

Knihovna evolučních optimalizačních algoritmů v prostředí Java

The library of evolutionary optimization algorithms in Java

Bc. Tomáš Král

Diplomová práce
2011



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2010/2011

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš KRÁL**
Osobní číslo: **A09478**
Studijní program: **N 3902 Inženýrská informatika**
Studijní obor: **Informační technologie**

Téma práce: **Knihovna evolučních optimalizačních algoritmů
v prostředí Java**

Zásady pro vypracování:

1. Seznamte se evolučními technikami optimalizace.
2. Naprogramujte vybrané evoluční algoritmy v Java v podobě knihovny, do které bude v budoucnu možnost přidávat další evoluční algoritmy.
3. Otestujte na vybraných testovacích problémech.
4. Zpracujte závěr.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. 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.
2. ZELINKA, I. Umělá inteligence I. VUT Brno, 1998, ISBN 80-214-1163-5.
3. KVASNIČKA, V., POSPÍCHAL, J., TIŇO, P., Evolučné algoritmy. Bratislava : STU Press, 2000, ISBN 80-227-1377-5.
4. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence 4., Academia, 2003, ISBN 80-200-1044-0.
5. DARWIN F.: Java: Kuchařka programátora, Computer Press, 2006, ISBN: 80-251-0944-5.
6. KISZKA B.: 1001 tipů a triků pro jazyk Java, Computer Press, 2009, ISBN: 978-80-251-2467-3.

Vedoucí diplomové práce:

Ing. Zuzana Oplatková, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

24. února 2011

Termín odevzdání diplomové práce:

18. května 2011

Ve Zlíně dne 24. února 2011



prof. Ing. Vladimír Vašek, CSc.
děkan



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

ABSTRAKT

Tato diplomová práce se zabývá problematikou evolučních algoritmů, a to se zaměřením na Diferenciální evoluci a SOMA. Cílem práce je vytvořit přehledný interpreter a menší knihovnu s evolučními algoritmy a testovacími funkcemi. Jsou kladeny velké nároky na jednoduché přidávání dalších evolučních algoritmů do aplikace.

V teoretické části je čtenář uveden do problematiky evolučních algoritmů, jejich historie a funkcionality. Praktická část zahrnuje popis grafického rozhraní, jednotlivých tříd v programu, návod jak implementovat novou třídu s evolučním algoritmem do programu a testování již implementovaných evolučních algoritmů.

Klíčová slova:

evoluce, optimalizace, evoluční algoritmus, Diferenciální evoluce, SOMA, knihovna, Java

ABSTRACT

This thesis deals with evolutionary algorithms, focusing on the differential evolution and SOMA. The aim of this work is to create a clear interpreter and a library with a library of algorithms and a library of test functions. They placed great requirements on the simple addition of evolutionary algorithms to the application.

In the theoretical part of the reader set of evolutionary algorithms in their history, functionality. The practical part includes a description of the graphical interface, each class in the program, instructions on how to implement a new class of evolutionary algorithms in the program and testing evolutionary algorithms already implemented.

Keywords:

evolution, optimization, evolutionary algorithm, Differential Evolution, SOMA, library, Java

Na tomto místě bych chtěl poděkovat své vedoucí diplomové práce Ing. Zuzaně Oplatkové, Ph.D., její rady a připomínky mě vždy posouvaly o krok vpřed.

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	10
I TEORETICKÁ ČÁST	11
1 POPIS EVOLUČNÍCH ALGORITMŮ	12
1.1 PŘIROZENÝ VÝBĚR CHARLESE DARWINA	12
1.2 HISTORIE	12
1.3 OPTIMALIZAČNÍ ALGORITMY	13
1.4 DĚLENÍ OPTIMALIZAČNÍCH ALGORITMŮ	13
1.4.1 Enumerativní	13
1.4.2 Deterministické	14
1.4.3 Stochastické.....	15
1.4.4 Smíšené	15
1.5 SPOLEČNÉ RYSY EVOLUČNÍCH ALGORITMŮ	16
1.5.1 Jednoduchost	16
1.5.2 Hybridnost.....	16
1.5.3 Používání dekadických čísel	16
1.5.4 Rychlost	17
1.5.5 Schopnost nalézt jehlu v kupce sena	17
1.5.6 Schopnost dát vícenásobná řešení	17
1.6 SLOŽITOST	18
1.7 TERMINOLOGIE	19
1.7.1 Účelová funkce.....	19
1.7.2 Hodnota účelové funkce	20
1.7.3 Prostor řešení.....	20
1.7.4 Populace.....	20
1.7.5 No Free Lunch Teorém.....	21
2 SOMA	22
2.1 PARAMETRY.....	22
2.1.1 PathLength	22
2.1.2 Step	23
2.1.3 D	23
2.1.4 PRT	23
2.1.5 PopSize	24
2.1.6 Migrace	24
2.1.7 MinDiv	24
2.2 PRINCIP SOMA.....	24

3	DIFERENCIÁLNÍ EVOLUCE	27
3.1	PARAMETRY.....	27
3.1.1	CR.....	27
3.1.2	D	27
3.1.3	NP	27
3.1.4	F.....	28
3.1.5	Generations	28
3.2	PRINCIP DIFERENCIÁLNÍ EVOLUCE	28
4	POUŽITÉ TESTOVACÍ PROBLÉMY.....	31
II	PRAKTICKÁ ČÁST	34
5	POPIS PROGRAMU	35
5.1	POPIS GRAFICKÉHO ROZHRAŇÍ	35
5.2	POPIS TŘÍD	39
5.2.1	Třída Main.....	39
5.2.2	Třídy testovacích funkcí	39
5.2.3	Třída GUI.....	39
5.2.4	Třídy optimalizačních algoritmů	40
5.3	PŘIDÁNÍ EVOLUČNÍHO ALGORITMU	40
5.4	VLÁKNA	42
6	POROVNÁNÍ IMPLEMENTOVANÝCH EVOLUČNÍCH ALGORITMŮ	43
6.1	UKÁZKA ZÁVISLOSTI EVOLUČNÍHO ALGORITMU DE/RAND/1 NA VSTUPNÍCH PARAMETRECH, TESTOVACÍ FUNKCE 1ST DE JONG	43
6.1.1	Hledání parametru CR	43
6.1.2	Hledání parametru F	45
6.2	UKÁZKA ZÁVISLOSTI EVOLUČNÍHO ALGORITMU DE/RAND/1 NA VSTUPNÍCH PARAMETRECH, TESTOVACÍ FUNKCE SCHWEFEL	47
6.2.1	Hledání parametru CR	47
6.2.2	Hledání parametru F	49
6.3	UKÁZKA ZÁVISLOSTI EVOLUČNÍHO ALGORITMU SOMA ALLTOONE NA VSTUPNÍCH PARAMETRECH, TESTOVACÍ FUNKCE 1ST DE JONG	51
6.3.1	Hledání parametru PRT	51
6.3.2	Hledání parametru PathLength.....	53
6.3.3	Hledání parametru Step	55
6.4	UKÁZKA ZÁVISLOSTI EVOLUČNÍHO ALGORITMU SOMA ALLTOONE NA VSTUPNÍCH PARAMETRECH, TESTOVACÍ FUNKCE SCHWEFEL	57
6.4.1	Hledání parametru PRT	58
6.4.2	Hledání parametru PathLength.....	60
6.4.3	Hledání parametru Step	62
6.5	SHRNUTÍ VÝSLEDKŮ TESTOVÁNÍ	65

ZÁVĚR	66
ZÁVĚR V ANGLIČTINĚ.....	67
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	69
SEZNAM OBRÁZKŮ.....	70
SEZNAM TABULEK	71
SEZNAM GRAFŮ.....	72
SEZNAM PŘÍLOH	73

ÚVOD

Úkolem této diplomové práce bylo naprogramovat v jazyce Java knihovnu s několika vybranými evolučními optimalizačními algoritmy a interpreter, který bude s touto knihovnou pracovat a nakonec otestovat funkčnost vybraných algoritmů. Program musí být napsán takovým způsobem, aby do něj bylo později možné co nejnlehčeji přidávat další evoluční algoritmy.

Teoretická část se zabývá nejprve dělením optimalizačních algoritmů na Enumerativní, Deterministické, Stochastické a Smíšené. Dále popisuje společné rysy evolučních algoritmů, jako jsou jednoduchost nebo rychlost. A jsou zde také popsány některé základní pojmy, které se používají v souvislosti s evolučními algoritmy. Další dvě kapitoly popisují evoluční algoritmy, které byly implementovány do výsledného programu. Definují vstupní parametry, se kterými algoritmy pracují a popisují princip jejich činnosti. Těmito algoritmy jsou SOMA a Diferenciální Evoluce. Teoretickou část uzavírá kapitola zabývající se použitými testovacími funkcemi.

Praktická část je rozdělena na dvě kapitoly, popis programu a porovnání implementovaných evolučních algoritmů. V první kapitole nejprve popisuje rozvržení jednotlivých ovládacích prvků v okně uživatelského grafického rozhraní, a poté stručně popisuje funkce jednotlivých tříd, ze kterých se program skládá, je to z toho důvodu, aby programátor, který by chtěl přidat další evoluční algoritmus, nebo jinak rozšířit stávající aplikaci, měl přehled, k čemu která třída slouží. Dále je zde popsáno, co musí třída s nově přidaným evolučním algoritmem obsahovat a které funkce musí volat, aby prvky jako například graf konvergence nebo indikátor průběhu, fungovaly správně. První kapitolu praktické části uzavírá vysvětlení, proč je aplikace napsána jako dvouvláknová a jakým způsobem jsou jednotlivá vlákna v aplikaci použita. Druhá kapitola praktické části se zabývá testováním dvou implementovaných evolučních algoritmů. Těmito algoritmy jsou Diferenciální evoluce, přesněji algoritmus DE/Rand/1, a SOMA v podobě AllToOne.

I. TEORETICKÁ ČÁST

1 POPIS EVOLUČNÍCH ALGORITMŮ

Evoluční algoritmy se inspirovaly přírodou a přírodními vědami. Jejich snahou je napodobit evoluci, která se probíhá v přírodě. Jejich filozofie se zakládá na teorii přirozeného výběru napsanou Charlesem Darwinem.

1.1 Přirozený výběr Charlese Darwina

Charles Robert Darwin byl britský přírodovědec žijící v letech 1809-1882. V letech 1831-1936 se plavil na lodi Beagle Jejího Veličenstva. Během této plavby Darwin sesbíral a shromáždil cenný přírodovědný materiál, který uveřejnil ve vědecké práci O Původu druhů, přírodním výběrem.

Charles Darwin byl jako první schopen podat vysvětlení o proměnlivosti druhů. Na tuto myšlenku ho přivedl kněz Thomas Malhaus, ve své knize Pojednání o zákonitostech populace. Uveřejňuje zde zjištění že všechny živé organismy mají v průměru více než dva potomky. Přes tuto skutečnost však objevil, že počet jedinců mezi generacemi je relativně stabilní.

Malhausovo dílo pomohlo Darwinovy pochopit, že mezi potomky existuje určitý výběr, kdo se dožije dospělosti a rozmnoží, a kdo zahyne. Tuto skutečnost Darwin pojmenoval jako Přírodní výběr. Jednotliví potomci se od sebe lehce liší drobnými mutacemi. A ti jedinci, jejichž mutace jim pomůže lépe shánět potravu, nebo zvýší šanci na uniknutí predátorovy, mají větší šanci, že se rozmnoží. Tato schopnost přežít se nazývá fitness.

Neboli, větší, silnější a chytřejší jedinci mají potomky a ti menší, slabší a hloupější ne. Příroda vybírá ty jedince, kteří jsou schopnější přežít v nelítostném prostředí, stejně jako člověk značným způsobem ovlivňuje, vlastnosti domácích zvířat za pomoci umělého výběru, tedy rozmnožováním nejlepších jedincům.

1.2 Historie

Mechanismy evoluce, které vyvinula příroda, lze s velkými úspěchy aplikovat také na technické problémy. Takto aplikované mechanismy nazýváme Evoluční Algoritmy (EA). Počátky využití řešení technických problémů za pomoci evolučních algoritmů se datují do 50 až 60 let dvacátého století, kdy se objevilo zhruba deset na sobě nezávislých projektů.

1.3 Optimalizační algoritmy

Optimalizační algoritmy slouží k nalezení minima dané účelové funkce tak, že hledají optimální numerickou kombinaci jejich argumentů. Tyto algoritmy můžeme je rozdělit například podle principu jejich činnosti, složitosti atp. Tato rozdělení nejsou samozřejmě jediná možná, nicméně vzhledem k tomu, že vcelku dobře vystihují současný stav, lze je brát jako jeden z možných pohledů na klasické, ale i moderní optimalizační metody. Názory na jejich zařazení se mírně liší. Lze se setkávat s tvrzeními, že například takzvané simulované žihání nepatří do evolučních technik, což je do jisté míry pravdou. Na druhou stranu je rovněž jinými „evolucionisty“ tvrzeno, že simulované žihání do evolučních technik patří přinejmenším jako jejich přímý předchůdce. Faktem je, že pokud vezmeme v úvahu simulované žihání s elitismem, pak by se tato varianta mohla brát jako evoluční algoritmus.

1.4 Dělení optimalizačních algoritmů

Jednotlivé třídy algoritmů představují obecně způsoby řešení daného problému metodami s různým stupněm efektivity a složitosti. Podle jejich vlastností dělíme algoritmy do těchto kategorií:

- Enumerativní
- Deterministické
- Stochastické
- Smíšené

1.4.1 Enumerativní

Algoritmus provede výpočet všech možných kombinací řešení daného problému. Tento přístup je vhodný pro problémy, u nichž jsou argumenty účelové funkce diskrétního charakteru a nabývají malého množství hodnot. Pokud by byl použit obecně, zcela reálně by mohl potřebovat na úspěšné ukončení čas, který je delší než existence našeho vesmíru.

1.4.2 Deterministické

Tato skupina algoritmů je postavena pouze na rigorózních metodách klasické matematiky. Algoritmy tohoto charakteru obvykle vyžadují omezující předpoklady, které těmto metodám umožňují podávat efektivní výsledky.

Tyto předpoklady obvykle jsou, že:

- Problém je lineární
- Problém je konvexní
- Prohledávaný prostor možných řešení je malý
- Prohledávaný prostor možných řešení je souvislý
- Účelová funkce je, pokud možno, unimodální (má pouze jeden extrém)
- Mezi parametry účelové funkce nejsou nelineární interakce
- Jsou dostupné informace o gradientu apod.
- Problém je definován v analytickém tvaru

Výsledkem deterministického algoritmu je pak jediné řešení

1.4.3 Stochastické

Algoritmy tohoto typu jsou založeny na využití náhody. Jde v podstatě o čistě náhodné hledání hodnot argumentů účelové funkce s tím, že výsledkem je vždy to nejlepší řešení, jež bylo nalezeno během celého náhodného hledání.

Algoritmy tohoto typu jsou obvykle:

- Pomalé
- Vhodné jen pro malé prohledávané prostory možných řešení (malý rozsah argumentů účelové funkce)
- Vhodné pro hrubý odhad

1.4.4 Smíšené

Algoritmy této třídy představují „rafinovanou“ směs metod deterministických a stochastických, které ve vzájemné spolupráci dosahují překvapivě dobrých výsledků. Poměrně silnou podmnožinou těchto algoritmů jsou již zmiňované evoluční algoritmy.

Algoritmy smíšeného charakteru jsou:

- Robustní, což znamená, že nezávisle na počátečních podmínkách velmi často najdou kvalitní řešení, jež bývá reprezentováno obvykle jedním či více globálními extrémy
- Efektivní a výkonné. Slova efektivní a výkonné zde znamenají, že jsou schopny nalézat kvalitní řešení během relativně malého počtu ohodnocení účelové funkce.
- Jsou odlišné od čistě stochastických metod (díky současné přítomnosti deterministických postupů)
- Mají minimální nebo žádné požadavky na předběžné informace
- Jsou schopné pracovat s problémy typu „černá skříňka“, to znamená, že nepotřebují ke své činnosti analytický popis problému
- Jsou schopny najít více řešení během jednoho spuštění

1.5 Společné rysy evolučních algoritmů

Evoluční algoritmy se snaží za pomoci využití modelů evolučních procesů nalézt řešení náročných a rozsáhlých úloh. Evoluční algoritmy mají některé společné vlastnosti:

- Jednoduchost
- Hybridnost
- Používání dekadických čísel
- Rychlost
- Schopnost nalézt jehlu v kupce sena
- Schopnost dát vícenásobná řešení

1.5.1 Jednoduchost

Tyto algoritmy lze často jednoduše naprogramovat.

1.5.2 Hybridnost

Schopnost pracovat s naprosto nesourodou množinou možných stavů, jako je například boolean, real, integer, různé číselné soustavy (hexadecimální...), často také pouze jednotlivé hodnoty. Takže taková výsledná množina může vypadat například takto: {1; 68; -14,5; π ; 1024; A; zelená; false}

1.5.3 Používání dekadických čísel

Není třeba pracovat s čísly v binárním tvaru. Tímto odpadá zkreslení způsobené nedostatečnou délkou binárního řetězce a také skokovou změnu způsobenou jeho mutací. Problém s mutací binárního řetězce jasně ukazuje tento příklad: Mezi čísly 7 a 8 je v dekadické soustavě pouze jedno překlopení zatímco v binární soustavě je mezi těmi samými čísly 0111 a 1000 je potřeba invertovat všechny čtyři bity.

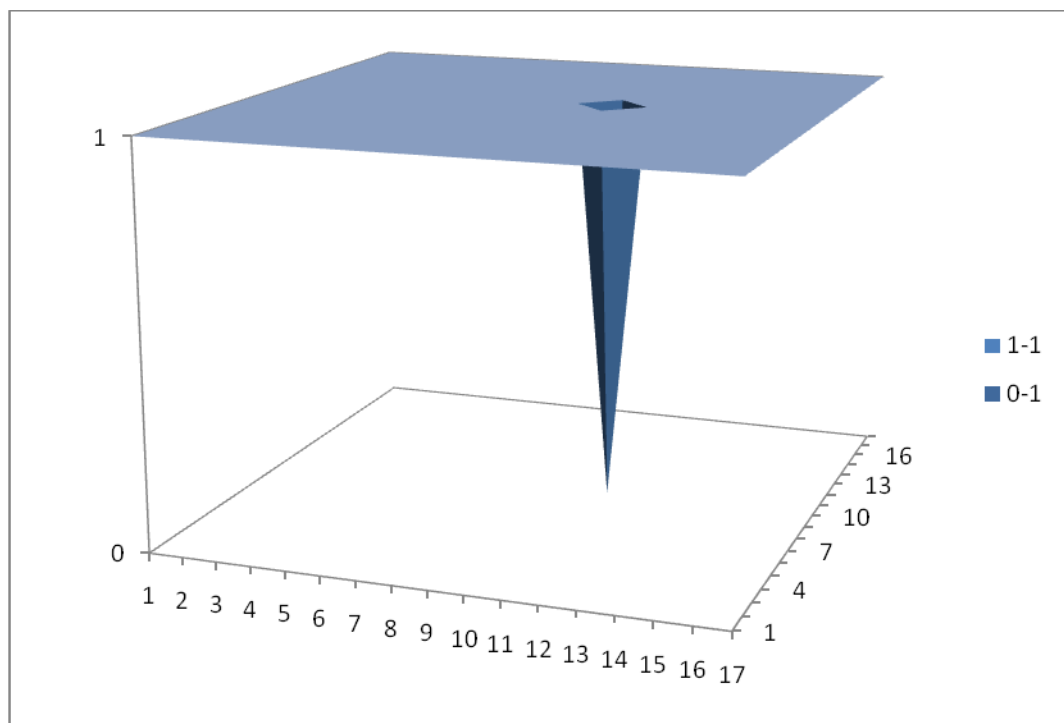
Tento problém částečně řeší takzvané Greyovo kódování, avšak stále je práce s dekadickými čísly výhodnější.

1.5.4 Rychlost

Díky menší výpočetní náročnosti oproti jiným metodám, jako jsou například enumerativní metody, lze říci, že evoluční algoritmy naleznou řešení mnohem rychleji než klasické metody.

1.5.5 Schopnost nalézt jehlu v kupce sena

Některé funkce mají takový tvar, že je velmi těžké, někdy až nemožné odhadnout, kterým směrem se nachází globální minimum, případně maximum. Záleží na charakteru hledaného optima. Schopnost algoritmů nalézt optimum u takovýchto funkcí je velice nízká. Takovouto funkci ukazuje Graf.1.



