická Diferenciální evoluce

Dále byly vybrány dva diskrétní chaotické systémy:

- Logická rovnice
- Henonova mapa

Sofistikovaný problém optimalizaci řízení chaotického systému jsme využili pro porovnání výkonnosti jednotlivých algoritmů. Pro test jsme nadefinovali účelové funkce *Targeting CF Targ1* a *CF Targ2* a to vše ještě pro tři typy *UPOs*. Dohromady tedy bylo nadefinováno 12 případových studií pro 4 strategie algoritmu SOMA a 6 variant klasické a Chaotické Diferenciální evoluce. Pro nastavení řídících parametrů bylo použito „meta“ přístupu, neboli nalezení nejlepší kombinace řídících parametrů samotnými evolučními algoritmy. Pro optimalizaci byl využit nejsložitější případ Henonovy mapy, *p-4 orbit* a účelové funkce *CF Targ2*. Těmito optimalizovanými parametry byly potom nastaveny evoluční algoritmy a spuštěno 50 simulací pro každou případovou studii.

Cíle, které byly vytyčeny v zadání, byly splněny až na algoritmus rojení částic (PSO). Algoritmus PSO, který byl naprogramován v prostředí Mathematica 7.0 jež byl vytvořen v rámci jiné diplomové práce, a byl cca 5 krát pomalejší než algoritmy, které byly k dispozici pro SOMU, Diferenciální evoluci a Chaotickou Diferenciální evoluci. Na vině byly nevhodné programové instrukce pro prostředí Mathematica 7.0. PSO bylo vytvořeno pomocí spousty vnořených cyklů *For* doplněného spoustou podmínek *If*. Mathematica 7.0 ovšem disponuje daleko rychlejšími příkazy jako je např. *Table* a mapováním datových struktur, kterými byly vytvořeny algoritmy pro Diferenciální evoluci, Chaotickou Diferenciální evoluci a algoritmus SOMA. První testovací simulace pro nalezení řídících parametrů pro algoritmus SOMA a oba typy Diferenciální evoluce běžely cca. 6 dní na jednom čtyř-jádrovém Xserveru. Samotná simulace optimalizace řízení deterministického chaosu běžela dalších 6 dní. Samotný algoritmus rojení částic PSO by dodal výsledky za

48 dní. Navíc Xservery byly obsazeny i jinými vědeckými výpočty a simulacemi. Předmětem této práce ale nebyla optimalizace kódu PSO, nýbrž posouzení výkonnosti výše uvedených evolučních algoritmů.

Zbylé evoluční algoritmy byly porovnány podle šesti hledisek opírajících se o statistické výsledky rozsáhlých 50 cyklů simulace.

Pro testování optimalizace řízení chaotického systému reprezentovaného Logickou rovnicí vycházely nejlépe výsledky pro klasickou Diferenciální evoluci. Ta nacházela jednoznačně lepší minimální hodnoty obou účelových funkcí. Přitom nacházela přijatelné výsledky nejrychleji ze všech evolučních algoritmů, což dokládaly nejmenší hodnoty ohodnocení účelové funkce. I v procentuálním vyjádření nalezení prvního akceptovatelného řešení neklesla hodnota pod 90%, s výjimkou nejsložitějšího příkladu *p-4 orbit* pro účelovou funkci *CF Targ1* které bylo na hodnotě 63,1% a i tak byl nejlepším výsledkem ze všech evolučních algoritmu při tomto nastavení. Jen pro porovnání průměrné hodnoty účelové funkce a pro minimální hodnotu prvního akceptovatelného řešení byly testy nejednoznačné, kdy se jednotlivé evoluční algoritmy střídaly v úspěšnosti při jednotlivých nastaveních parametrů chaotického systému a účelové funkce.

Druhým testovaným chaotickým systémem byla Henonova mapa. Stejně jako pro předcházející systém podávala nejlepší výsledky Diferenciální evoluce. Nacházela jednoznačně nejlepší minimální hodnoty účelových funkcí, ke kterým se u tohoto chaotického systému přidala i průměrná hodnota účelové funkce ze všech cyklů. Jediným nastavením kde nebyla Diferenciální evoluce úspěšná *p-2 orbit* a *CF Targ1*. Navíc byla jednoznačně nejrychlejší ze všech algoritmů, což dokládala nejmenší počet ohodnocení účelové funkce. Procentuálně je Diferenciální evoluce pro nalezení prvního akceptovatelného řešení skoro stoprocentní, jen pro případ zmiňovaný u minimální hodnoty a průměrné hodnoty účelové funkce je o 0,4% degradován.

Testování v této práci vyšlo nejlépe pro Diferenciální evoluci. Neznamená to ovšem, že je to nejvýkonnější algoritmus, ale že pro námi zvolenou podmnožinu problému optimalizace řízení deterministického chaosu vykazoval nejlepší výsledky. Nutno podotknout, že ostatní evoluční algoritmy pracovaly velmi uspokojivě.