Graf. 1 Funkce typu „hledání jehly v kupce sena“

1.5.6 Schopnost dát vícenásobná řešení

Evoluční algoritmy jsou většinou nastaveny tak, aby dávaly jenom jedno nejlepší řešení. Jednoduchou úpravou však lze vybrat například pět nejlepších řešení a uživatel si již může sám vybrat, které řešení mu nejvíce vyhovuje.

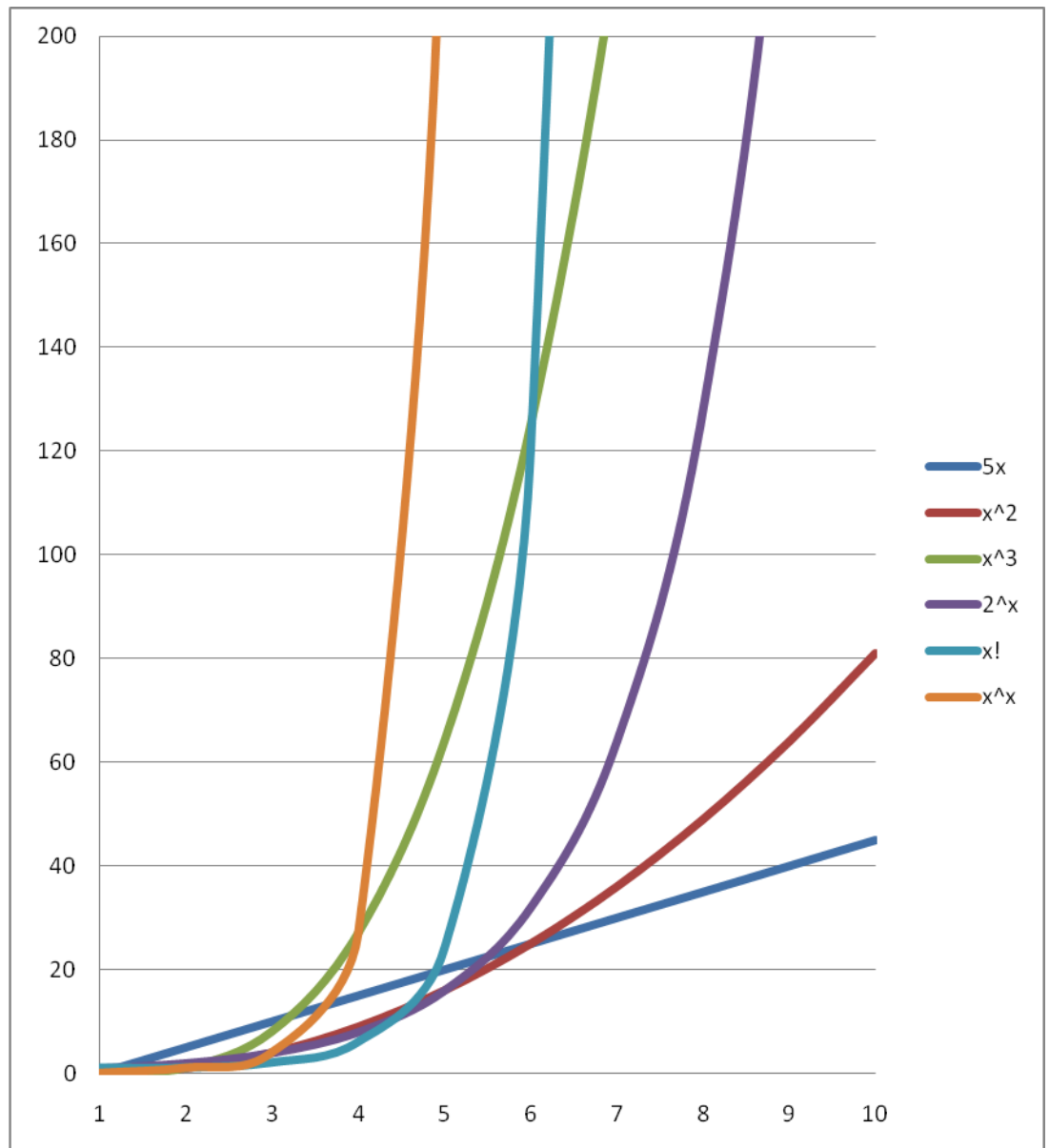
1.6 Složitost

Složitost se v teoretické informatice nemyslí přímo časová náročnost, i když de facto v konečném výsledku ano, ale především závislost potřebného výkonu na rostoucím počtu vstupních údajů. U drtivé většiny problémů roste jejich složitost nelineárně s lineárně rostoucím vstupem. V takovémto případě mluvíme o složitostech polynomiálních případně exponenciálních. Tabulka Tab.1., a graf Graf. 2., názorně ukazují složitosti problémů pro x vstupních parametrů.

Funkce	$5x$	x^2	x^3	2^x	$x!$	x^x
x						
0	0	0	0	1	1	0
1	5	1	1	2	1	1
2	10	4	8	4	2	4
3	15	9	27	8	6	27
4	20	16	64	16	24	256
5	25	25	125	32	120	3125
6	30	36	216	64	720	46656
7	35	49	343	128	5040	823543
8	40	64	512	256	40320	16777216
9	45	81	729	512	362880	3,87E+08
10	50	100	1000	1024	3628800	1E+10
20	100	400	8000	1048576	2,4329E+18	1,05E+26
50	250	2500	125000	1,13E+15	3,04141E+64	8,88E+84
100	500	10000	1000000	1,27E+30	9,3326E+157	1E+200

Tab. 1 Složitost problémů pro x vstupních parametrů

Vzhledem k tomu, že tak obrovská čísla jako je $1,27E+30$ neboli $1,27 \cdot 10^{30}$, se špatně představují, tak pro srovnání od velkého třesku uplynulo zhruba $1E18$ sekund. To znamená, že kdyby byla na počítači při velkém třesku spuštěna takto náročná úloha a každou mikrosekundu by bylo provedeno jedno ohodnocení účelové funkce tak by úloha stále ještě nebyla dopočítána neboť od velkého třesku uplynulo „jenom“ 10^{24} μ s.



Graf. 2 Složitost problémů pro x vstupních parametrů

1.7 Terminologie

1.7.1 Účelová funkce

Tímto termínem označujeme funkci, jejíž optimalizací nalezneme optimální nastavení proměnných. Každou účelovou funkci si lze představit jako geometrický problém, jehož nejnižší bod v $N+1$ rozměrném prostoru chceme nalézt. Kde N udává počet proměnných, (vstupních parametrů) které chceme optimalizovat.

Správné zvolení účelové funkce je tedy velice důležité proto, abychom skutečně našli řešení na námi hledaný problém.

1.7.2 Hodnota účelové funkce

S jednotlivými jedinci pracujeme jako s body v prostoru. Každému chromozomu (jedinci v populaci) přiřadíme jeho hodnotu fitness. Tato hodnota představuje, jak moc daný chromozom řeší námi testovaný problém. Jak je tedy jasné, je velice důležité správně formulovat fitness funkci, aby řešení nalezené evolučním algoritmem bylo skutečně požadované řešení.

1.7.3 Prostor řešení

Ať už řešíme jakýkoliv problém, vždy se snažíme nalézt to nejlepší řešení. Množina všech možných řešení se nazývá prohledávaný prostor nebo jen prostor řešení. Každý bod v tomto prostoru reprezentuje jedno řešení s určitou hodnotou fitness. A někde v tomto prostoru je hledané optimum.

Prohledávaný prostor řešení nemusí být spojitý.

1.7.4 Populace

Pro evoluční algoritmy je typická práce s populací. Každý jedinec v populaci představuje jedno řešení. Populace je tedy skupina možných kombinací vstupních argumentů, jejichž optimální nastavení je hledáno. Ke každému jedinci také patří hodnota jeho vhodnosti neboli fitness.

Evoluční algoritmy stále dokola generují novou populaci na základě staré generace. To podle jakého klíče tak činí, se liší algoritmus od algoritmu.

Každý jedinec v populaci je vytvořen podle vzorového jedince, kterému se říká specimen. Tento vzor jedinců bývá definován typem proměnné, jako je například integer, real atd., a jeho spodní a horní hranicí (Lo a Hi).

Hranice jsou důležité pro omezení prohledávaného prostoru, například kvůli fyzikální nerealizovatelnosti jako je například záporná plocha.

Toto se opakuje podle počtu hledaných proměnných (dimenzí). Takový vzorový specimen pak může vypadat například takto:

$$\text{Specimen} = \{ \{ \text{Integer} \{0, 100\} \}, \{ \text{Real} \{-32.2, 32.2\} \} \}$$

1.7.5 No Free Lunch Teorém

Pro všechny optimalizační algoritmy, obecně platí takzvaný „No Free Lunch Teorém“ (NFL). Tento teorém tvrdí, že neexistuje takový algoritmus, který by dokázal vyřešit jakýkoliv problém lépe než kterýkoliv jiný algoritmus, neboli neexistuje algoritmus, který by byl nejlepší na všechny možné druhy problémů

Velmi často se tento teorém popisuje na obědě (lunch) v restauraci. Nelze jednoznačně vybrat jídlo, které bude v každé restauraci na světě z celého menu nejchutnější, nejlevnější a zároveň nejrychleji připraveno.

2 SOMA

Název algoritmu SOMA je vlastně zkratka z Samo-organizující se migrační algoritmus neboli: **S**elf-**O**rganizing **M**igration **A**lgorithm. Tento algoritmus byl v roce 1999 vytvořen prof. Ing. Ivan Zelinka, Ph.D.

Tento algoritmus řadíme mezi evoluční (algoritmy) a to i navzdory tomu, že v průběhu běhu algoritmu nedochází k vytváření nových potomků, jak bývá běžné u klasických evolučních algoritmů, ale již vytvoření jedinci migrují po hyperploše problému.

Činnost algoritmu SOMA je založena na vektorových operacích, stejně tak jako tomu je například u Diferenciální evoluce nebo u Particle swarm optimization (PSO). Přesnější je však zařadit tento algoritmus mezi memetické nebo hejnové algoritmy.

2.1 Parametry

Stejně jako u ostatních algoritmů, je i běh algoritmu SOMA silně závislý na nastavení vstupních parametrů. Celou řadou pokusů byly nalezeny intervaly, ve kterých algoritmus SOMA dosahuje průměrně nejlepších výsledků. Není zde jistota, že při nastavení parametrů v doporučeném intervalu, nalezne SOMA vždy globální optimum, ale průměrně se takto dosahuje lepších výsledků.

Vstupní parametry algoritmu SOMA jsou:

- PathLength
- Step
- D
- PRT
- PopSize
- Migrace
- MinDiv

2.1.1 PathLength

Tento parametr určuje vzdálenost, kterou jedinec urazí za jedno migrační kolo. Přesněji řečeno ukazuje poměr vzdálenosti jedince od toho nejlepšího (neboli leadera), a celkové

vzdálenosti, kterou jedinec urazí za jedno migrační kolo. To znamená, že při $PathLength = 0$ se jedinec vůbec nepohne a k žádné optimalizaci tedy nedojde, při $PathLength = 1$ se zastaví na pozici leadera a při $PathLength = 2$ se jedinec dostane na pozici za leadera do stejné vzdálenosti, na jaké začínal.

Z toho vyplývá, že pokud je $PathLength < 1$ migrující jedinec se vůbec nedostane na pozici leadera a celý optimalizační proces se tím silně degraduje. Pro řádný běh je tedy nutné aby byla hodnota $PathLength$ nastavena na hodnotu větší než 1. Doporučovaný interval bývá: $< 1,1; 5 >$. Testováním bylo zjištěno, že hodnota $PathLength = 3$ je dostačující na téměř všechny problémy.

Tento parametr se dříve označoval jako Mass [8]

2.1.2 Step

Tento parametr určuje vzdálenost mezi jednotlivými vzorky, a tím i počet vzorků, při jednom migračním kole. Pokud víme, že optimalizujeme jednoduchou nebo dokonce unimodální funkci, stačí tento parametr nastavit na vysokou hodnotu, čímž se velice urychlí hledání. Pokud však víme, že tvar prohledávané hyperplochy je velice komplikovaný, je potřeba nastavit tento parametr na malou hodnotu, čímž bude během jednoho migračního kola uděláno více vzorků a prohledávaný prostor tedy bude zmapován podrobněji. Za žádných okolností se však nedoporučuje nastavovat hodnotu Step tak, aby došlo k tomu, že by jeden s testovaných vzorků byl zároveň aktuální leader. V takovém případě by se totiž funkce velice rychle zastavila v lokálním extrému.

Nastavení parametru Step na hodnotu 0,11 je tedy mnohem lepší než nastavení na 0,1. Doporučený rozsah hodnot parametru Step tedy je $< 0,11; PathLength >$

2.1.3 D

Parametr D neboli dimenzions, udává počet dimenzí hledaného problému, neboli počet proměnných, které bude algoritmus optimalizovat.

2.1.4 PRT

Parametr PRT, neboli perturbace ovlivňuje perturbační vektor. Určuje míru stochastické složky, možná by bylo přesnější říci nemíru stochastické složky, v algoritmu SOMA. Perturbační vektor říká zdali se bude aktivní jedinec pohybovat (směrem) k leaderovy přímo, nebo bude některou dimenzi ignorovat. Tento parametr, který nám tedy udává míru

„mutace“ se nastavuje v rozmezí $<0; 1>$. Testováním bylo zjištěno, že ideální hodnota tohoto parametru je zhruba 0,1. Pokud $PRT = 1$, stochastická složka zcela zaniká a algoritmus téměř vždy zůstane v prvním lokálním optimu.

2.1.5 PopSize

Tento řídicí parametr udává počet jedinců, se kterými bude algoritmus SOMA pracovat. Minimální hodnota tohoto parametru je 2. Celkové by se nemělo klesnout pod hodnotu 10, neboť při menším počtu jedinců je algoritmus velice degradován. Maximální hodnota tohoto parametru není nijak omezena, ale nastavovat zde astronomické hodnoty nemá smysl. Doporučená hodnota se pohybuje kolem $0,3D - 5D$.

Tento parametr se dříve označoval jako NP[8].

2.1.6 Migrace

Udává, kolikrát se celá populace přeorganizuje, než dojde k ukončení vyhledávání. Nastavení hodnoty tohoto parametru je velice obtížné, neboť se velice těžko odhaduje, jak dlouho bude algoritmu trvat, než nalezne optimum.

2.1.7 MinDiv

Parametr MinDiv, neboli minimální diverzita, je stejně jako Migrace ukončovací parametr. Tento parametr je definován jako maximální povolený rozdíl hodnot mezi leaderem a nejhorším jedincem populace. Pro nastavení tohoto parametru je nezbytné znát alespoň přibližné hodnoty, kterých může funkce nabývat. Pokud je například známo, že se hodnoty funkce budou pohybovat v rozsahu 10 – 11 tak nastavení parametru $MinDiv = 1$ okamžitě zastaví optimalizaci. V takovémto případě se doporučuje nastavení parametru $MinDiv = 0,01$.

Tento parametr se dříve označoval jako AcceptedError[8].

2.2 Princip SOMA

Princip algoritmu SOMA je graficky znázorněn na (Obr.2).

Každý jedinec populace je ohodnocen účelovou funkcí a je zvolen leader, neboli jedinec s nejlepší hodnotou účelové funkce, pro následující migrační kolo. To znamená, že se všichni jedinci začnou pohybovat směrem k Leaderovi pomocí skoků, jejichž velikost je dána parametrem Step, až do vzdálenosti PathLength. Po každém skoku si každý jedinec přepočítá

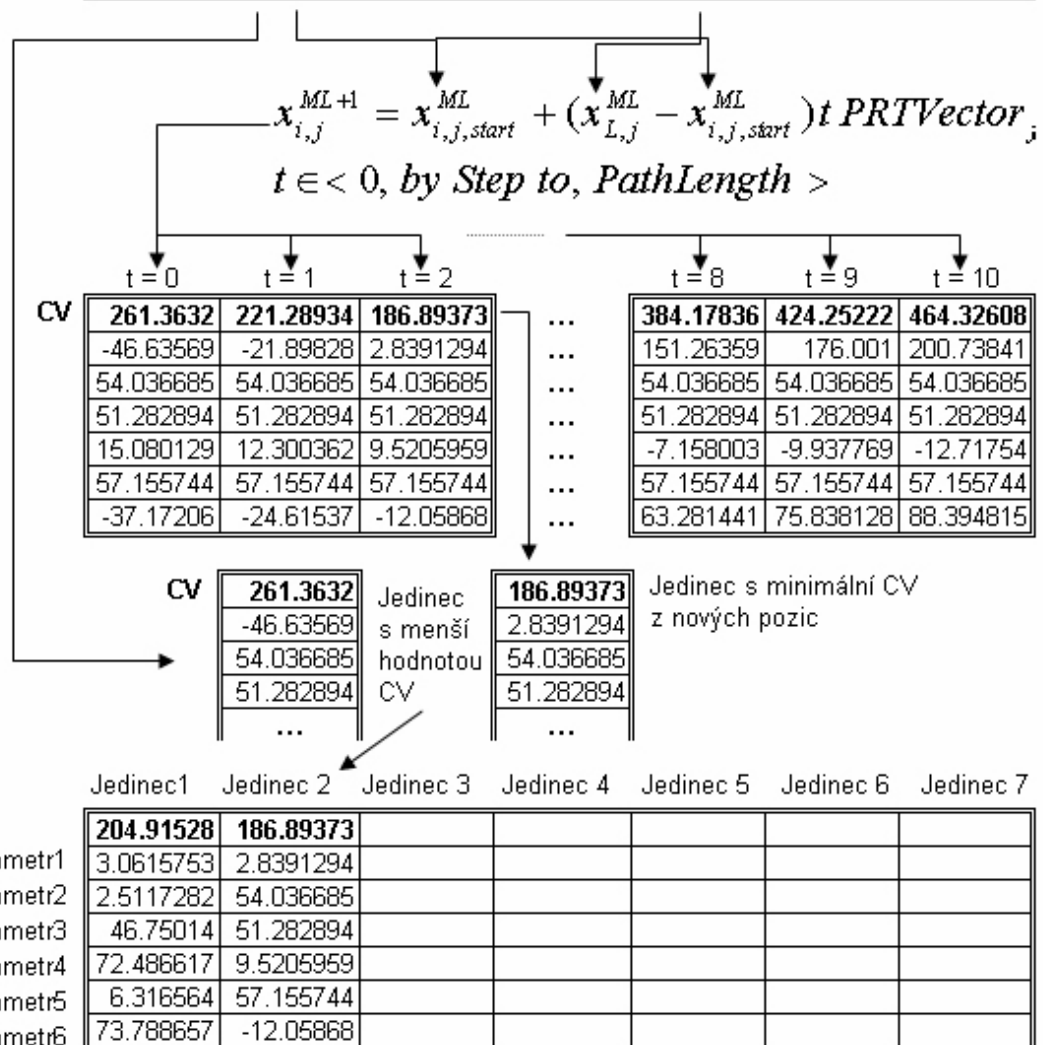
svou hodnotu účelové funkce na nové pozici, na které se ocitl, a zároveň si pamatuje tu nejlepší pozici, na které během tohoto kola byl. Pohyb jedince pokračuje, dokud není dosaženo pozice, ve vzdálenosti dané parametrem PathLength.

Po ukončení běhu se jedinec vrací na nejlepší pozici, ve které se během putování ocitl. Neboli po skončení aktuálního migračního kola jsou všichni jedinci, mimo Leadera, přemístěni na nejlepší pozici, kterou aktuální migrační kolo potkali. A celý proces se znovu opakuje, dokud není splněna podmínka Migrace nebo MinDev.

Parametry SOMA		PRT vektor, pro běh každého jedince je jiný	
Step	0.3	If Rand < PRT then 1 else 0	↔ 1
PathLength	3	If Rand < PRT then 1 else 0	↔ 0
PRT	0.1	If Rand < PRT then 1 else 0	↔ 0
AcceptedError	0.1	If Rand < PRT then 1 else 0	↔ 1
Migrations	1000	If Rand < PRT then 1 else 0	↔ 0
NP	7	If Rand < PRT then 1 else 0	↔ 1

Účelová funkce $f(x) = \text{Abs}(\text{Parametr1}) + \text{Abs}(\text{Parametr2}) + \dots + \text{Abs}(\text{Parametr6})$

	Aktivní jedinec			Leader			
	Jedinec 1	Jedinec 2	Jedinec 3	Jedinec 4	Jedinec 5	Jedinec 6	Jedinec 7
CV	204.91528	261.3632	163.79679	121.73019	107.52784	121.06024	120.20974
Parametr1	3.0615753	-46.63569	5.0246553	38.723912	35.822343	0.0715185	23.761224
Parametr2	2.5117282	54.036685	85.104704	0.2928606	24.111443	4.2879691	20.384665
Parametr3	46.75014	51.282894	11.347164	3.0796963	24.657689	60.241731	33.437248
Parametr4	72.486617	15.080129	2.916686	3.6713463	5.8142407	4.5385164	4.0482021
Parametr5	6.316564	57.155744	58.829537	26.610056	12.43856	23.891907	4.2271271
Parametr6	73.788657	-37.17206	0.5740442	49.352316	4.6835676	28.028598	34.351273



Obr. 1. Princip SOMA – převzato [8]

3 DIFERENCIÁLNÍ EVOLUCE

Diferenciální evoluce (DE) je poměrně mladým evolučním algoritmem (od roku 1995, Price, Storm). Principiálně je dost podobný na genetické algoritmy, potomci se vyrábí z rodičů, pracuje s generacemi, atd. Algoritmus se vyvinul z takzvaného genetického žhání.

3.1 Parametry

Stejně jako u ostatních algoritmů, je i běh Diferenciální evoluce silně závislý na nastavení vstupních parametrů.

Vstupní parametry Diferenciální evoluce jsou:

- CR
- D
- NP
- F
- Generations

3.1.1 CR

Parametr CR udává míru mutace při vytváření nového jedince. Může nabývat hodnot $\langle 0;1 \rangle$. Pokud bude parametr CR nastaven na hodnotu blízkou nebo dokonce rovnající se 0, evoluce se rychle zastaví, protože potomci budou čistou kopií rodiče. Pokud se však parametr CR nastaví na hodnotu blízkou nebo dokonce rovnou 1, bude každý potomek tvořen jenom šumovým vektorem a algoritmus se tím degraduje na náhodné hledání. Tento parametr se doporučuje nastavit v rozsahu 0,8 - 0,9.

3.1.2 D

Parametr D neboli dimenzions, udává počet dimenzí hledaného problému, neboli počet proměnných, které bude algoritmus optimalizovat.

3.1.3 NP

Tento řídicí parametr udává počet jedinců v populaci, se kterými bude algoritmus Diferenciální evoluce pracovat. Pro běh algoritmu je nezbytně nutné, aby tento parametr

byl nastaven minimálně na hodnotu 4. Toto je totiž minimální počet jedinců, při kterých algoritmus ještě funguje. Doporučené je však alespoň 10D.

3.1.4 F

Parametr F neboli mutační konstanta, udává míru mutace při vytváření šumového vektoru. Jeho hodnota se může pohybovat v rozsahu 0 – 2. Není však vhodné, aby tento parametr nabýval hodnoty 0, neboť je potom šumový vektor vyroben pouze z jednoho jedince a nikoliv z jedinců tří. Doporučuje se nastavit tento parametr v rozsahu 0,3 – 0,9.

3.1.5 Generations

Tento parametr dává, kolikrát se celá populace obrodí, než dojde k ukončení vyhledávání. Nastavení hodnoty tohoto parametru je velice obtížné, neboť se velice těžko odhaduje, jak dlouho bude algoritmu trvat, než nalezne optimum.

3.2 Princip Diferenciální evoluce

Cílem DE je v cyklech zvaných „generace“ vyšlechtit co nejlepší populaci jedinců ve smyslu hodnot účelové funkce, jenž je spojen s každým jedincem. Během každé generace se provádí tyto kroky:

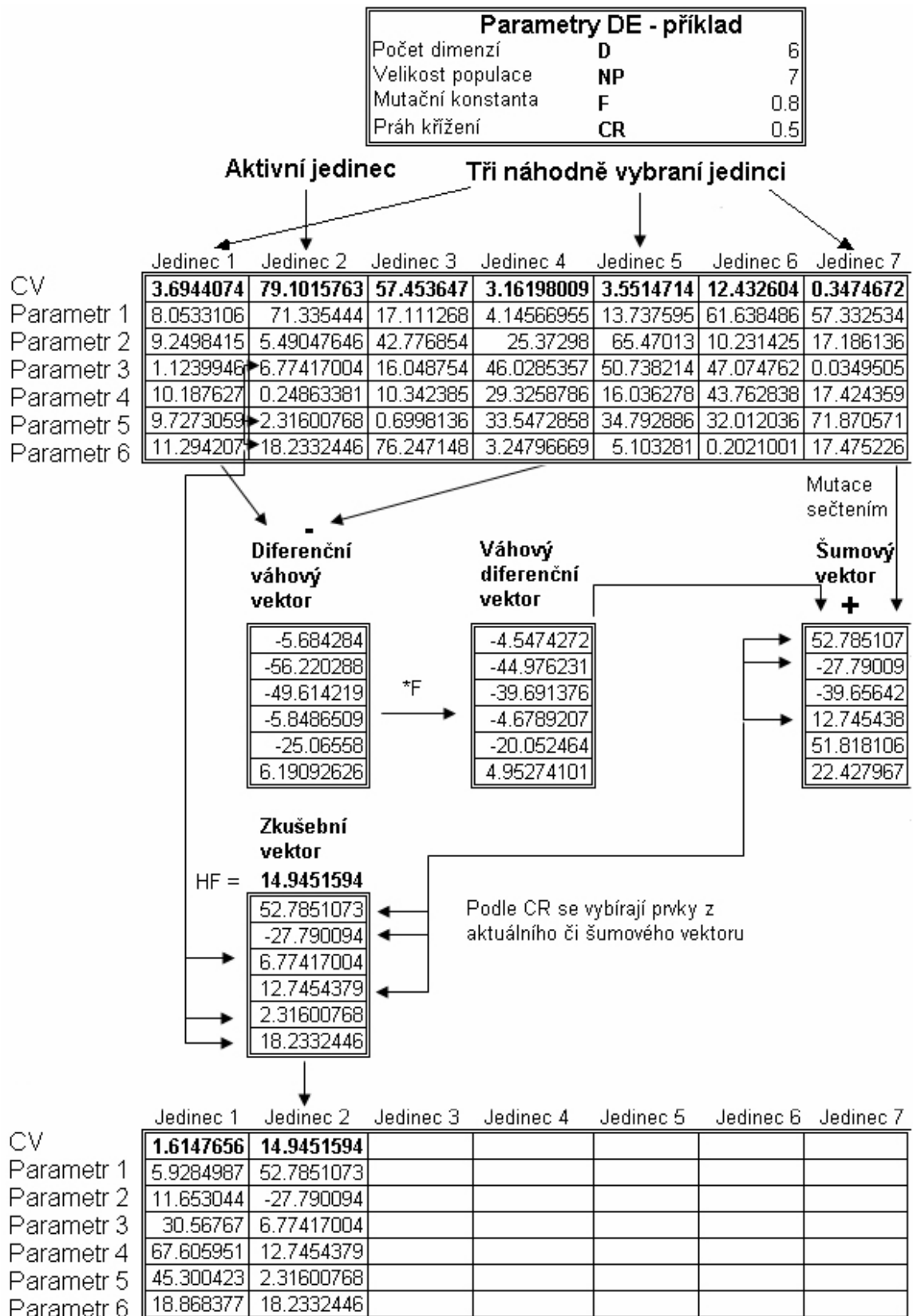
Nejprve se stanoví parametry (F, CR, NP, Generations, Dim). Poté je nutné nadefinovat prototyp jedince (Specimen) – tj. z jakých typů čísel se budou skládat jedinci, např. reálných, celočíselných atd. Po sestavení Specimen se vytvoří populace vygenerováním množiny jedinců podle prototypového vektoru. U každého jedince se musí počítat s jedním prvkem navíc a tím je hodnota účelové funkce. Během generace se provádí ještě cyklus, který zabezpečuje postupné evoluční šlechtění každého jedince z populace viz. (Obr.3).

V tomto cyklu se postupně vybírá jeden jedinec (aktivní jedince, cílový vektor) za druhým až do konce populace (tím je ukončena jedna generace) a pro každého z nich je proveden evoluční cyklus, v němž je prováděna mutace a křížení, (tj). Náhodně se zvolí tři další různé vektory (jedinci) z populace. První dva se od sebe odečtou, získá se tzv. diferenční vektor. Ten se vynásobí mutační konstantou F, která jej tím pádem změní, a získá se váhový diferenční vektor. Ten se přičte k třetímu náhodně vybranému vektoru a získá se tzv. šumový vektor. Poté si připraví tzv. „zkušební vektor“ a z cílového a šumového vektoru se bere postupně jeden prvek za druhým (první z obou, druhý z obou...) a pro takto vybranou každou dvojici se generuje náhodné číslo v rozsahu 0 – 1 a porovnává se s

konstantou CR. Pokud je toto číslo menší než CR, pak se do příslušné pozice ve zkušebním vektoru umístí prvek z vektoru šumového a v opačném případě z vektoru cílového.

Tak se získá zkušební vektor, jehož hodnota účelové funkce se porovná s hodnotou účelové funkce cílového vektoru. Na pozici cílového vektoru v nové populaci je vybrán ten vektor – jedinec, který má hodnotu účelové funkce lepší [8].

Diferenciální evoluce je ukončena pouze tehdy, provede-li se uživatelem zadaný počet generací. Jiný ukončovací parametr tento algoritmus nemá.



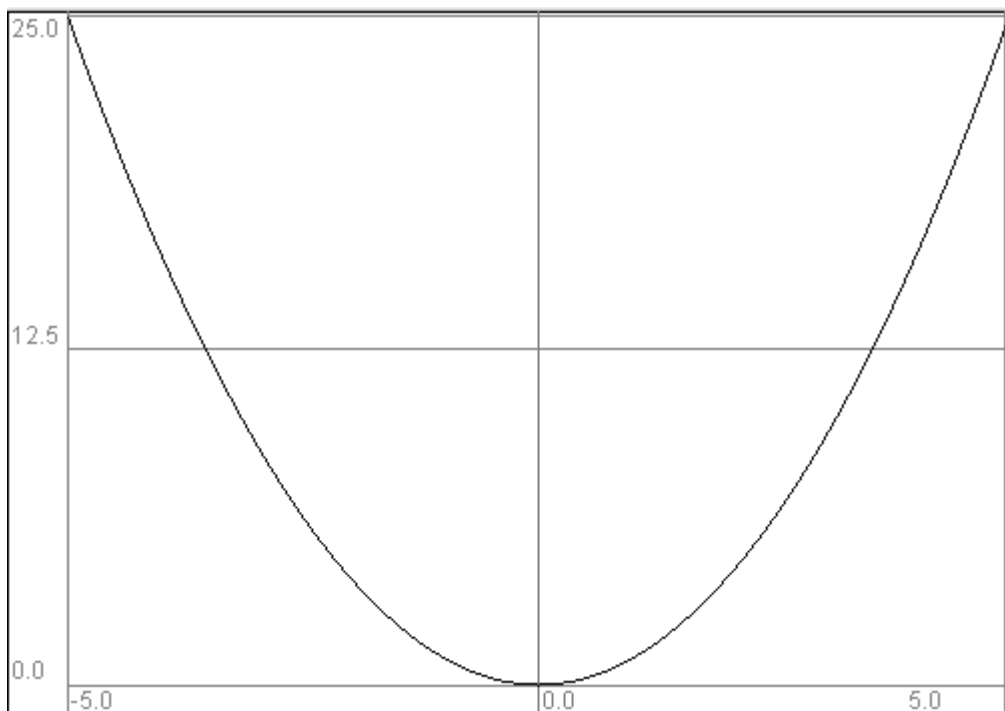
Obr. 2. Princip DE – převzato z [8]

4 POUŽITÉ TESTOVACÍ PROBLÉMY

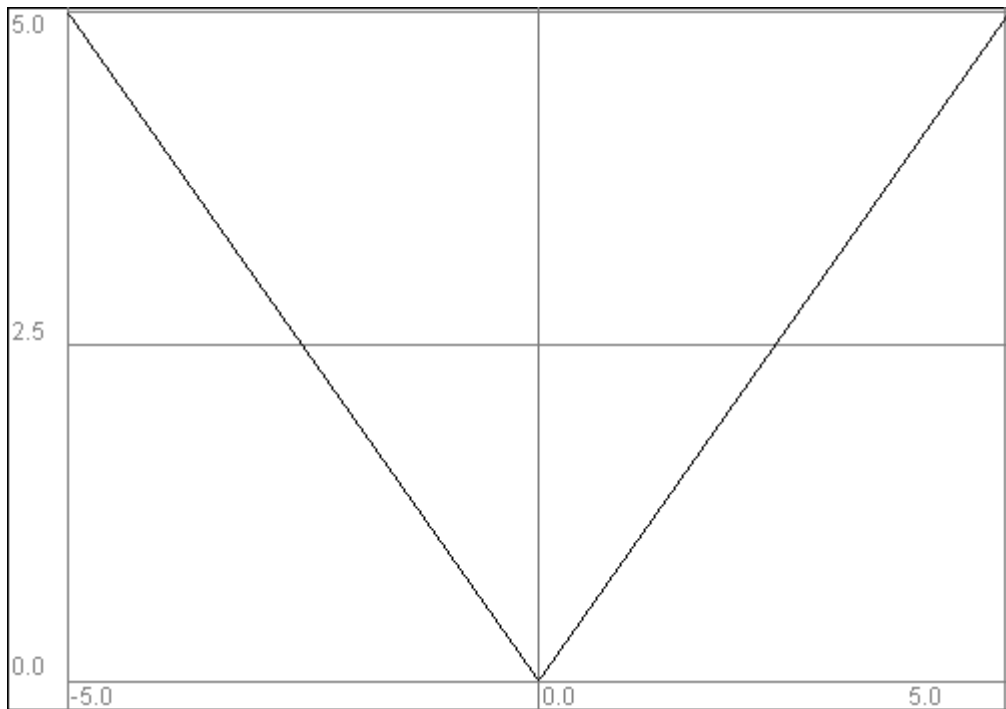
Aby bylo možné otestovat evoluční algoritmy, zda fungují správně, bylo třeba vytvořit množinu testovacích problémů. Jako tyto testovací problémy byly zvoleny dva unimodální a dva multimodální problémy.

Zvolené testovací problémy jsou:

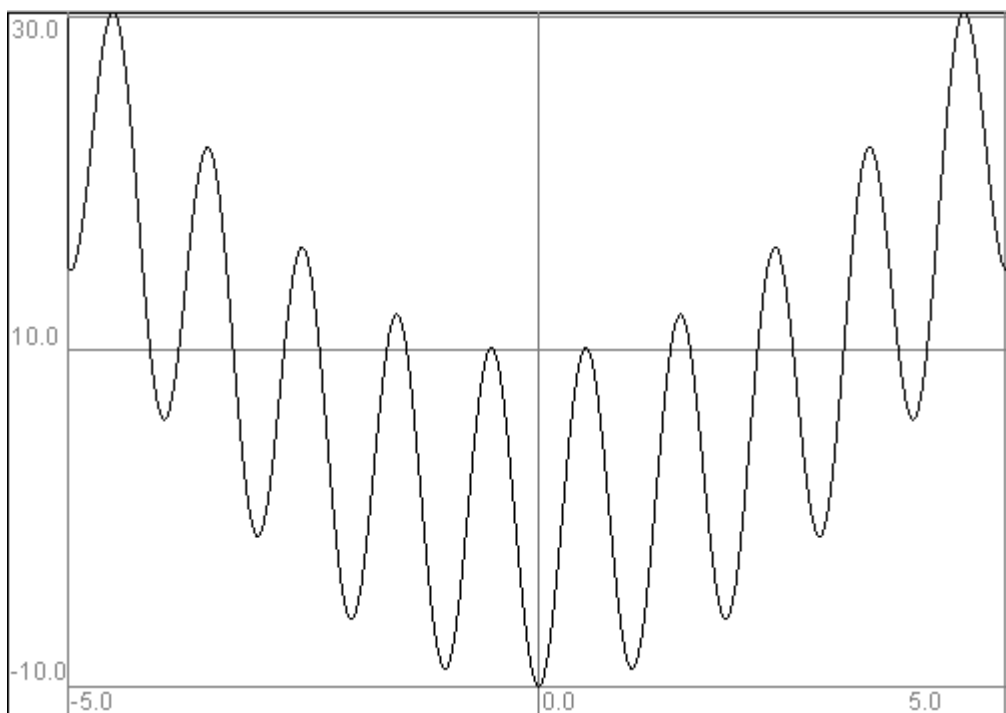
- 1st De Jong
- 3rd De Jong
- Rastrigin
- Schwefel



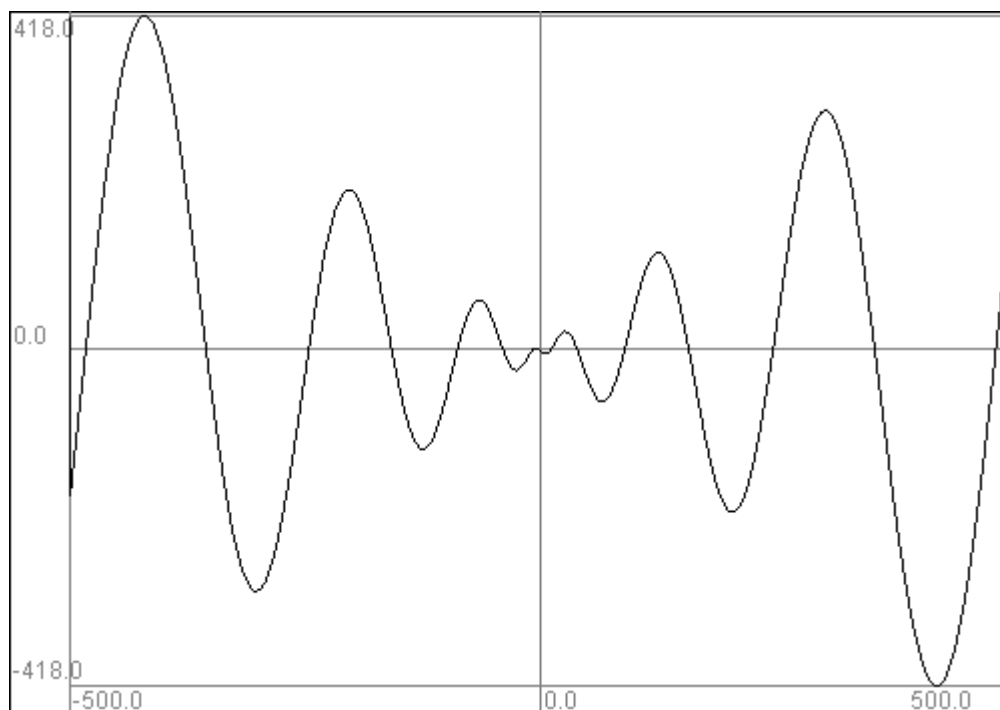
Obr. 3. 1st De Jong



Obr. 4. 3rd De Jong



Obr. 5. Rastrigin



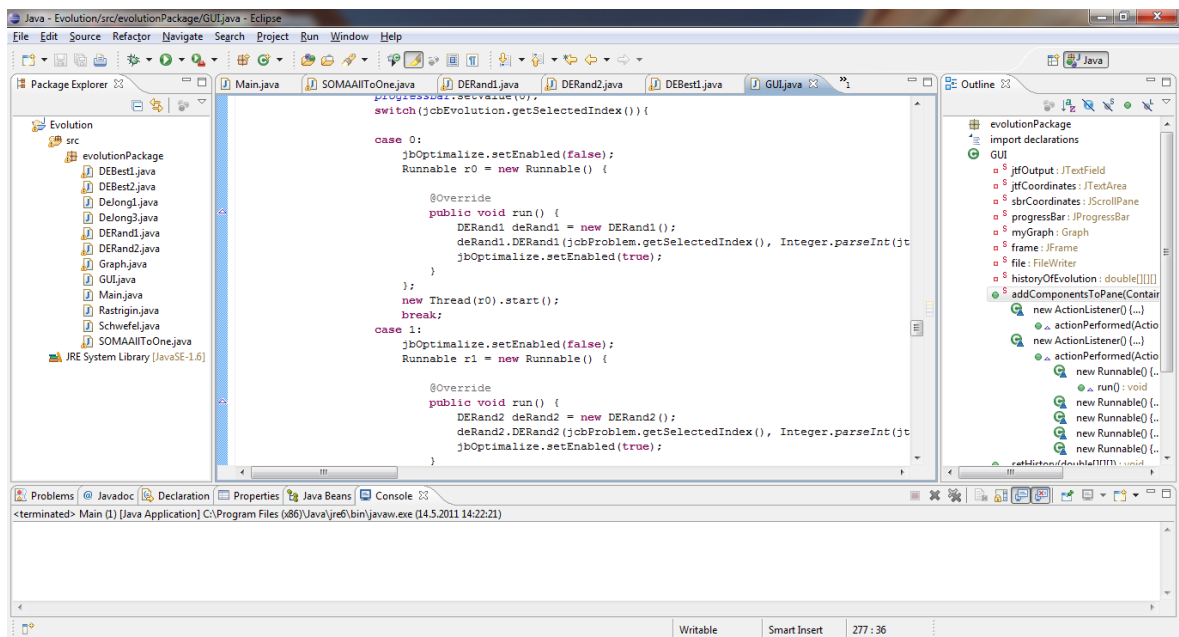
Obr. 6. Schwefel

PRAKTICKÁ ČÁST

5 POPIS PROGRAMU

Jako praktickou část své diplomové práce jsem napsal menší knihovnu evolučních algoritmů a grafický interpret, který umí s touto knihovnou pracovat a aplikovat ji na testovací problémy.