ZÁVĚR V ANGLIČTINĚ

The main task of this thesis was to find an optimal set of selected evolutionary algorithms and their subsequent application to the problem of optimization of deterministic chaos. It was specifically about the evolutionary algorithms:

- SOMA
- Differential evolution
- Chaotic Differential evolution

Sophisticated problem of optimization of a chaotic system control we use to compare performance of different algorithms. For the test we have defined cost function *Targeting Targ1 CF* and *CF Targ2* and yet for all three types *UPOs*. Altogether it was defined 12 case studies for 4 strategies of algorithms SOMA and 6 variants of the classic and chaotic differential evolution. To set the control parameters were used "meta" approach, which is to find the best combination of control parameters of evolutionary algorithms themselves. Optimization was used for the most complicated case Henon map, p-4 orbit and the cost function CF Targ2. These optimized parameters were then adjusted evolutionary algorithms, and running 50 simulations for each case study.

Objectives that were set in the task have been met to the particle swarm algorithm (PSO). PSO algorithm, which has been programmed in Mathematica 7.0 environment that was created in another thesis, and was about 5 times slower than the algorithms that were available for SOMA, chaotic differential evolution and differential evolution. To blame were unsuitable programming instructions for use Mathematica 7.0. PSO was developed through a lot of nested loops *For*, supplemented by many conditions *If*. Mathematica 7.0 has much faster, has commands such as *Table* and mapping of data structures, which were created for algorithms of the Differential evolution, Chaotic Differential evolution and SOMA. The first simulation test for finding the control parameters for the SOMA algorithm and Differential evolution both types running around 6 days on a single four-core Xserver. Mere simulation optimization of deterministic chaos ran another 6 days. Particle swarm algorithm PSO itself was calculated the results for 48 days. Moreover Xserver were occupied by other scientific calculations and simulations. The goal of this work wasn’t to optimize the code PSO, but the performance of efficiency evaluation of evolutionary algorithms.

The remaining evolutionary algorithms were compared by the six criteria which are based on statistical results of extensive simulations of 50 cycles.

For testing, optimization of a chaotic system represented by Logistic equation reached the best results for classical differential evolution. She clearly found a better minimum value of both cost functions. Yet found acceptable results which were faster of all possible evolutionary algorithms, as evidenced by a minimum of cost function evaluations. Even in percentage terms to find the first acceptable solution value isn't reduced below 90%, with the exception of the most difficult example of p-4 orbit for the cost function CF Targ1 which was 63.1% in value and it was the best result of all evolutionary algorithms in this setting. Just to compare the average value of cost function and the minimum value of the first acceptable solution tests were inconclusive, the individual evolutionary algorithms rotate into success for each parameter chaotic system and the cost function.

The second chaotic system was tested Henonova map. Like the previous system to report the best results of Differential evolution. Clearly found the best minimum value of cost functions that are in this chaotic system has added the average cost function of each cycle. The only setting where the Differential evolution has not been successful was *p-2 orbit* and *CF Targ1*. Furthermore, it was clearly the fastest of all algorithms, which demonstrated the smallest number of cost function evaluations. Differential evolution is the percentage of first finding an acceptable solution almost one hundred percent. Just in case the aforementioned minimum value and average cost function value is degraded by 0.4%.

Testing of this work was best for the differential evolution. That does not mean that this is the most efficient algorithm, but for that we selected a subset of the problem of optimization of deterministic chaos, showed the best results. Needless to say that other evolutionary algorithm worked very satisfactorily.

SEZNAM POUŽITÉ LITERATURY

[1] SCHUSTER, H. G. *Handbook of Chaos Control*. 2. Rev. Edition. Villey VCH, 2007. 849s. ISBN 978/3527406050.

[2] GONZALES-MIRANDA, J. M., *Synchronization and Control of Chaos*: An Introduction for Scientists an Engineers. World Scientific Publishing Company, 2004. 224 s. ISBN 978-1860944888.

[3] ZELINKA, Ivan. *Aplikovaná Informatika*. Zlín. UTB, 1999. 183 p. ISBN 80-214-1423-5.

[4] ZELINKA, Ivan. *Umělá inteligence v problémech globální optimalizace*. BEN, 2002, 190 p. ISBN 80-7300-069-5.

[5] ZELINKA, I., OPLATKOVÁ, Z., OŠMERA, P., ŠEDA, M., VČELAŘ, F. *Evoluční výpočetní techniky - principy a aplikace*. BEN - technická literatura, Praha, 2008, ISBN 80-7300- 218-3.

[6] KVASNIČKA, V., POSPÍCHAL, J., TIŇO, P., *Evolučné algoritmy*. Bratislava: STU Press, 2000.