Celý program je napsán v programovacím jazyce Java za použití vývojového nástroje *Eclipse IDE for Java Developers* ve verzi *Helios Service Release 2*. (viz Obr. 7.) celý projekt je uložen na CD přiloženém k diplomové práci.



Obr. 7. Vývojové prostředí *Eclipse IDE for Java Developers* ve verzi *Helios Service Release 2*

5.1 Popis grafického rozhraní

Celý program pracuje v jednom okně (viz Obr. 8. a Obr. 9.). Toto okno běží v samostatném vlákně, takže se nezasekne při provádění výpočtu. Vzhledově se okno skládá z několika základních částí.

V horní části je umístěn panel, ve kterém je nabídka *File*, která skrývá možnosti *Save* a *Exit*.

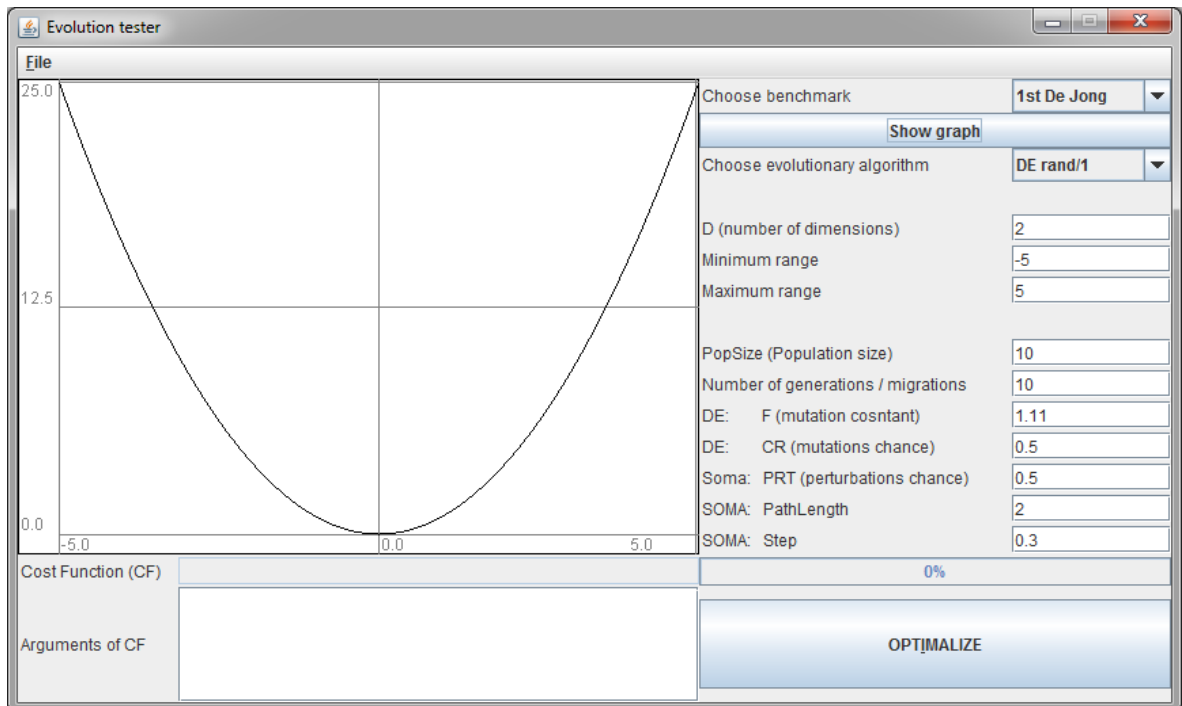
Vlevo je umístěn graf, ve kterém je vykreslen buď testovací problém (viz Obr. 8.) nebo graf konvergence (viz Obr. 9.). V grafu konvergence zelená linka znázorňuje nejlepšího jedince z populace a červená zase jedince nejhoršího. Na grafu je tedy zřetelně vidět rychlost konvergence evolučního algoritmu.

Pod grafem jsou dvě výstupní textová pole, do kterých se zaznamenávají výsledky evoluce. Do textového pole *Cost fiction (CF)* se zapisuje hodnota účelové funkce nejlepšího jedince a do pole *Afguments of CF* se vypíše hodnoty jednotlivých argumentů nejlepšího jedince, neboli nastavení optimálního řešení nalezené evolucí.

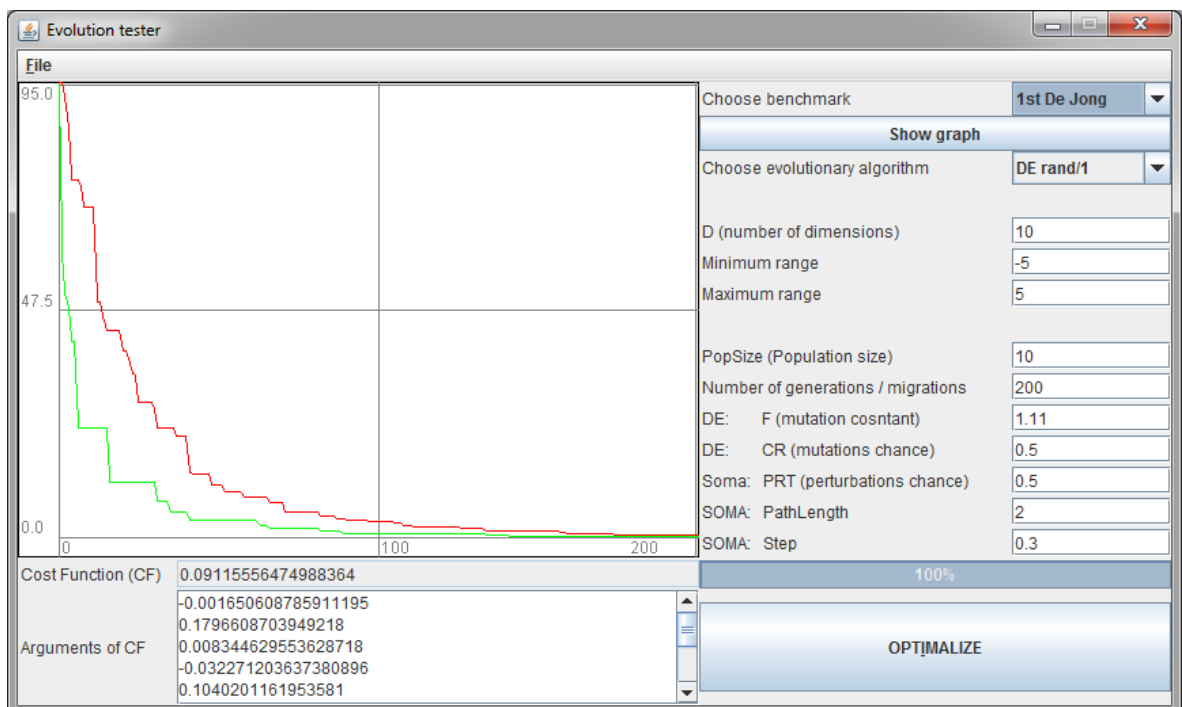
V pravé části okna se nachází pole rozevírací nabídky pro výběr testovacího problému *Choose benchmark*, tlačítko pro zobrazení grafu aktuálně vybraného testovacího problému *Show graph*, pole rozevírací nabídky pro výběr evolučního algoritmu *Choose evolution algorithm*, dále pak tři textová pole pro definování prohledávaného prostoru, počet dimenzí *D (number of dimensions)* a definice hranice prohledávaného prostoru *Minimum range* a *Maximum range*.

Dále následují textová pole pro zadávání parametrů pro jednotlivé testovací algoritmy. Těmito parametry jsou velikost populace *PopSize (Population size)*, počet generací u Diferenciální Evoluce případně migrací u algoritmu SOMA, které evoluční vykoná, než se ukončí *Number of generations / migrations*. Následují dva parametry pouze pro Diferenciální Evoluci, mutační konstanta *F (mutation constant)* a *CR (station chance)*. A tři parametry pouze pro algoritmus SOMA, parametr pro definici míry perturbace *PRT (perturbation chance)*, parametr udávající délku cesty *PathLength*, a nakonec parametr definující délku kroku *Step*.

V pravém spodním rohu se nachází indikátor průběhu právě probíhající evoluce a tlačítko *Optimize* pro spuštění optimalizace.



Obr. 8. Grafické rozhraní s grafem testovací funkce



Obr. 9. Grafické rozhraní s grafem konvergence

Volba *Save* uloží informace o všech jedincích z poslední evoluce, aby bylo možné tyto informace dále vyhodnocovat a zpracovávat, do textového souboru, který si uživatel

vybere pomocí klasického dialogu pro uložení. Jeden takový výpis je na Obr. 10. Výpis se drží jednoduchých pravidel:

- Každý řádek obsahuje právě jednoho jedince z generace
- Každý jedinec je vždy na tom samém řádku každou generaci
- Řádek population # odděluje jednotlivé generace a zároveň říká, kolikátá generace právě skončila.
- V jednotlivých sloupcích jsou vypsané atributy jedince
- V posledním sloupci je hodnota účelové funkce jedince

```

out - Poznámkový blok
Soubor Úpravy Formát Zobrazení Nápověda
249.8541879162118; -190.23864455250595; 98617.85710044438;
-75.65792276679872; -187.60497843945836; 40919.74921265652;
283.38456896841103; 185.37897249374225; 114672.17737224775;
-252.64153184556292; -206.52499983832996; 106480.31917149478;
-12.515850321246717; -489.21811255491366; 239491.008161056;
-337.90792466881646; -295.1709021334999; 201307.6270202907;
-405.71291136618083; -110.4400392727236; 176799.96872378324;
-365.38476017210496; 254.2592069057721; 198153.76726237888;
8.647177611844882; 338.2879639761894; 114513.52025180642;
-164.64887318341482; 203.89087462025032; 68680.74019397885;
population 0
249.8541879162118; -113.36340865820641; 75278.37764187719;
-75.65792276679872; -187.60497843945836; 40919.74921265652;
283.38456896841103; 185.37897249374225; 114672.17737224775;
-252.64153184556292; -206.52499983832996; 106480.31917149478;
-12.515850321246717; -489.21811255491366; 239491.008161056;
250.52657775676954; 302.28782616788897; 154141.49601182656;
-405.71291136618083; -1.160383155211406; 164604.3129382894;
324.2499077652176; 254.2592069057721; 169785.74698210432;
8.647177611844882; 55.913341256352055; 3201.0754111000724;
-164.64887318341482; 203.89087462025032; 68680.74019397885;
population 1
249.8541879162118; -113.36340865820641; 75278.37764187719;
-75.65792276679872; -187.60497843945836; 40919.74921265652;
283.38456896841103; 185.37897249374225; 114672.17737224775;
-22.469863539587152; -206.52499983832996; 43157.47032570986;
-12.515850321246717; -386.9012960035059; 149849.2593584563;
250.52657775676954; 302.28782616788897; 154141.49601182656;
-405.71291136618083; -1.160383155211406; 164604.3129382894;
324.2499077652176; 254.2592069057721; 169785.74698210432;
8.647177611844882; 55.913341256352055; 3201.0754111000724;
-164.64887318341482; 203.89087462025032; 68680.74019397885;
population 2
-41.118007088709156; -113.36340865820641; 14541.952929554644;
-75.65792276679872; -187.60497843945836; 40919.74921265652;
283.38456896841103; 185.37897249374225; 114672.17737224775;

```

Obr. 10. Výstup aplikace uložený do souboru

Volba *Exit* řádně ukončí program, stejně tak jako to udělá například křížek v pravém horním rohu obrazovky.

5.2 Popis tříd

Aby se dalo ve zdrojovém kódu lépe vyznat, je podle tříd rozdělen do několika samostatných souborů. Tyto třídy jsou:

- Main
- DeJong1
- DeJong3
- Rastrigin
- Schwefel
- GUI
- DEBest1
- DEBest2
- DERand1
- DERand2
- SOMAAIIToOne

5.2.1 Třída Main

Třída Main neobsahuje příliš mnoho řádků zdrojového kódu. Vytvoří pouze vlákno, ve kterém spustí grafické rozhraní celé aplikace.

5.2.2 Třídy testovacích funkcí

Ani třídy testovacích funkcí (DeJong1, DeJong3, Rastrigin, Schwefel) nejsou příliš rozsáhlé, obsahují pouze jednu metodu, která vrací hodnotu účelové funkce na základě přijatého argumentu.

5.2.3 Třída GUI

Definuje grafické rozhraní a jednotlivé události, které může uživatel vyvolat. Celé grafické rozhraní je spuštěno jako samostatný proces a evoluční algoritmy jsou vždy volány v jiném vlákně. Tím se předchází problému, že by se celé okno evolučního algoritmu při běhu

zaseklo. Lze v něm tedy dále provádět úpravy kromě zavolání další evoluce. Aby nedošlo, byť nedopatřením, k několika násobnému spuštění evoluce, označí se tlačítko jako nestisknutelné až do doby, kdy skončí běh evolučního algoritmu.

5.2.4 Třídy optimalizačních algoritmů

Program obsahuje třídy těchto evolučních algoritmů: Diferenciální Evoluce, a to sice celkem ve čtyřech mutacích: *DE/Best/1*, *DE/Best/2*, *DE/Best/1,DE/Best/2* a evoluční algoritmus *SOMA* v podobě *AllToOne*. Každá třída je napsaná, aby pracovala samostatně, takže je možné kteroukoliv třídu kdykoliv oddělat, upravit nebo přidat třídy další. Vybraný evoluční algoritmus je vždy spuštěn jako samostatné vlákno, takže nedochází k „zamrznutí“ grafického rozhraní.

5.3 Přidání evolučního algoritmu

Pro přidání nového evolučního algoritmu je potřeba napsat tento algoritmus jako vlastní třídu. Tuto třídu je potom potřeba implementovat do balíčku *evolutionPackage* ke zbytku programu.

```
package evolutionPackage;
```

Dále je pro přidání třídy s novým evolučním algoritmem nutné přidat jeho název do rozevíracího pole, aby si jej mohl uživatel vybrat. To se učiní jednoduchým připsáním do pole *Stringů* jménem *typeOfEvolution* ve třídě *GUI* na řádku 30.

```
String[] typeOfEvolution = {"DE rand/1", "DE rand/2", "DE  
best/1", "DE best/2", "SOMA/AllToOne"};
```

A přidat volání této třídy do větvení *switch* (s patřičným pořadovým číslem) ve třídě *GUI* na řádku 298.

Pro správné vykreslení grafu konvergence je nezbytné, aby nová třída s evolučním algoritmem obsahovala proměnou *best*, ve které bude uložena historie nejlepších jedinců a *worst*, ve které bude uložena historie nejhorších jedinců.

```
ArrayList<double[]> best;
```

```
ArrayList<double[]> worst;
```

A pro správné uložení do souboru je nutné po dokončení evoluce zavolat funkci *setHistory* a předat jí třírozměrné pole s celkovou historií. Toto pole musí být formátu

```
double [ číslo generace ] [ počet dimenzí ] [ velikost populace ]
```

Aby bylo implementování nové třídy kompletní, je ještě zapotřebí zprovoznit indikátor průběhu. To se udělá jednoduše tak, že třída s evolučním algoritmem bude předávat třídě *GUI* hodnotu, kterou má zobrazit v indikátoru průběhu. Učiní tak zavoláním metody *setProgressBar* s celočíselným parametrem. Například takto

```
setProgressBar ( (int) (100*rated/endRated) );
```

5.4 Vlákna

Aby nedocházelo k tomu, že se celá aplikace bude jevit jako zaseknutá, během doby, kdy bude systém zaneprázdněn výpočty pro evoluční algoritmus, je aplikace rozdělena do dvou vláken. Jedno vlákno se stará o běh grafického rozhraní:

```
public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            GUI.createAndShowGUI();
        }
    });
}
```

A druhé vlákno provádí samotné výpočty pro evoluční algoritmus:

```
Runnable r0 = new Runnable() {

    @Override
    public void run() {
        DERand1 deRand1 = new DERand1();
        deRand1.DERand1(jcbProblem.getSelectedIndex(),
            Integer.parseInt(jtfDimensions.getText()),
            Double.parseDouble(jtfMin.getText()),
            Double.parseDouble(jtfMax.getText()),
            Integer.parseInt(jtfPopSize.getText()),
            Double.parseDouble(jtfMutationCR.getText()),
            Double.parseDouble(jtfMutationChance.getText()),
            Integer.parseInt(jtfGeneration.getText()));
        jbOptimize.setEnabled(true);
    }
};
new Thread(r0).start();
```

Nedojde tedy k zaseknutí grafického rozhraní. Bylo také třeba omezit tlačítko *Optimize*, aby ho bylo možné stisknout pouze v případě, že není spuštěno druhé vlákno s běžícím evolučním algoritmem.

```
jbOptimize.setEnabled(false);
```

6 POROVNÁNÍ IMPLEMENTOVANÝCH EVOLUČNÍCH ALGORITMŮ

Schopnost jakéhokoli evolučního algoritmu silně závisí na nastavení vstupních parametrů a tvaru použité testovací funkce. Proto je možné, že algoritmus, který je obecně považován za lepší vyjde v následujících testováních jako algoritmus horší.

Neexistuje nic jako ideální nastavení parametrů evolučního algoritmu pro všechny problémy. Stejně tak jako neexistuje ideální evoluční algoritmus (viz. No Free Lunch Teorém).

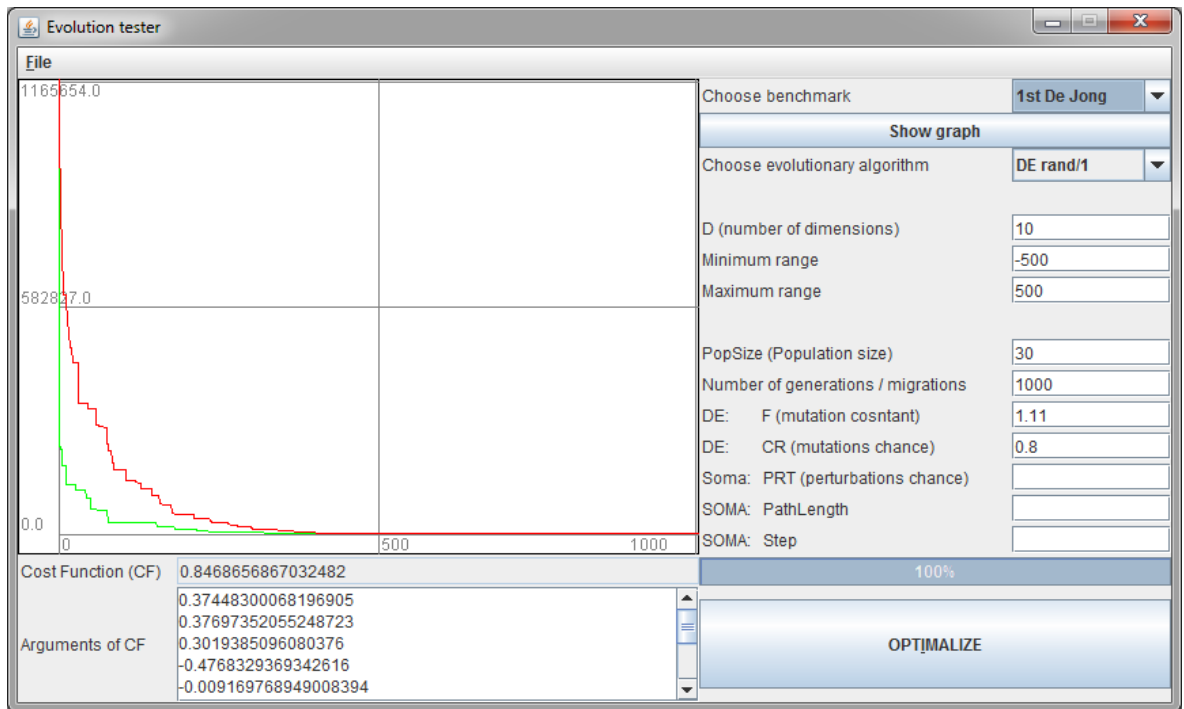
Následující testy tato tvrzení potvrzují. Jako testovací funkce byly zvoleny 1st De Jong (viz Obr. 3.) a Schwefel (viz Obr. 6.) a testovací funkce DE/Rand/1 a SOMA/AllToOne.

6.1 Ukázka závislosti evolučního algoritmu DE/Rand/1 na vstupních parametrech, testovací funkce 1st De Jong

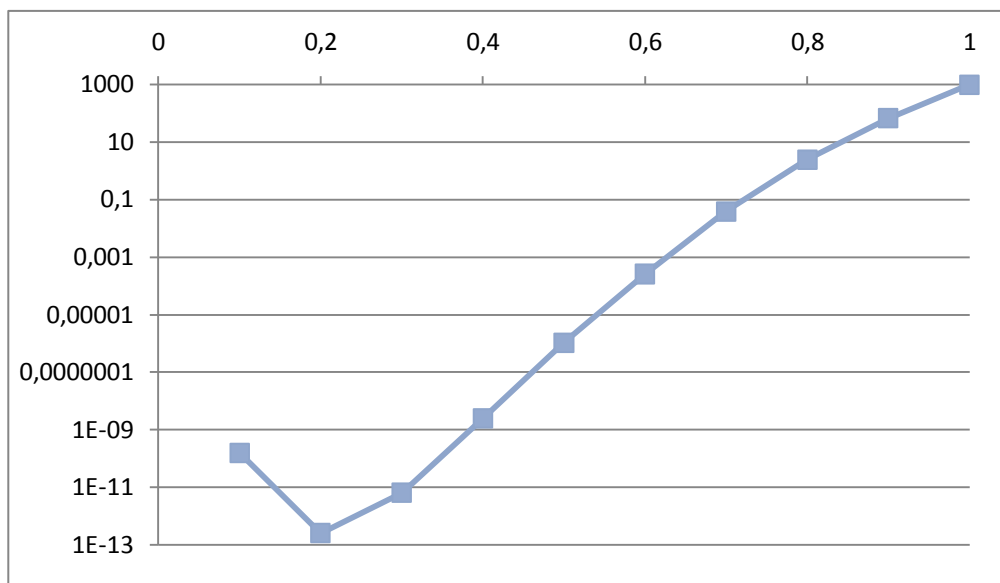
Jako první bude ručně nalezeno nejlepší nastavení evolučního algoritmu DE/Rand/1 pro testovací problém jménem 1st De Jong (viz Obr. 3.).

6.1.1 Hledání parametru CR

Graf. 3. zobrazuje závislost Diferenciální Evoluce s označením DE/Rand/1 na parametru CR. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 11.). Pro testování byla použita testovací funkce 1st De Jong (viz Obr. 3.) tato funkce má optimum přesně v nule, takže se snadno hodnotí přesnost, neboť samotný výsledek evolučního algoritmu udává nepřesnost od ideálního výsledku (viz Graf. 3.), u kterého musela být použita logaritmická stupnice a z tabulky Tab. 2. viditelné ideální nastavení tohoto parametru je někde kolem 0,2.



Obr. 11. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru CR



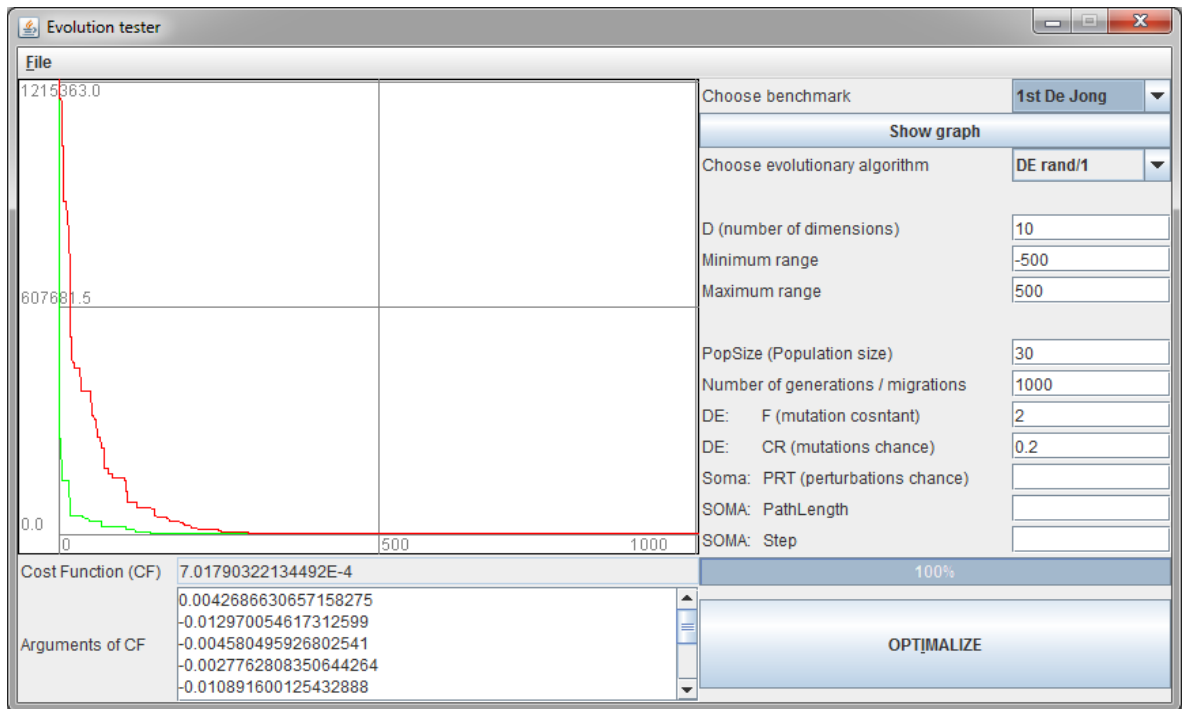
Graf. 3. Graf závislosti Diferenciální evoluce na vstupním parametru CR

Hodnota CR	0,1	0,2	0,3	0,4	0,5
Č. měření					
1	2,40E-10	2,02E-13	2,82E-12	3,56E-09	4,46E-07
2	1,20E-10	2,32E-13	1,37E-11	7,49E-10	1,52E-06
3	1,77E-10	3,31E-13	3,15E-12	7,14E-10	1,80E-07
4	1,22E-10	1,33E-13	1,10E-11	5,12E-09	5,46E-07
5	1,26E-10	3,98E-13	1,71E-12	2,23E-09	2,67E-06
PRŮMĚR	1,57E-10	2,59E-13	6,48E-12	2,47E-09	1,07E-06
Hodnota CR	0,6	0,7	0,8	0,9	1
Č. měření					
1	1,03E-04	9,05E-02	4,62E+00	4,60E+01	6,99E+02
2	2,88E-04	2,95E-02	2,31E+00	5,33E+01	1,04E+03
3	6,13E-04	1,89E-02	2,44E+00	5,04E+01	7,04E+02
4	1,59E-04	3,94E-02	1,62E+00	1,56E+02	1,30E+03
5	1,39E-04	1,35E-02	1,28E+00	3,00E+01	1,21E+03
PRŮMĚR	2,61E-04	3,83E-02	2,45E+00	6,71E+01	9,90E+02

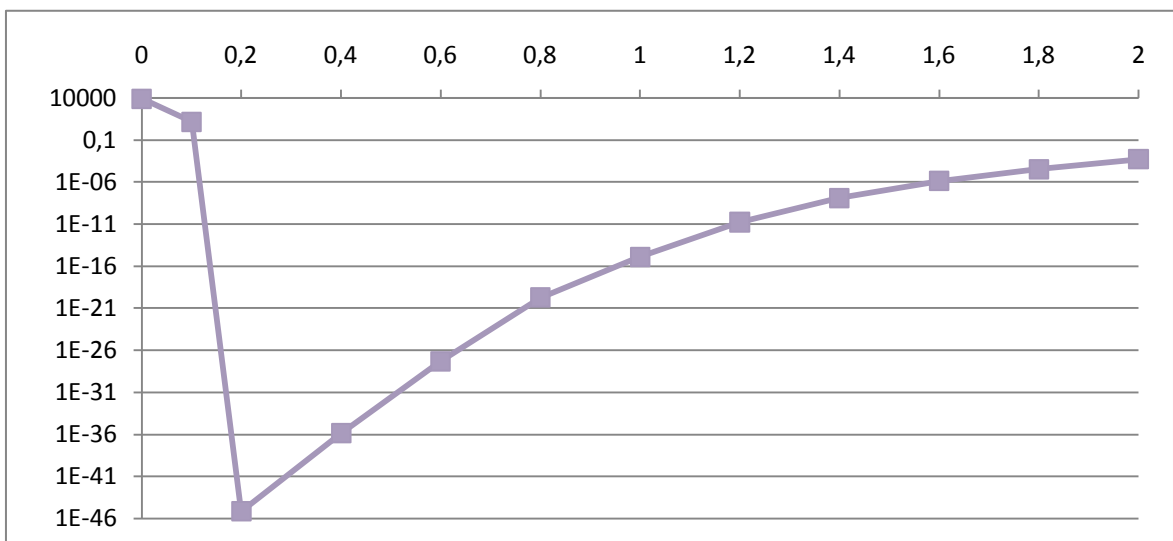
Tab. 2. Tabulka závislosti Diferenciální evoluce na vstupním parametru CR

6.1.2 Hledání parametru F

Graf. 4. zobrazuje závislost Diferenciální Evoluce s označením DE/Rand/1 na parametru F. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 12.). Pro testování byla použita testovací funkce 1st De Jong (viz Obr. 3.) tato funkce má optimum přesně v nule, takže se snadno hodnotí přesnost, neboť samotný výsledek evolučního algoritmu udává nepřesnost (viz Graf. 4.), u kterého musela být použita logaritmická stupnice a z tabulky Tab. 3. viditelné ideální nastavení tohoto parametru je někde okolo 0,2.



Obr. 12. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru F



Graf. 4. Graf závislosti Diferenciální evoluce na vstupním parametru F

Hodnota F	0	0,1	0,2	0,4	0,6	0,8
Č. měření						
1	4295,722576	3,91E-04	1,76E-45	2,26E-38	7,24E-28	4,37E-20
2	8065,951644	69,91601076	4,18E-48	1,07E-37	1,11E-28	1,52E-20
3	8604,698967	0,00283153	6,99E-47	4,18E-36	8,27E-29	1,66E-20
4	6601,666371	2,74E-05	3,66E-46	1,40E-36	6,71E-28	9,06E-21
5	10393,36117	0,002096607	1,44E-45	1,44E-36	8,42E-28	6,14E-21
PRŮMĚR	7,59E+03	1,40E+01	7,28E-46	1,43E-36	4,86E-28	1,81E-20
Hodnota F	1	1,2	1,4	1,6	1,8	2
Č. měření						
1	2,25E-16	1,95E-11	2,20E-08	2,45E-06	3,23E-05	8,08E-04
2	3,08E-16	1,11E-11	9,29E-09	9,01E-07	4,50E-05	3,62E-04
3	2,39E-15	9,49E-12	7,38E-09	4,65E-07	1,58E-05	4,16E-04
4	1,88E-16	2,65E-11	7,82E-09	8,27E-07	2,32E-05	5,94E-04
5	3,34E-15	1,71E-11	1,58E-08	1,79E-06	5,07E-05	3,26E-04
PRŮMĚR	1,29E-15	1,67E-11	1,25E-08	1,29E-06	3,34E-05	5,01E-04

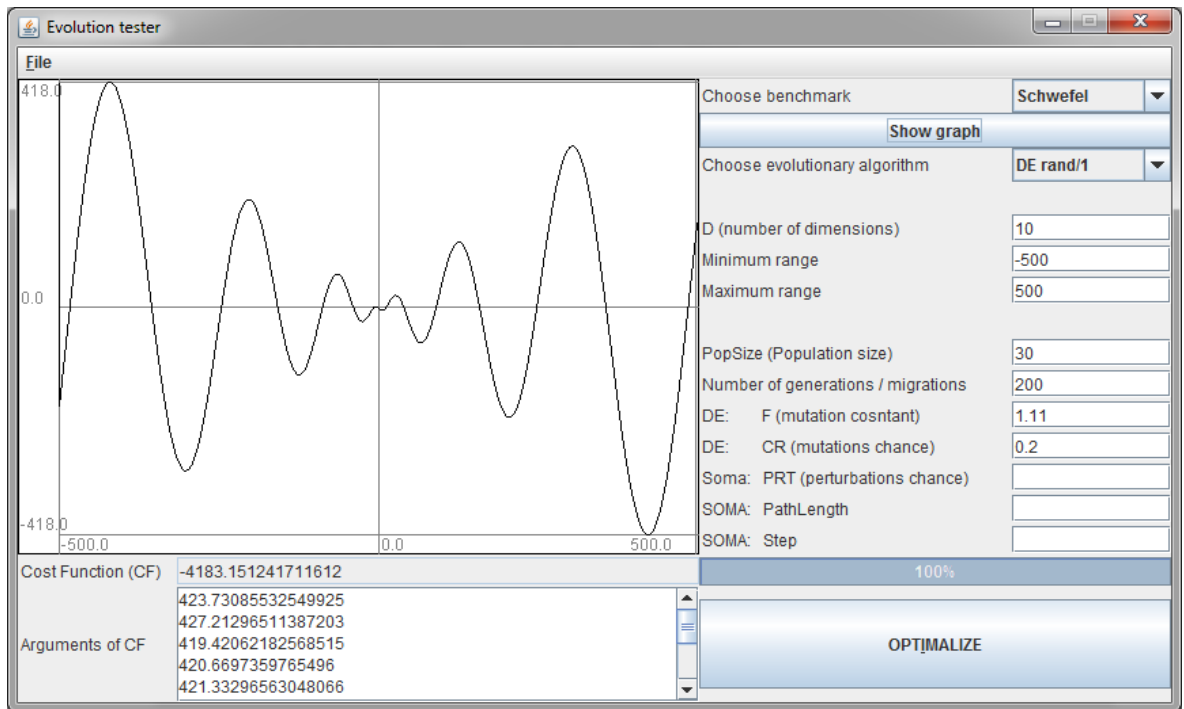
Tab. 3. Tabulka závislosti Diferenciální evoluce na vstupním parametru F

6.2 Ukázka závislosti evolučního algoritmu DE/Rand/1 na vstupních parametrech, testovací funkce Schwefel

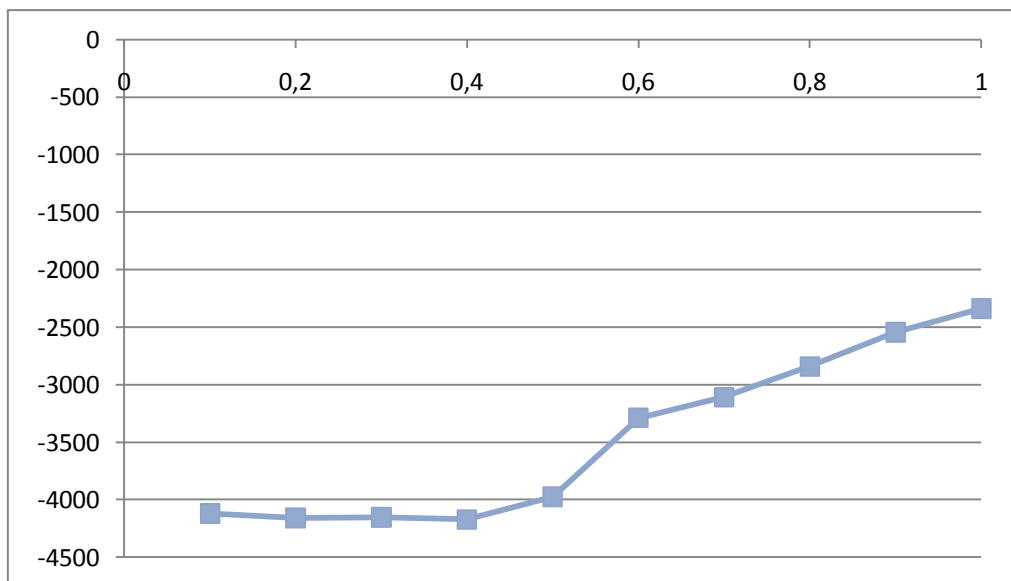
Jako další bude ručně nalezeno nejlepší nastavení evolučního algoritmu DE/Rand/1. Pro testování byla použita testovací funkce Schwefel (viz Obr. 6.) tato funkce má podle literatury [7] optimum v bodě 420,97 (mé testování tento bod upřesnilo až na 420,968746) což odpovídá ohodnocení účelové funkce $-418,9828872724338 \cdot \text{počet dimenzí}$.

6.2.1 Hledání parametru CR

Graf. 5. zobrazuje závislost Diferenciální Evoluce s označením DE/Rand/1 na parametru CR. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 15.). Výsledky testování jsou zapsány v tabulce Tab. 4. a pro přehlednost zobrazeny v grafu Graf. 5. Z výsledků testování je zřejmé, že ideální nastavení tohoto parametru je 0,1 až 0,5 nejlépe však vypadá hodnota 0,4.