[7] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: *Umělá inteligence*, Academia, 1993, ISBN 80-200-0496-3.

[8] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: *Umělá inteligence 4*., Academia, 2003, ISBN 80-200-1044-0.

[9] ŠENKEŘÍK, Roman. *Optimal Control of Deterministic Chaos*. Zlín, 2008. 316 s. Dizertační práce. Univerzita Tomáše Bati, Fakulta aplikované informatiky. ISBN 978-80-7318-783-5.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

a	Parametr Henonovy mapy
AllToAll	Strategie algoritmu SOMA všichni ke všem
AllToAllAdaptive	Strategie algoritmu SOMA adaptivně všichni ke všem
AllToOne	Strategie algoritmu SOMA všichni k jednomu
AllToOneRand	Strategie algoritmu SOMA všichni k jednomu náhodně
AS	Aktuální stav účelové funkce Targeting CF
b	Parametr Henonovy mapy
c_1	Učící faktor algoritmu PSO
c_2	Učící faktor algoritmu PSO
CC_i	Parametr účelové funkce Leadera pro strategii svazky algoritmu SOMA
CF Targ1	Účelová funkce chaotického systému
CF Targ2	Účelová funkce chaotického systému
Cluster	Strategie algoritmu SOMA svazky
CR	Řídící parametr Diferenciální evoluce určující práh křížení,
D	Počet argumentů účelové funkce
DE	Diferenciální evoluce
F	Mutační konstanta Diferenciální evoluce
F_{max}	parametr pro ETDAS
F_n	Perturbace Logické rovnice
gBest	Funkce algoritmu PSO pro hodnotu nejlepšího jedince
Generace	Ukončovací parametr Diferenciální evoluce
HB_i	Parametr vrchní meze pro strategii Svazky algoritmu SOMA
H_i	Hodnota vrchní hranice intervalu
IND_i	Parametr účelové funkce pro strategii svazky algoritmu SOMA

K	parametr pro ETDAS
LB_i	Parametr spodní meze pro strategii Svazky algoritmu SOMA
L_o	Hodnota spodní hranice intervalu
M	Počet jedinců pro populaci
Migrace	Ukončovací parametr algoritmu SOMA určující počet generací
MinDiv	Ukončovací parametr algoritmu SOMA
N	Počet argumentů optimalizované funkce
NP	Řídící parametr Diferenciální evoluce určující počet jedinců
pBest	Funkce algoritmu PSO pro hodnotu nejlepší pozice aktuálního jedince
PathLength	Řídící parametr algoritmu SOMA určující délku cesty aktivního jedince
PopSize	Řídící parametr algoritmu SOMA určující počet jedinců
PSO	Particle swarm optimalization
PRT	Řídící parametr algoritmu SOMA určující perturbaci
r	Parametr Logické rovnice
R	parametr pro ETDAS
seeds	Vedoucí jedinec skupiny algoritmu PSO
SOMA	Self-Organizing Migration Algorithm
Step	Řídící parametr algoritmu SOMA určující krok
TS	Cílový stav účelové funkce Targeting CF
v	Šumový vektor Diferenciální evoluce
Vmax	Maximální rychlost částic algoritmu PSO
w	Váha setrvačnosti algoritmu PSO
w_{end}	Koncová hodnota setrvačnosti algoritmu PSO
w_{start}	Počáteční hodnota setrvačnosti algoritmu PSO
χ	Constriction faktor algoritmu PSO

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: ADRESÁŘOVÁ STRUKTURA PŘILOŽENÉHO CD

Pro lepší orientaci v adresářové struktuře přiloženého CD, přikládám tento stručný popis

- **\Data** - kořenový adresář pro všechny exportovaná data
- **\Data\Optimalizace EA** - obsahuje soubory s daty pro optimalizaci parametrů evolučních algoritmů
- **\Data\Optimalizace Chaos -** obsahuje soubory s daty pro optimalizaci parametrů chaotických systémů
- **\Diplomová práce** – obsahuje tuto práci ve formátu .pdf a .docx
- **\Obrázky** – obsahuje všechny obrázky vytvořené pro tuto práci ve formátu .png
- **\Simulace** – kořenový adresář pro všechny vytvořené simulace
- **\Simulace\Parametry EA** – obsahuje zdrojové kódy vytvořené v prostředí Mathematica 7.0. DataParametru, TestDE, TestSOMA, Zpracovani DE a Zpracovani SOMA.nb
- **\Simulace\Testovani EA** - obsahuje zdrojové kódy vytvořené v prostředí Mathematica 7.0 Zpracovani Final Henon, Zpracovani Final LEQ a soubory s jednotlivými simulacemi nazvané podle kombinace parametrů pro které simulaci počítali.