Obr. 13. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru CR



Graf. 5. Graf závislosti Diferenciální evoluce na vstupním parametru CR

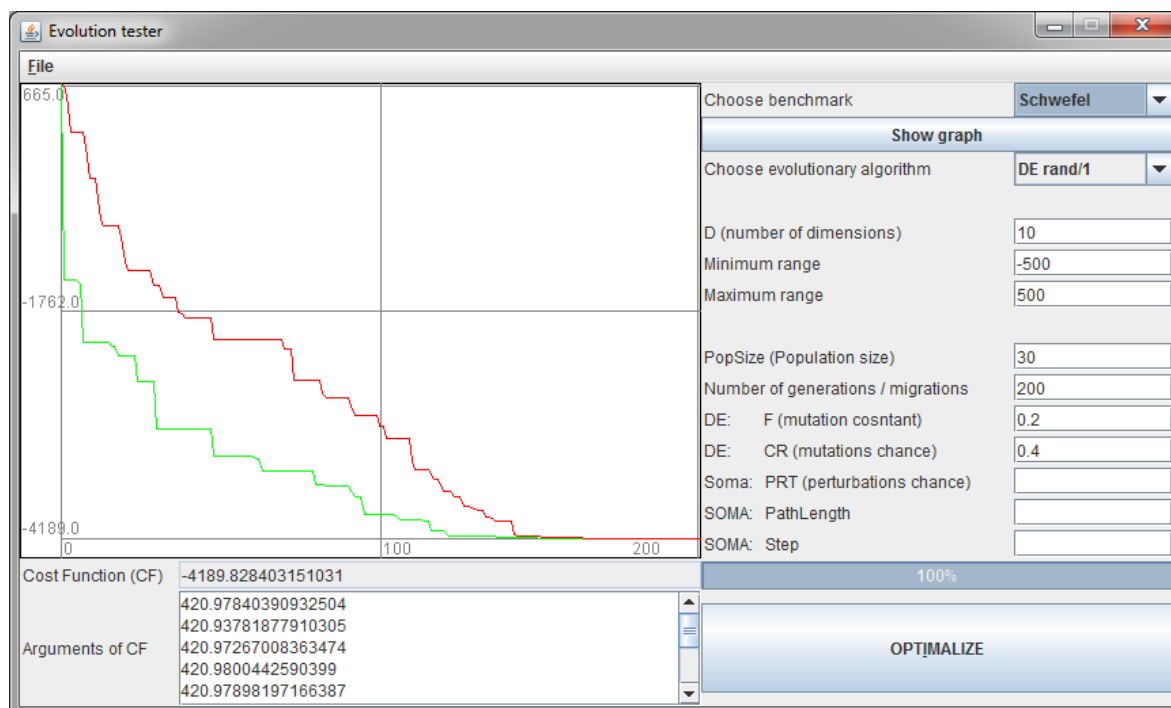
Hodnota CR	0,1	0,2	0,3	0,4	0,5
Č. měření					
1	-4104,0213	-4162,4876	-4162,3814	-4174,6246	-4013,2954
2	-4139,7902	-4141,7271	-4166,4966	-4173,0360	-4007,7043
3	-4103,2428	-4165,0415	-4149,3778	-4164,8476	-3917,9325
4	-4132,4032	-4160,5545	-4147,6579	-4177,7139	-4034,5540
5	-4124,5410	-4168,9526	-4145,2035	-4169,3613	-3904,3086
PRŮMĚR	-4120,7997	-4159,7527	-4154,2234	-4171,9167	-3975,5589
Hodnota CR	0,6	0,7	0,8	0,9	1
Č. měření					
1	-3437,8790	-2904,9424	-2881,4025	-2367,1454	-2328,2067
2	-3112,8309	-3263,6318	-2822,8525	-2742,3926	-2241,8477
3	-3102,9670	-2982,7727	-2871,5464	-2705,4049	-2300,3614
4	-3412,9455	-3409,6125	-2958,2710	-2474,2993	-2672,5265
5	-3385,8543	-2984,3388	-2679,5981	-2438,8019	-2152,7364
PRŮMĚR	-3290,4953	-3109,0596	-2842,7341	-2545,6088	-2339,1357

Tab. 4. Tabulka závislosti Diferenciální evoluce na vstupním parametru CR

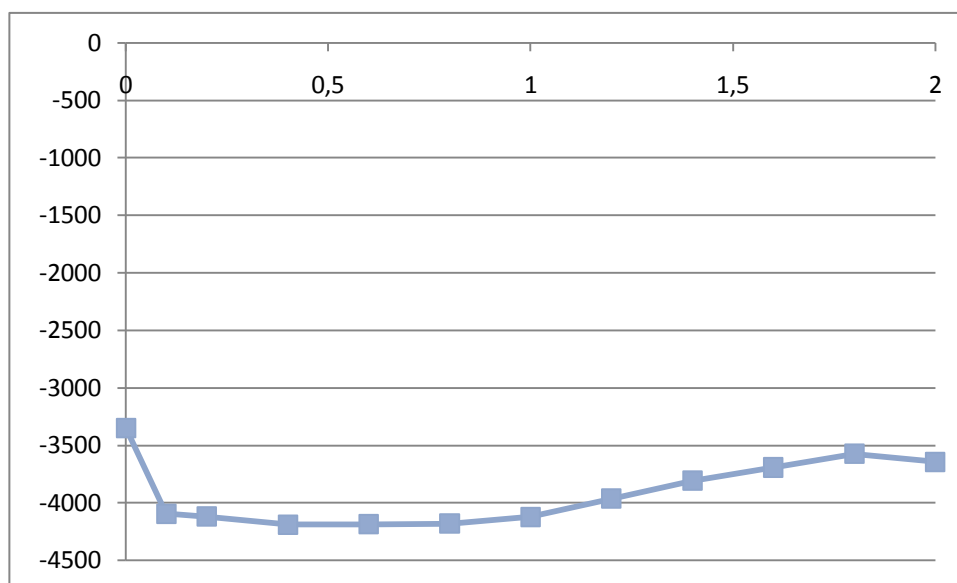
6.2.2 Hledání parametru F

Graf. 6. zobrazuje závislost Diferenciální Evoluce s označením DE/Rand/1 na parametru CR. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 15.). Pro testování byla použita testovací funkce Schwefel (viz Obr. 6.) tato funkce má podle literatury [7] optimum v bodě 420,97 (mé testování tento bod upřesnilo na 420, 968746) což odpovídá ohodnocení účelové funkce $-418, 9828872724338 \cdot$ počet dimenzí. Výsledky testování jsou zapsány v tabulce Tab. 5. a pro přehlednost zobrazeny v grafu Graf. 6.

Z výsledků testování je zřejmé, že ideální nastavení tohoto parametru je 0,1 až 1 nejlépe však vypadá hodnota 0,4.



Obr. 14. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru F



Graf. 6. Graf závislosti Diferenciální evoluce na vstupním parametru F

Hodnota F	0	0,1	0,2	0,4	0,6	0,8
Č. měření						
1	-2978,94726	-4189,77322	-4189,82887	-4188,98974	-4186,80330	-4187,23665
2	-3866,95606	-4066,07639	-4189,80972	-4189,82224	-4177,01488	-4177,29878
3	-3359,72029	-3952,84249	-4189,82885	-4189,82033	-4188,90016	-4175,21358
4	-3002,20326	-4071,15119	-4071,39053	-4189,72590	-4188,60972	-4176,90572
5	-3532,54129	-4189,27274	-3952,95219	-4189,80199	-4188,26317	-4183,66156
PRŮMĚR	-3348,07363	-4093,82321	-4118,76203	-4189,63204	-4185,91825	-4180,06326
Hodnota F	1	1,2	1,4	1,6	1,8	2
Č. měření						
1	-4180,69513	-3998,52433	-3609,69659	-3684,01636	-3433,80713	-3743,92434
2	-4151,44546	-3787,99035	-3796,79075	-3609,79861	-3557,02138	-3689,16522
3	-4100,18385	-4070,02978	-3845,27578	-3660,26147	-3506,62086	-3738,47263
4	-4098,74415	-3883,05066	-3851,44962	-3759,26340	-3737,14100	-3477,86179
5	-4081,77221	-4079,54467	-3929,80985	-3745,02470	-3637,04527	-3567,33181
PRŮMĚR	-4122,56816	-3963,82796	-3806,60452	-3691,67291	-3574,32713	-3643,35116

Tab. 5. Tabulka závislosti Diferenciální evoluce na vstupním parametru F

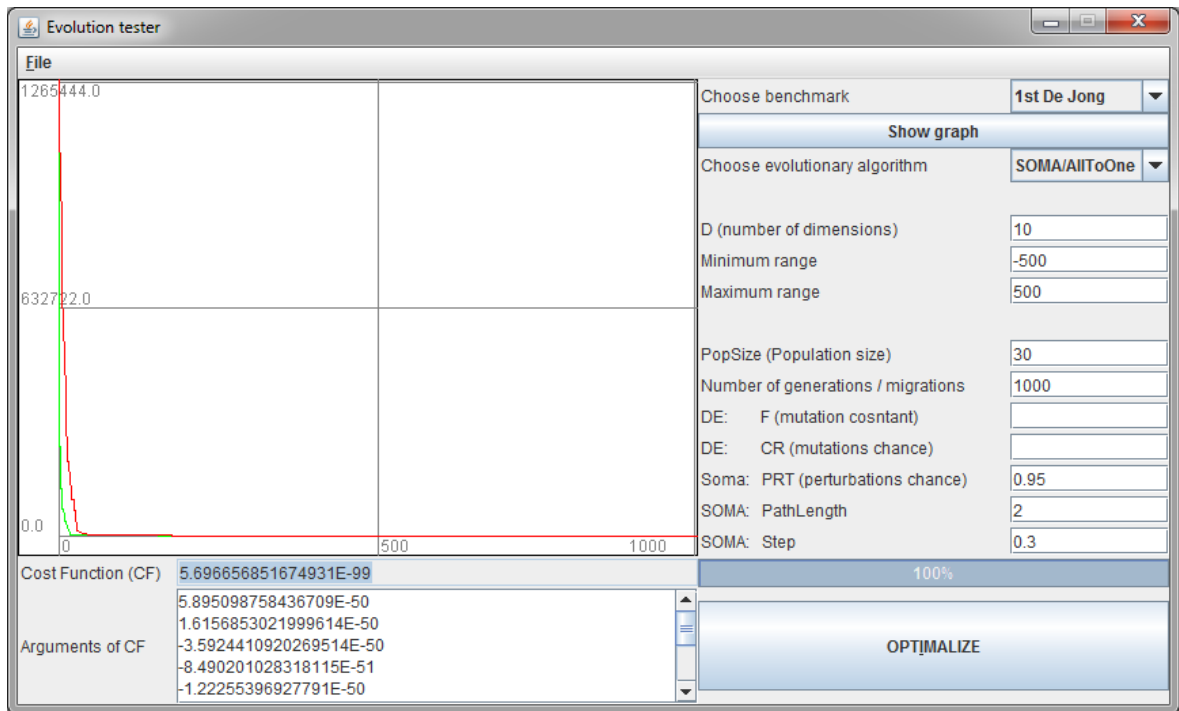
6.3 Ukázka závislosti evolučního algoritmu SOMA AllToOne na vstupních parametrech, testovací funkce 1st De Jong

Nyní bude zopakován ten samý postup jako u algoritmu DE/Rand/1 u algoritmu SOMA AllToOne k ručnímu nalezení vhodného nastavení pro testovací problém jménem 1st De Jong (viz Obr. 3.).

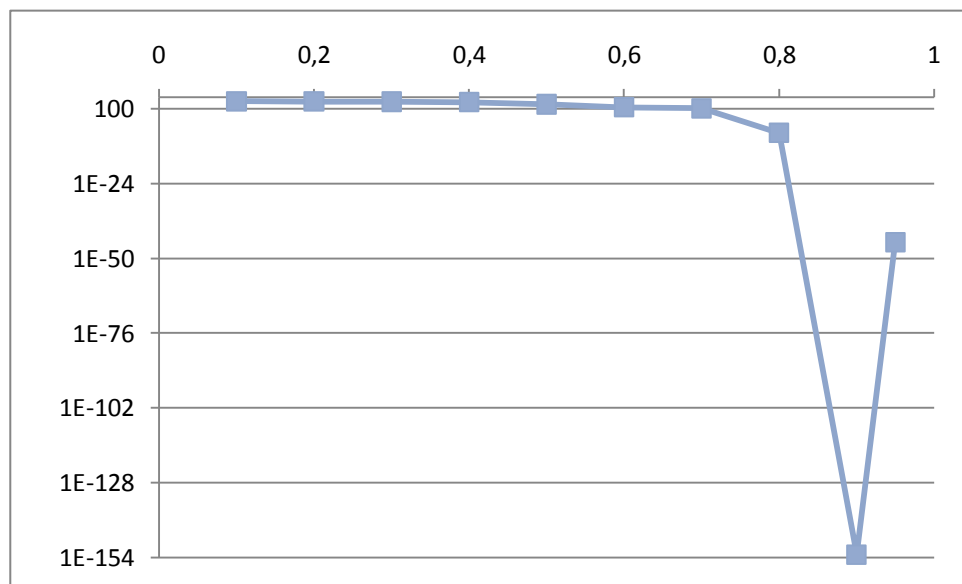
6.3.1 Hledání parametru PRT

Graf. 7. zobrazuje závislost SOMA AllToOne na parametru PRT. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 11.). Pro testování byla použita testovací funkce 1st De Jong (viz Obr. 3.) tato funkce má optimum přesně v nule, takže se snadno hodnotí přesnost, neboť samotný výsledek evolučního algoritmu udává nepřesnost od výsledku ideálního.

Z výsledků testování je zřejmé, že ideální nastavení tohoto parametru je kolem hodnoty 0,9. (viz Graf. 7. u kterého musela být použita logaritmická stupnice a z tabulky Tab. 6.).



Obr. 15. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PRT



Graf. 7. Graf závislosti Diferenciální evoluce na vstupním parametru PRT

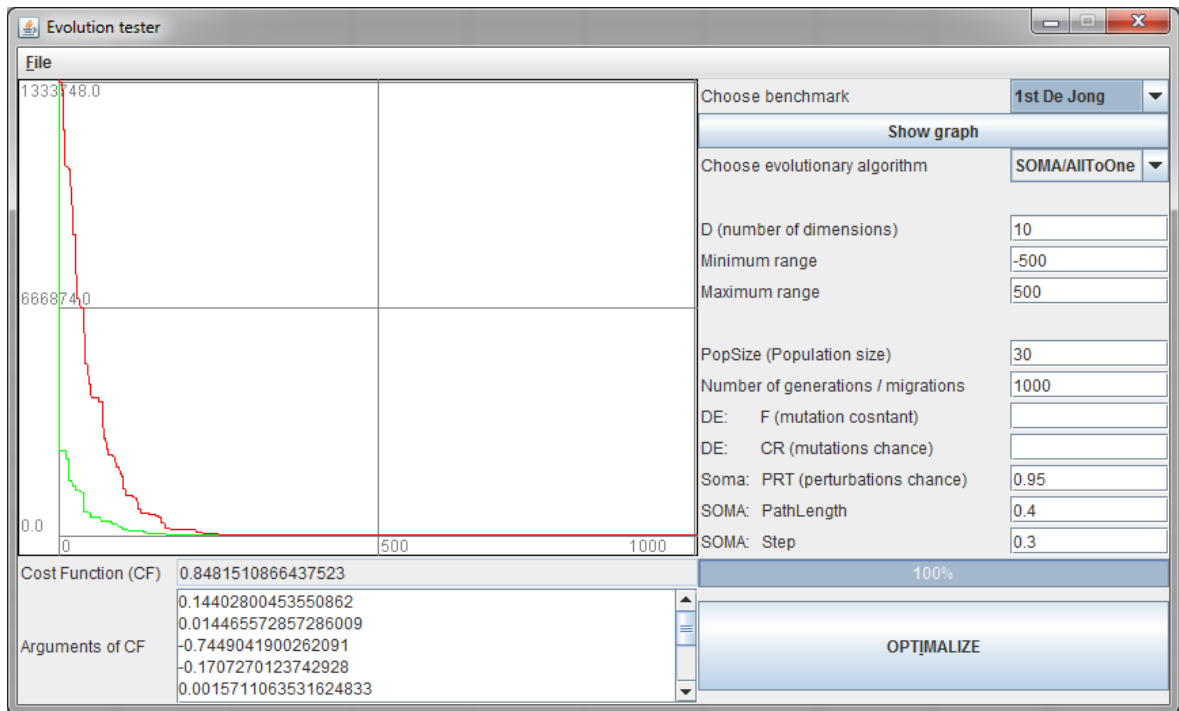
Hodnota PRT	0,1	0,2	0,3	0,4	0,5
Č. měření					
1	4,83E+04	2,31E+04	1,75E+04	1,54E+04	2,93E+03
2	1,75E+04	3,81E+04	3,01E+04	1,52E+04	2,18E+03
3	3,43E+04	2,89E+04	5,50E+04	8,14E+03	2,45E+03
4	6,19E+04	1,03E+04	1,47E+04	2,00E+04	5,97E+03
5	1,07E+04	4,61E+04	1,19E+04	2,66E+04	2,82E+03
PRŮMĚR	3,45E+04	2,93E+04	2,59E+04	1,71E+04	3,27E+03
Hodnota PRT	0,6	0,7	0,8	0,9	0,95
Č. měření					
1	4,31E+01	1,85E+01	9,94E-07	1,31E-154	4,37E-98
2	3,25E+02	6,51E+01	1,77E-07	5,58E-153	5,86E-100
3	5,76E+02	1,10E+01	5,10E-07	4,20E-156	1,85E-97
4	3,08E+02	2,27E+02	2,31E-11	3,49E-154	1,68E-44
5	1,48E+02	3,79E+02	7,25E-08	7,11E-156	5,70E-99
PRŮMĚR	2,80E+02	1,40E+02	3,51E-07	1,21E-153	3,35E-45

Tab. 6. Tabulka závislosti SOMA na vstupním parametru PRT

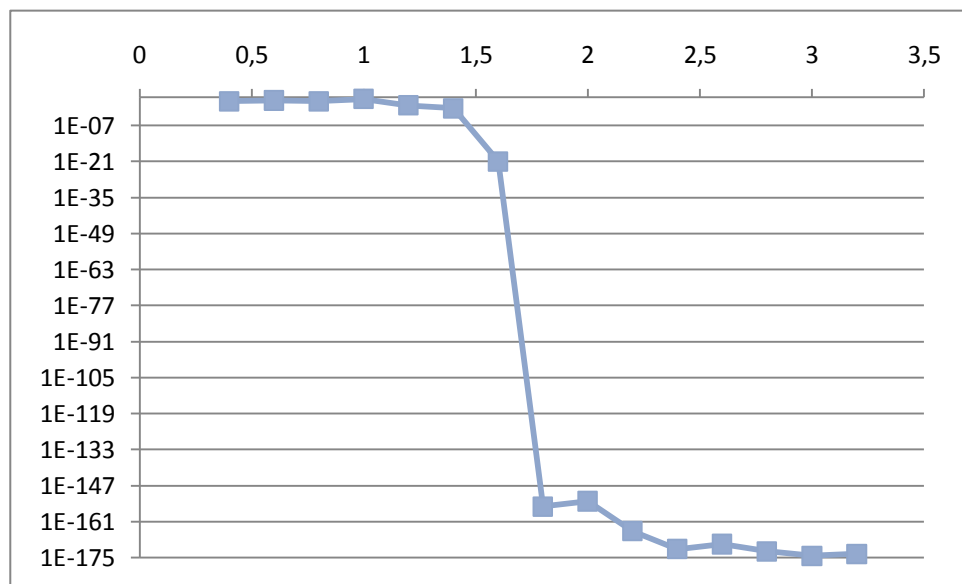
6.3.2 Hledání parametru PathLength

Graf. 8. zobrazuje závislost SOMA AllToOne na parametru PathLength. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 16.). Pro testování byla použita testovací funkce 1st De Jong (viz Obr. 3.) tato funkce má optimum přesně v nule, takže se snadno hodnotí přesnost, neboť samotný výsledek evolučního algoritmu udává nepřesnost.

Z výsledků testování (viz Graf. 8. u kterého musela být použita logaritmická stupnice a z tabulky Tab. 7.) je zřejmé, že hodnoty parametru Step menší než 1,6 silně degradují schopnost algoritmu SOMA nalézt optimum, proto se ani nedoporučuje nastavovat hodnoty menší 1,1. Kolem hodnoty 1,6 zaznamenává algoritmus prudký nárůst a za hodnotou 2,4 se již zase ustaluje. Neboť čím je vyšší hodnota tohoto parametru, tím roste paměťová a výpočetní náročnost a hlavně počet ohodnocení účelové funkce, což je hlavním kritériem při srovnávání různých optimalizačních algoritmů, byla zvolena hodnota 2,4 pro další postup.



Obr. 16. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PathLength



Graf. 8. Graf závislosti SOMA na vstupním parametru PathLength

Hodnota PathLength	0,4	0,6	0,8	1	1,2
Č. měření					
1	3,79E+02	1,23E+03	9,74E+01	3,05E+03	9,45E-03
2	2,56E+02	3,64E+02	2,28E+02	3,26E+03	9,88E+00
3	1,82E+01	1,58E+01	7,07E+02	8,46E+02	2,04E+01
4	9,41E+00	3,54E+00	4,81E+01	1,27E+03	4,53E-01
5	4,91E+02	1,03E+03	2,50E+01	1,83E+03	1,31E-01
PRŮMĚR	2,31E+02	5,29E+02	2,21E+02	2,05E+03	6,18E+00
Hodnota PathLength	1,4	1,6	1,8	2	2,2
Č. měření					
1	0,76967	5,50E-21	6,05E-159	2,66E-157	1,03E-164
2	0,18577	1,53E-78	1,92E-157	4,30E-153	1,24E-169
3	0,03224	3,90E-70	3,99E-155	5,35E-158	6,32E-166
4	0,29404	4,58E-35	1,23E-156	9,46E-161	4,83E-168
5	0,91624	3,08E-57	3,33E-158	3,95E-159	2,20E-170
PRŮMĚR	4,40E-01	1,10E-21	8,28E-156	8,60E-154	2,18E-165
Hodnota PathLength	2,4	2,6	2,8	3	3,2
Č. měření					
1	5,53E-172	6,00E-173	9,34E-176	2,74E-175	1,45E-175
2	1,77E-172	8,79E-173	2,06E-174	1,47E-177	1,45E-178
3	3,93E-172	1,00E-172	1,15E-174	2,39E-174	1,07E-174
4	8,16E-174	2,34E-171	1,20E-172	8,94E-177	4,61E-177
5	4,68E-175	8,66E-170	5,88E-174	1,26E-175	1,21E-173
PRŮMĚR	2,26E-172	1,78E-170	2,59E-173	5,61E-175	2,66E-174

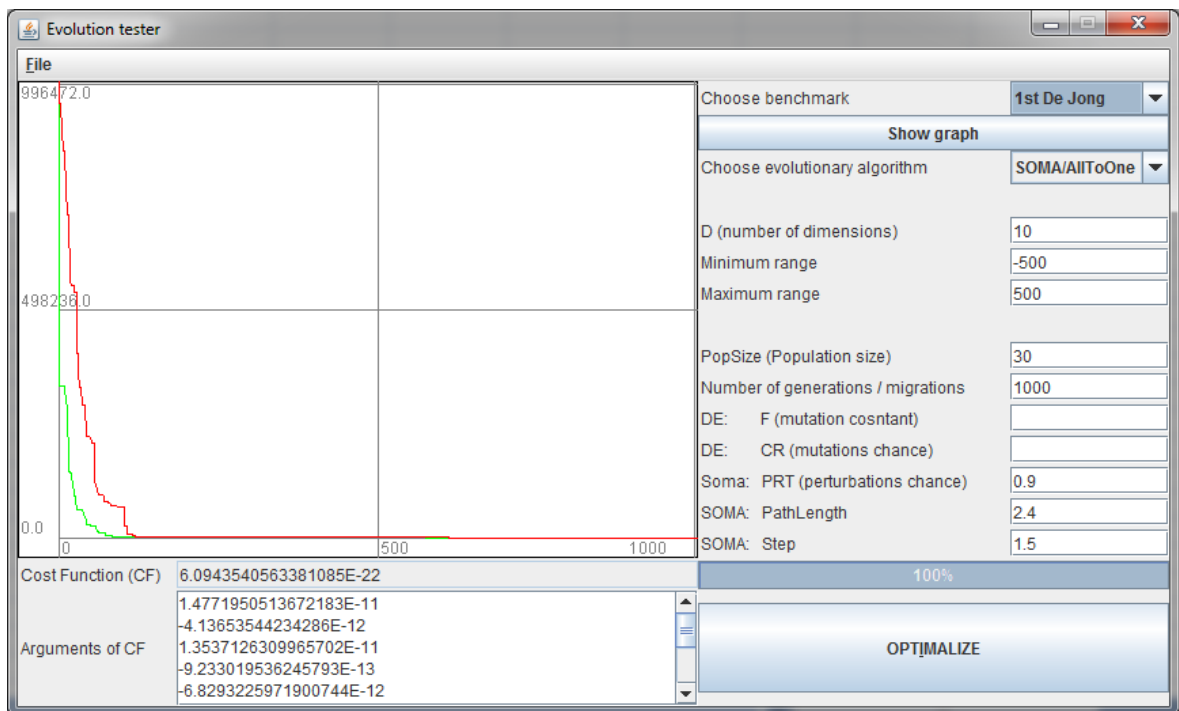
Tab. 7. Tabulka závislosti SOMA na vstupním parametru PathLength

6.3.3 Hledání parametru Step

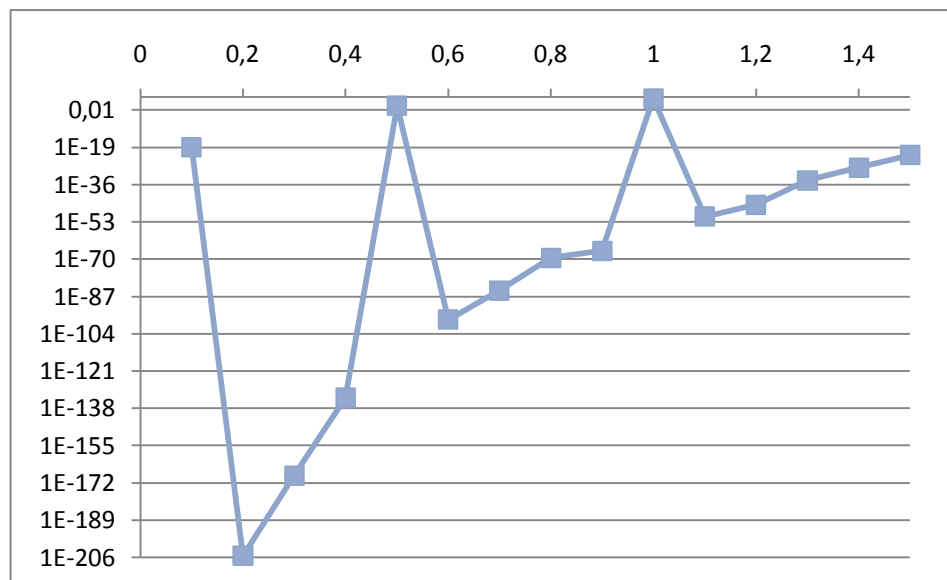
Graf. 9. zobrazuje závislost SOMA AllToOne na parametru Step. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 16.). Pro testování byla použita testovací funkce 1st De Jong (viz Obr. 3.) tato funkce má optimum přesně v nule, takže se snadno hodnotí přesnost, neboť samotný výsledek evolučního algoritmu udává nepřesnost.

Z výsledků testování (viz Graf. 9. u kterého musela být použita logaritmická stupnice a z tabulky Tab. 8) vyplývá překvapující zjištění, že není tak úplně pravd, že čím je hodnota parametru Step menší a tím je tedy prohledávaná hyperplocha zkoumá podrobněji, tím je nalezeno lepší řešení. Nejlepší nastavení parametru Step pro tento případ je tedy někde kolem hodnoty 0,2. Také je zde jasně vidět, že při použití hodnot 0,5 a 1 často dochází

k tomu, že se jedinci zastaví na pozici leadera a dochází tak k silné degradaci algoritmu SOMA.



Obr. 17. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru Step



Graf. 9. Graf závislosti SOMA na vstupním parametru Step

Hodnota Step	0,1	0,2	0,3	0,4	0,5
Č. měření					
1	1,159E-21	4,151E-207	2,198E-171	4,312E-137	1,076E-08
2	4,819E-19	2,633E-205	8,952E-171	5,493E-138	2,281E-06
3	2,349E-23	1,036E-210	9,533E-169	1,627E-135	4,387E+00
4	1,508E-34	3,571E-213	7,551E-172	2,599E-136	2,165E-25
5	8,607E-35	5,665E-213	1,225E-171	2,847E-133	2,004E-01
PRŮMĚR	9,661E-20	5,348E-206	1,933E-169	5,733E-134	9,174E-01
Hodnota Step	0,6	0,7	0,8	0,9	1
Č. měření					
1	6,808E-58	1,340E-49	8,049E-44	4,618E-39	6,762E+02
2	1,099E-58	4,857E-49	2,202E-42	1,151E-40	4,112E+02
3	2,070E-57	2,049E-49	3,934E-43	2,557E-41	3,493E+02
4	4,283E-57	1,665E-48	5,757E-43	7,084E-41	9,430E+02
5	9,735E-58	4,660E-48	1,389E-44	6,373E-42	5,754E+02
PRŮMĚR	1,624E-57	1,430E-48	6,531E-43	9,671E-40	5,910E+02
Hodnota Step	1,1	1,2	1,3	1,4	1,5
Č. měření					
1	6,269E-29	8,073E-25	3,343E-20	1,812E-17	6,367E-13
2	1,818E-27	5,170E-25	4,259E-21	6,965E-17	4,084E-13
3	7,134E-28	9,570E-26	1,178E-20	8,226E-17	3,628E-12
4	1,490E-27	1,905E-26	2,791E-20	1,139E-16	2,472E-12
5	1,491E-28	2,934E-25	3,576E-21	4,422E-17	4,266E-13
PRŮMĚR	8,467E-28	3,465E-25	1,619E-20	6,562E-17	1,514E-12

Tab. 8. Tabulka závislosti SOMA na vstupním parametru Step

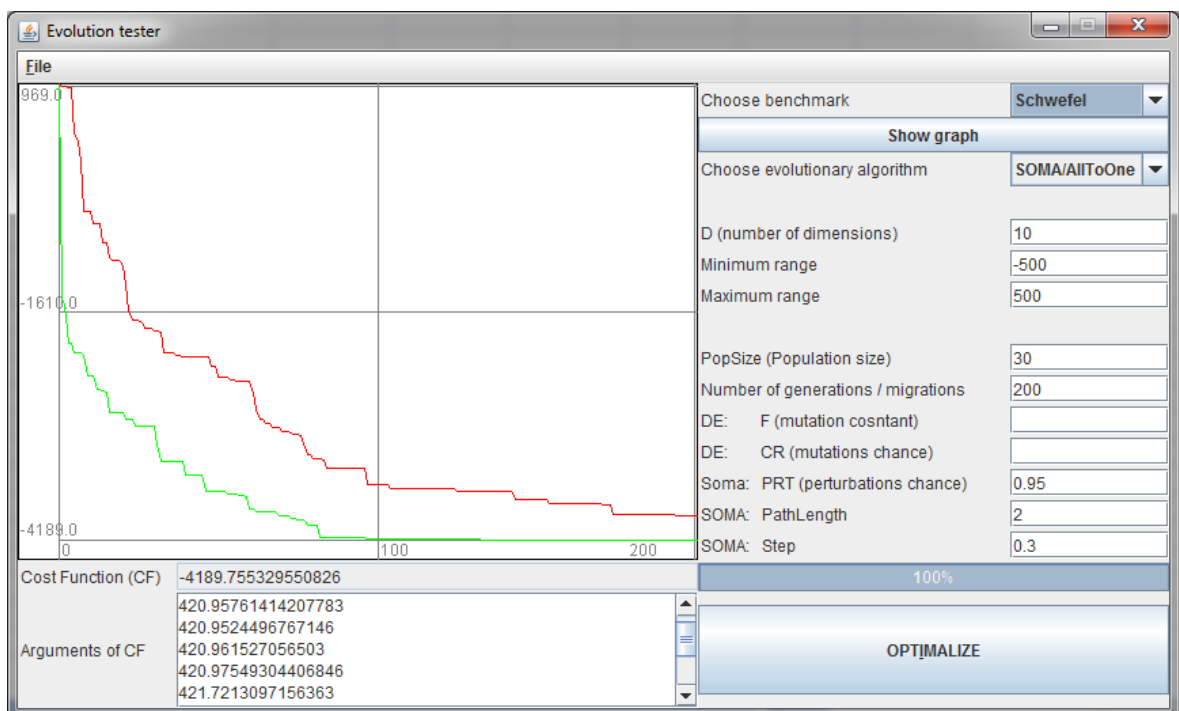
6.4 Ukázka závislosti evolučního algoritmu SOMA AllToOne na vstupních parametrech, testovací funkce Schwefel

Nyní bude zopakován ten samý postup jako u algoritmu DE/Rand/1 u algoritmu SOMA AllToOne k ručnímu nalezení vhodného nastavení. Pro testování byla použita testovací funkce Schwefel (viz Obr. 6.) tato funkce má podle literatury [7] optimum v bodě 420,97 (mé testování tento bod upřesnilo až na 420, 968746) což odpovídá ohodnocení účelové funkce -418, 9828872724338 * počet dimenzí.

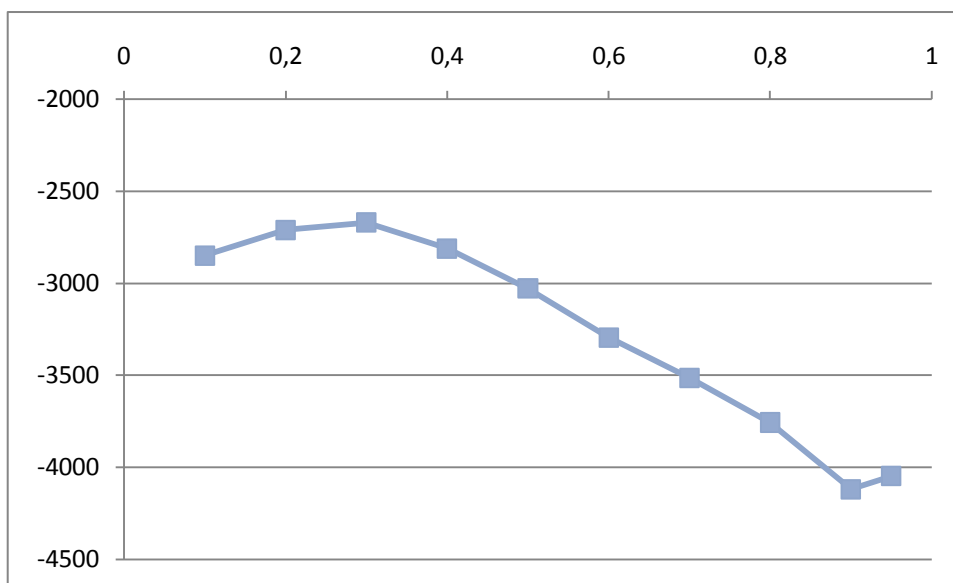
6.4.1 Hledání parametru PRT

Graf. 10. zobrazuje závislost SOMA AllToOne na parametru PRT. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 18.). Pro testování byla použita testovací funkce Schwefel (viz Obr. 6.)

Z výsledků testování je zřejmé, že ideální nastavení tohoto parametru je kolem hodnoty 0,9. (viz Graf. 10. a Tab. 9.).



Obr. 18. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PRT



Graf. 10. Graf závislosti SOMA na vstupním parametru PRT

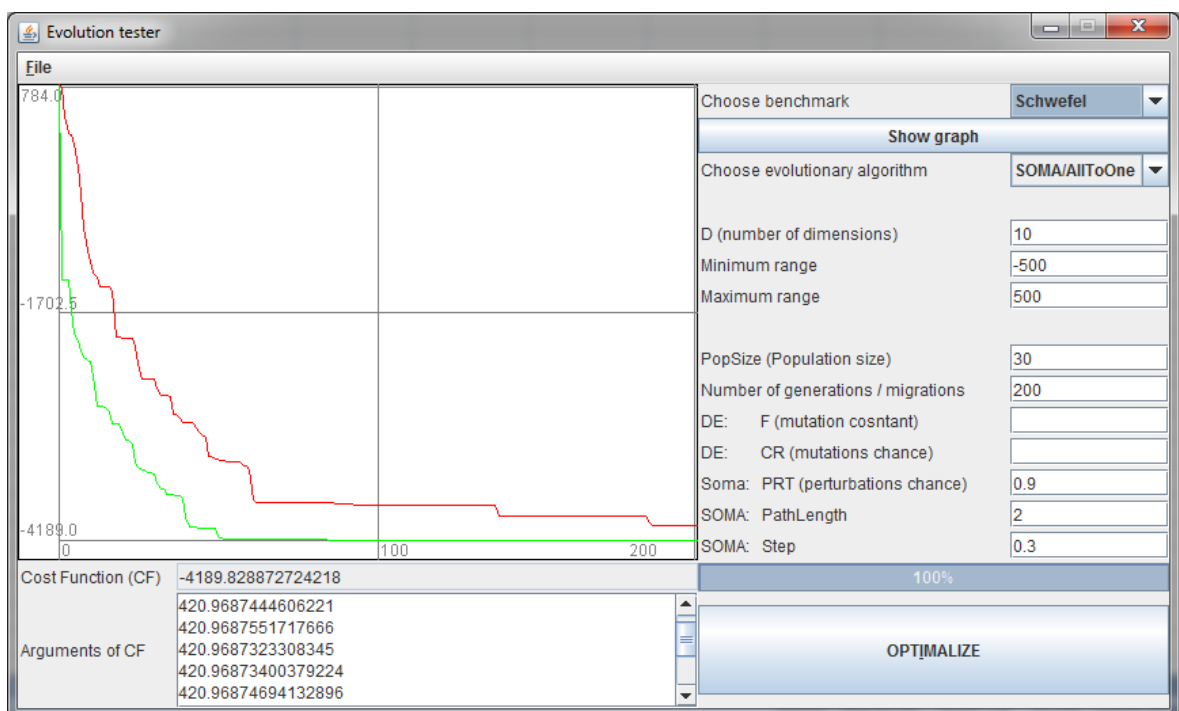
Hodnota CR	0,1	0,2	0,3	0,4	0,5
Č, měření					
1	-2290,4192	-3043,7663	-2581,3967	-3321,1151	-3523,1541
2	-3711,6284	-2752,9432	-2908,2951	-2333,8580	-3429,2793
3	-3275,0210	-2495,2407	-2487,6214	-2945,0917	-2382,4666
4	-2467,5260	-2537,9639	-2817,1355	-3143,0911	-3315,0971
5	-2500,1292	-2720,2239	-2557,8031	-2313,2163	-2488,7795
PRŮMĚR	-2848,9448	-2710,0276	-2670,4503	-2811,2744	-3027,7553
Hodnota CR	0,6	0,7	0,8	0,9	0,95
Č, měření					
1	-3476,5578	-3301,2228	-3735,8117	-4189,8286	-3952,9522
2	-3217,6180	-3952,8309	-4071,3905	-4071,0093	-4071,3797
3	-3532,6734	-3577,5523	-3775,2837	-4069,7433	-4071,3875
4	-3062,0832	-3478,8518	-3676,3624	-4071,1448	-4067,8849
5	-3179,9623	-3255,5290	-3518,4965	-4189,6953	-4070,6501
PRŮMĚR	-3293,7789	-3513,1973	-3755,4690	-4118,2842	-4046,8509

Tab. 9. Tabulka závislosti SOMA na vstupním parametru PRT

6.4.2 Hledání parametru PathLength

Graf. 11. zobrazuje závislost SOMA AllToOne na parametru PathLength. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 19.). Pro testování byla použita testovací funkce Schwefel (viz Obr. 6.)

Z výsledků testování (viz Graf. 11. a z tabulky Tab. 10.) je zřejmé, že hodnoty parametru Step menší než 1,2 silně degradují schopnost algoritmu SOMA nalézt optimum, proto se ani nedoporučuje nastavovat hodnoty menší 1,1. Čím je vyšší hodnota parametru PathLength, tím roste paměťová a výpočetní náročnost a hlavně počet ohodnocení účelové funkce, což je hlavním kritériem při srovnávání různých optimalizačních algoritmů, byla zvolena hodnota 2 pro další postup.



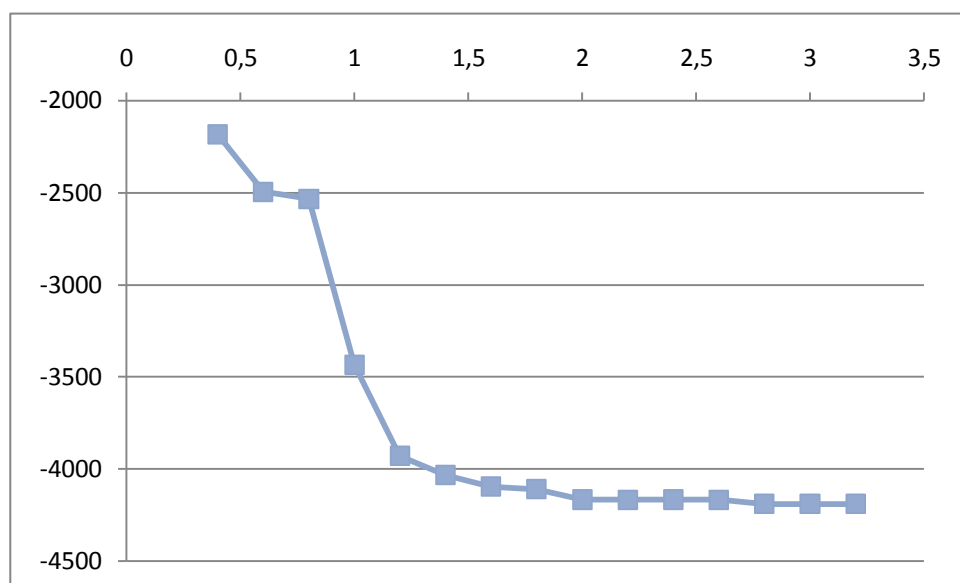
Obr. 19. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PathLength

Hodnota PathLength	0,4	0,6	0,8	1	1,2
Č. měření					
1	-2448,22341	-2504,03114	-2650,54018	-3344,16743	-3834,48068
2	-2289,45073	-2462,81221	-2827,91372	-3564,65752	-4185,50634
3	-2194,58344	-2537,57450	-2452,00822	-3334,58174	-3833,26579
4	-2089,46751	-2477,76703	-2449,11469	-3636,25778	-3906,64784
5	-1896,61078	-2494,88910	-2289,88678	-3290,50408	-3885,31277
PRŮMĚR	-2183,66718	-2495,41480	-2533,89272	-3434,03371	-3929,04268

Hodnota PathLength	1,4	1,6	1,8	2	2,2
Č. měření					
1	-4071,11274	-4071,73627	-4026,56838	-4189,80469	-4189,82887
2	-4065,47438	-4189,57637	-4189,82765	-4189,82887	-4189,82856
3	-4069,16429	-4071,39054	-4070,18250	-4071,30446	-4189,82887
4	-3952,51094	-4071,39052	-4071,39054	-4189,73986	-4071,39054
5	-4008,97303	-4071,33581	-4189,82587	-4189,82887	-4189,82887
PRŮMĚR	-4033,44708	-4095,08590	-4109,55899	-4166,10135	-4166,14114

Hodnota PathLength	2,4	2,6	2,8	3	3,2
Č. měření					
1	-4189,49437	-4189,82887	-4071,39054	-4189,82887	-4189,82887
2	-4189,82777	-4189,82887	-4189,82887	-4189,82887	-4189,82550
3	-4071,39043	-4071,38936	-4189,82887	-4189,82887	-4189,82887
4	-4189,82887	-4189,82871	-4189,82887	-4189,82887	-4189,82887
5	-4189,82887	-4189,82887	-4189,82887	-4189,82887	-4189,82887
PRŮMĚR	-4166,07406	-4166,14094	-4189,82887	-4189,82887	-4189,82820

Tab. 10. Tabulka závislosti SOMA na vstupním parametru PathLength



Graf. 11. Graf závislosti SOMA na vstupním parametru PathLength

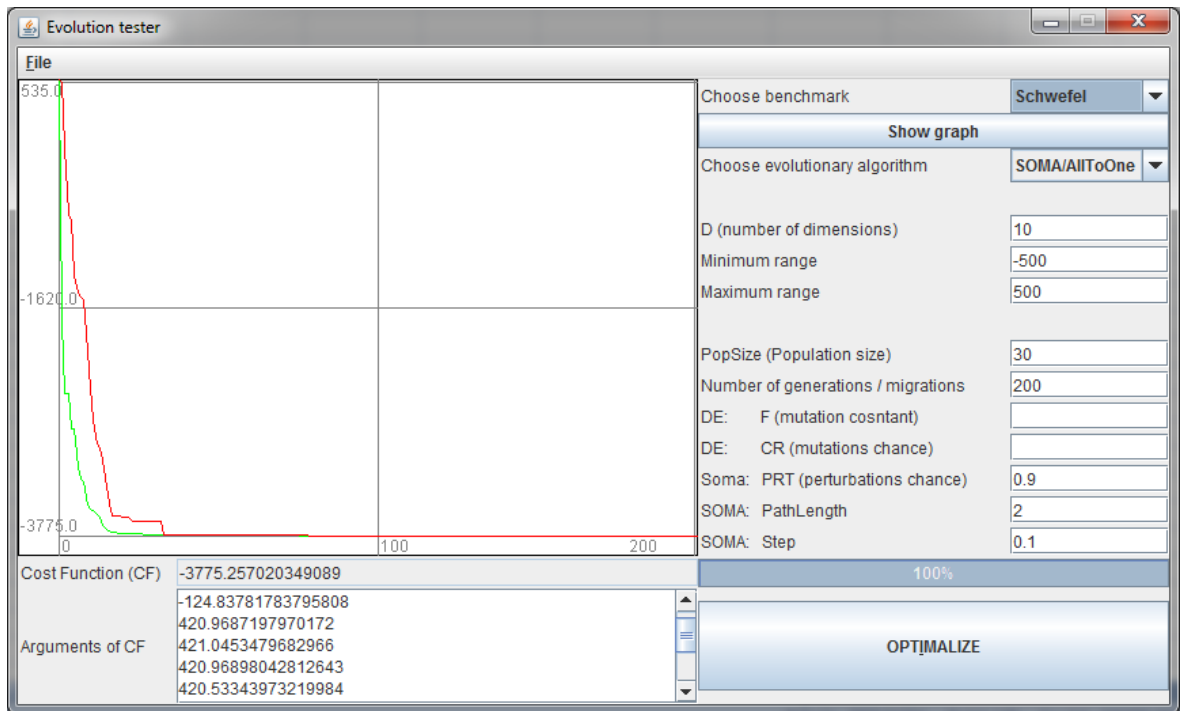
6.4.3 Hledání parametru Step

Graf. 12. zobrazuje závislost SOMA AllToOne na parametru Step. Všechny ostatní parametry zůstaly po celou dobu testování nezměněny (viz Obr. 20.). Pro testování byla použita testovací funkce Schwefel (viz Obr. 6.) tato funkce má optimum přesně v nule, takže se snadno hodnotí přesnost, neboť samotný výsledek evolučního algoritmu udává nepřesnost.

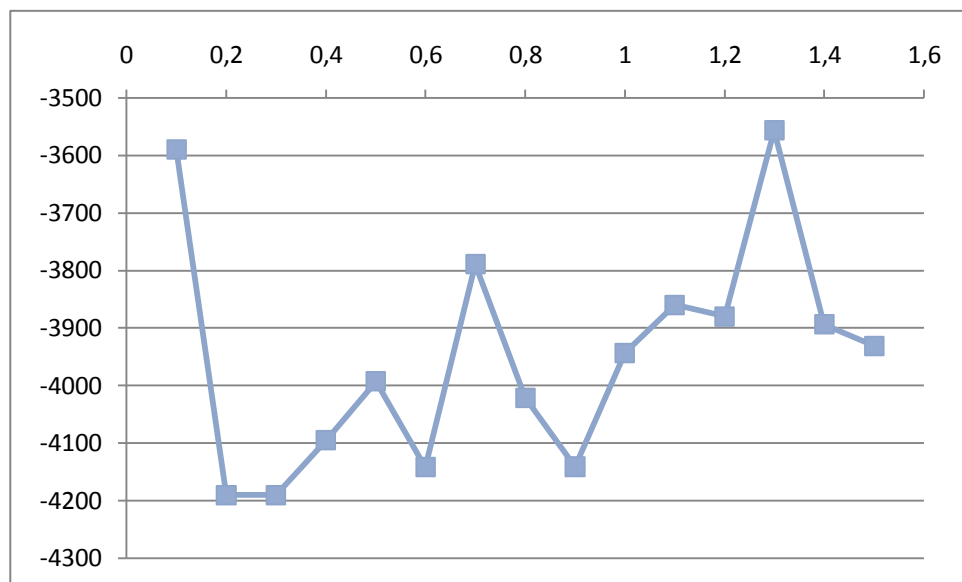
Z výsledků testování (viz Graf. 12. u kterého musela být použita logaritmická stupnice a z tabulky Tab. 11) vyplývá překvapující zjištění, že není tak úplně pravda, že čím je hodnota parametru Step menší a tím je tedy prohledávaná hyperplocha zkoumá podrobněji, a tím je nalezeno lepší řešení. Nejlepší nastavení parametru Step pro tento případ je tedy někde mezi hodnotami 0,2 a 0,3.

Hodnota Step	0,1	0,2	0,3	0,4	0,5
Č, měření					
1	-3854,24720	-4189,82887	-4189,82743	-4071,39053	-3952,71106
2	-3716,07369	-4189,82887	-4189,82887	-4071,39054	-3944,74007
3	-3597,60219	-4189,82887	-4189,82887	-4189,82887	-3834,51387
4	-3419,95706	-4189,82886	-4189,82885	-3952,95220	-4040,50970
5	-3360,73802	-4189,82887	-4189,82886	-4189,82887	-4189,82887
PRŮMĚR	-3589,72363	-4189,82887	-4189,82858	-4095,07820	-3992,46071
Hodnota Step	0,6	0,7	0,8	0,9	1
Č, měření					
1	-4189,308	-4148,865	-4182,254	-4183,947	-3901,447
2	-4071,381	-3858,788	-4033,903	-4071,005	-3813,438
3	-4187,214	-4013,512	-3703,679	-4187,459	-4096,051
4	-4189,828	-3428,483	-4067,629	-4071,388	-3967,661
5	-4071,292	-3494,548	-4118,214	-4189,828	-3936,242
PRŮMĚR	-4141,805	-3788,839	-4021,136	-4140,725	-3942,968
Hodnota Step	1,1	1,2	1,3	1,4	1,5
Č, měření					
1	-3333,134	-3873,494	-3563,569	-4055,632	-3749,244
2	-4027,408	-4025,187	-3717,497	-4058,495	-4142,107
3	-3819,557	-4011,501	-3339,719	-3927,705	-3864,148
4	-4174,357	-3549,341	-3720,101	-3528,327	-4064,090
5	-3943,158	-3940,379	-3443,087	-3898,534	-3834,543
PRŮMĚR	-3859,523	-3879,981	-3556,794	-3893,739	-3930,827

Tab. 11. Tabulka závislosti SOMA na vstupním parametru Step



Obr. 20. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru Step



Graf. 12. Graf závislosti SOMA na vstupním parametru Step

6.5 Shrnutí výsledků testování

Pro lepší srovnání byly nejlepší výsledky, získané během testování, zaznamenány do tabulky Tab. 12 a Tab. 14. Z výsledných údajů vyplývá, že algoritmem SOMA AllToOne bylo dosaženo lepších výsledků což je sice pravda, ale algoritmus SOMA AllToOne potřeboval k dosažení tohoto výsledku více ohodnocení účelové funkce. Pokud vyrovnáme počet ohodnocení účelové funkce tím, že přidáme počet generací před ukončením algoritmu DE/Rand/1 (viz Tab. 13.) zjistíme, že Diferenciální evoluce dosáhla lepšího výsledku u testovací funkce 1st De Jong. U funkce Schwefel to podle tabulek vypadá, že oba algoritmy našly totožné řešení, avšak není tomu tak. To pouze testovací program nevypisuje s dostatečně velkým počtem desetinných míst.

	CR	F	Hodnota účelové funkce
DE/Rand/1 1st De Jong	0,2	0,2	7,28E-46
DE/Rand/1 Schwefel	0,4	0,4	-4189,63204

Tab. 12. Shrnutí výsledků testování

	CR	F	Hodnota účelové funkce
DE/Rand/1 1st De Jong	0,2	0,2	3,90E-257
DE/Rand/1 Schwefel	0,4	0,4	-4189,828872724338

Tab. 13. Shrnutí výsledků testování

	PRT	PathLength	Step	Hodnota účelové funkce
SOMA AllToOne 1st De Jong	0,9	2,4	0,2	5,35E-206
SOMA AllToOne Schwefel	0,9	2	0,2	-4189,828872724338

Tab. 14. Shrnutí výsledků testování

ZÁVĚR

V úvodu této diplomové práce bylo nejprve nutné vytvořit literární rešerši na téma evolučních algoritmů. Popsat a vysvětlit dělení optimalizačních algoritmů a definovat vstupní parametry, se kterými algoritmy pracují. Dále je zde popsán princip činnosti algoritmů implementovaných do knihovny programu. Těmito algoritmy jsou SOMA v modifikaci AllToOne a Diferenciální Evoluce v modifikacích DE/Rand/1, DE/Rand/2, DE/Best/1, DEBest/2, a popsát použité testovací funkce.

Cílem praktické části diplomové práce bylo naprogramovat knihovnu s několika vybranými evolučními optimalizačními algoritmy a interpreter, který bude s touto knihovnou pracovat. Na konci praktické části byla otestována funkčnost vybraných algoritmů.

Výsledný program musel být napsán multiplatformě, proto byl zvolen programovací jazyk Java a při jeho vývoji byly použity pouze standardní knihovny. Program musel být napsán takovým způsobem, aby do něj bylo později možné co nejsnadněji přidávat další evoluční algoritmy. Jak již bylo zmíněno výše, diplomová práce byla pojata jako započítání knihovny různých evolučních algoritmů a výroba interpreteru, který bude s touto knihovnou dále pracovat. Výsledná aplikace je tedy pojata jako nástroj pro srovnávání a testování různých evolučních algoritmů. Výsledný program je tedy takový základní modul, na který je možné snadno navazovat dalšími prvky, ať je to přidání dalšího evolučního algoritmu, další testovací funkce, nebo jen dalších prvků usnadňující testování, například aby program udělal více evolucí a výsledek zprůměroval nebo vrátil ve formě tabulky, nebo zobrazení 3D grafu testovací funkce.

V praktické části diplomové práce je nejprve popsán výsledný program, rozvržení ovládacích prvků grafického rozhraní, rozdělení zdrojového kódu do tříd a stručný popis jak přidat nový evoluční algoritmus do aplikace, aby plně využíval již implementovaných prvků jako je například graf konvergence nebo ukazatel průběhu.

V druhé polovině praktické části je porovnání dvou implementovaných evolučních algoritmů. Těmito algoritmy jsou Diferenciální evoluce a SOMA. Výsledky testování potvrzují, že neexistuje ideální nastavení vstupních parametrů evolučního algoritmu.

ZÁVĚR V ANGLIČTINĚ

At the beginning of this thesis was first necessary to create a literature search on the topic of evolutionary algorithms. Describe and explain the division of optimization algorithms and define input parameters with which the algorithms work. Then there is the principle of operation described algorithms implemented in the library. These algorithms are modified in SOMA AllToOne and Differential Evolution DE/Rand/1 modifications, DE/Rand/2, DE/Best/1, DEBest / 2, and describe the used test functions.

The aim of the practical part was programmed with a library of some selected evolutionary optimization algorithms and interpreter, who will work with this library. At the end of the practical test the functionality of selected algorithms.

The resulting program had to be written multiplatform, because the Java programming language and its development, we used only the standard library. The program had to be written in such a way that it was later to be as easy to add other evolutionary algorithms. As mentioned above, the thesis has been conceived as a starting library of different evolutionary algorithms and production of the interpreter, who will work with the library to work. The resulting application is conceived as a tool for comparing and testing different evolutionary algorithms. The resulting program is such a basic module, which can be easily followed by other elements, whether it's adding another evolutionary algorithm, additional test function, or just another element to facilitate testing, for example, the program has done more the result of evolution and averaged the scores and returned in the form of a table or 3D chart display test function.

The practical part of the thesis is first described the resulting program to control layout graphical user interface, the distribution of source code into classes and a brief description of how to add a new evolutionary algorithm to applications to take full advantage of already implemented features such as convergence or chart the progress bar.

In the second half of the practical part is a comparison of two evolutionary algorithms implemented. These algorithms are the differential evolution and SOMA. The test results confirm that there is no ideal set of input parameters of the evolutionary algorithm.

SEZNAM POUŽITÉ LITERATURY

- [1] KALÁTOVÁ, Eva; DOBIÁŠ, Jaroslav. Evoluční algoritmy [online]. [s.l.] : [s.n.], Duben 2000 [cit. 2011-04-24]. Dostupné z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/gen_alg2/index.htm>.
- [2] PAVLOVIČ, Jan. Multikriteriální hybridní evoluční algoritmy pro výběr a optimalizaci dekontaminačních technologií [online]. 2006. Teze Dizertační práce. MASARYKOVA UNIVERZITA. Vedoucí práce prof. RNDr. Jiří Hřebíček, CSc. Dostupné z WWW: <http://is.muni.cz/th/4035/fi_r/teze_extended.txt>.
- [3] KOUKAL, Jiří. Evoluční teorie. Jirikoukal.com : vše je jen otázka úhlu pohledu [online]. Duben 2006, [cit. 2011-04-26]. Dostupný z WWW: <<http://www.jirikoukal.com/download/files/veznovodilema.pdf>>.
- [4] BELOV, Nikita. Umělá inteligence : Genetické programování. In Referáty MPF [online]. [cit. 2011-04-26]. Dostupné z WWW: <<http://sites.google.com/site/belov222/home/01/geneticke-programovani>>.
- [5] GAJDA, Zbyšek. EVOLUTIONARY DESIGN OF FRACTAL IMAGES [online]. [s.l.], 2004. 70 s. Ročníkový projekt. Vysoké učení technické v Brně.
- [6] PAVLOVIČ, Jan. Multikriteriální hybridní evoluční algoritmy pro výběr a optimalizaci dekontaminačních technologií [online]. [s.l.], 2006. 28 s. Teze Dizertační práce. MASARYKOVA UNIVERZITA.
- [7] ZELINKA, Ivan; OPLATKOVÁ, Zuzana; ŠEDA, Miloš; OŠMERA, Pavel; VČELAŘ, František. Evoluční výpočetní techniky : Principy a aplikace. [s.l.] : BEN, 2009. 534 s. ISBN 978-80-7300-218-3.
- [8] Zelinka I., Umělá inteligence v problémech globální optimalizace, BEN, Praha, 2002, ISBN 80-7300-069-5

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EA	Evoluční Algoritmy
ESS	Evolučně Stabilní Strategie -- Evolution Stable Strategy
GA	Genetické Algoritmy
NFL	No Free Lunch Teorém
SOMA	Samo – Organizující se Migrační Algoritmus
DE	Diferenciální evoluce
D	Dimenze
CR	Práh Křížení
F	Mutační konstanta
PRT	Perturbace

SEZNAM OBRÁZKŮ

Obr. 1. Princip SOMA – převzato [8]

Obr. 2. Princip DE – převzato z [8]

Obr. 3. 1st De Jong

Obr. 4. 3rd De Jong

Obr. 5. Rastrigin

Obr. 6. Schwefel

Obr. 7. Vývojové prostředí *Eclipse IDE for Java Developers* ve verzi *Helios Service Release 2*

Obr. 8. Grafické rozhraní s grafem testovací funkce

Obr. 9. Grafické rozhraní s grafem konvergence

Obr. 10. Výstup aplikace uložený do souboru

Obr. 11. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru CR

Obr. 12. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru F

Obr. 13. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru CR

Obr. 14. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru F

Obr. 15. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PRT

Obr. 16. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PathLength

Obr. 17. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru Step

Obr. 18. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PRT

Obr. 19. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru PathLength

Obr. 20. Nastavení parametrů pro ruční nalezení ideálního nastavení parametru Step

SEZNAM TABULEK

- Tab. 1. Složitost problémů pro x vstupních parametrů
- Tab. 2. Tabulka závislosti Diferenciální evoluce na vstupním parametru CR
- Tab. 3. Tabulka závislosti Diferenciální evoluce na vstupním parametru F
- Tab. 4. Tabulka závislosti Diferenciální evoluce na vstupním parametru CR
- Tab. 5. Tabulka závislosti Diferenciální evoluce na vstupním parametru F
- Tab. 6. Tabulka závislosti SOMA na vstupním parametru PRT
- Tab. 7. Tabulka závislosti SOMA na vstupním parametru PathLength
- Tab. 8. Tabulka závislosti SOMA na vstupním parametru Step
- Tab. 9. Tabulka závislosti SOMA na vstupním parametru PRT
- Tab. 10. Tabulka závislosti SOMA na vstupním parametru PathLength
- Tab. 11. Tabulka závislosti SOMA na vstupním parametru Step
- Tab. 12. Shrnutí výsledků testování
- Tab. 13. Shrnutí výsledků testování
- Tab. 14. Shrnutí výsledků testování

SEZNAM GRAFŮ

Graf. 1. Funkce typu „hledání jehly v kupce sena“

Graf. 2. Složitost problémů pro x vstupních parametrů

Graf. 3. Graf závislosti Diferenciální evoluce na vstupním parametru CR

Graf. 4. Graf závislosti Diferenciální evoluce na vstupním parametru F

Graf. 5. Graf závislosti Diferenciální evoluce na vstupním parametru CR

Graf. 6. Graf závislosti Diferenciální evoluce na vstupním parametru F

Graf. 7. Graf závislosti SOMA na vstupním parametru PRT

Graf. 8. Graf závislosti SOMA na vstupním parametru PathLength

Graf. 9. Graf závislosti SOMA na vstupním parametru Step

Graf. 10. Graf závislosti SOMA na vstupním parametru PRT

Graf. 11. Graf závislosti SOMA na vstupním parametru PathLength

Graf. 12. Graf závislosti SOMA na vstupním parametru Step

SEZNAM PŘÍLOH

PŘÍLOHA P I: NÁZEV PŘÍLOHY