

# **PSO Algoritmus v prostředí Mathematica**

PSO Algorithm in the Mathematica environment

Bc. Michal Pluháček



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2010/2011

## ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Michal PLUHÁČEK**

Osobní číslo: **A09486**

Studijní program: **N 3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Téma práce: **PSO algoritmus v prostředí Mathematica**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Naprogramujte PSO algoritmus v prostředí Mathematica.
3. Proveďte prostorovou a časovou optimalizaci kódu.
4. Otestujte algoritmus na sadě vybraných testovacích funkcí.
5. Výsledky testování přehledně graficky a tabulkově zobrazte.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. DORIGO M., Ant Colony Optimization and Swarm Intelligence, Springer, 2006, ISBN 35402267292.
2. HOSTE, Jim. Mathematica DeMYSTiFied. McGraw-Hill Professional, 2008. 408 s. ISBN 978-0071591447.
3. ZELINKA, I. Umělá inteligence - hrozba nebo naděje. BEN - technická literatura, 2003, ISBN 80-7300-068-7.
4. ZELINKA I., OPLATKOVÁ Z., ŠEDA M., OŠMERA P., VČELAŘ F., Evoluční výpočetní techniky - principy a aplikace, BEN, Praha, 2008, ISBN 80-7300-218-3.
5. RUSKEEPAA, Heikki. Mathematica Navigator: Mathematics, Statistics and Graphics, Third Edition. Academic Press, 2009. 1136 s. ISBN 978-0123741646.
6. ZELINKA, Ivan. Aplikovaná Informatika. Zlín. UTB, 1999. 183 s. ISBN 80-214-1423-5.
7. ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. BEN, 2002, 190 s. ISBN 80-7300-069-5.
8. EBERHART R., KENNEDY J., Swarm Intelligence, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, 2001, ISBN 15586059593.

Vedoucí diplomové práce:

**Ing. Roman Šenkeřík, 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**

Cílem práce je implementace evolučního algoritmu PSO v prostředí Mathematica a jeho otestování na testovacích funkcích. Výstupem práce bude knihovna (m-file) pro použití ve výzkumu a výuce na Univerzitě Tomáše Bati ve Zlíně. Teoretická část je zaměřena na problematiku evolučních algoritmů, představuje algoritmus PSO, použité testovací funkce a vývojové prostředí Wolfram Mathematica. Praktická část obsahuje výsledky testování zvolených strategií PSO na vybraných testovacích funkcích. Výsledky jsou přehledně zobrazeny v tabulkách a grafech.

Klíčová slova: PSO, evoluční algoritmy, Wolfram Mathematica, optimalizace

## **ABSTRACT**

The goal of this work is an implementation of the PSO evolutionary algorithm in the Mathematica environment, and tests of the algorithm by test functions. The output of the work is an m-file library usable in research and education at the Tomas Bata University in Zlin. The theoretical part presents the general theory of evolutionary algorithms; it also introduces the PSO algorithm, test functions used, and the Wolfram Mathematica environment. The practical part presents the results of testing selected PSO strategies by specified test functions. The results are arranged in tables and charts.

Keywords: PSO, evolutionary algorithms, Wolfram Mathematica, optimization

Děkuji vedoucímu práce Ing. Romanu Šenkeříkovi Ph.D. za pomoc, cenné rady a trpělivost při konzultacích.

**Motto:**

*Intelligentní lidé se snaží problémy řešit,  
geniální se je snaží nedělat!*

Albert Einstein

**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**

<b>ÚVOD.....</b>	<b>10</b>
<b>I TEORETICKÁ ČÁST.....</b>	<b>11</b>
<b>1 EVOLUČNÍ ALGORITMY .....</b>	<b>12</b>
1.1 NO FREE LUNCH TEORÉM .....	12
1.2 SLOŽITÉ OPTIMALIZAČNÍ PROBLÉMY .....	13
1.2.1 Obchodní cestující.....	13
1.2.2 Problém batohu .....	14
1.3 PRINCIPY VYBRANÝCH EVOLUČNÍCH OPTIMALIZAČNÍCH ALGORITMŮ .....	14
1.3.1 Ant Colony Optimization .....	14
1.3.2 Diferenciální evoluce .....	15
1.3.3 SOMA .....	15
<b>2 PSO .....</b>	<b>17</b>
2.1 SWARM INTELLIGENCE – INTELIGENCE HEJNA.....	17
2.2 ZÁKLADNÍ POJMY A PRINCIP PSO.....	18
2.3 PARAMETRY PSO .....	19
2.3.1 Dimenze .....	19
2.3.2 Rozsah .....	19
2.3.3 Specimen .....	19
2.3.4 Počet částic.....	20
2.3.5 Počet iterací.....	20
2.3.6 Učící faktory.....	20
2.4 MODIFIKACE PSO ALGORITMU .....	20
2.4.1 Omezení maximální rychlosti $V_{\max}$ .....	20
2.4.2 Setrvačnost $w$ .....	21
2.4.3 Constriction faktor .....	21
<b>3 TESTOVACÍ FUNKCE.....</b>	<b>23</b>
3.1 UNIMODÁLNÍ FUNKCE .....	23
3.1.1 1. De Jongova funkce.....	23
3.1.2 3. De Jongova funkce.....	24
3.1.3 4. De Jongova funkce.....	26
3.2 MULTIMODÁLNÍ FUNKCE.....	27
3.2.1 2. De Jongova funkce.....	27
3.2.2 Schwefelova funkce .....	28
3.2.3 Rastriginova funkce .....	29
<b>4 WOLFRAM MATHEMATICA .....</b>	<b>32</b>
4.1 VÝVOJOVÉ PROSTŘEDÍ .....	32
4.1.1 Náповěda.....	33
4.2 HISTORIE VÝVOJE.....	34
<b>II PRAKTICKÁ ČÁST .....</b>	<b>35</b>
<b>5 IMPLEMENTACE PSO ALGORITMU .....</b>	<b>36</b>
5.1 PŘEHLED VERZÍ .....	36
5.1.1 Seznam vytvořených verzí: .....	36

5.2	ZÁKLADNÍ STRUKTURA .....	37
5.2.1	Vstupy .....	37
5.2.2	Lokální proměnné .....	38
5.2.3	Algoritmus.....	38
5.2.4	Výstup .....	39
5.3	SPECIFIKA .....	39
5.3.1	PSO_BASE_VMAX .....	39
5.3.2	PSO_BASE_CONS.....	39
5.3.3	PSO_BASE_WEIGHT .....	40
5.3.4	PSO_ADV_VMAX.....	40
<b>6</b>	<b>TESTOVÁNÍ .....</b>	<b>41</b>
6.1	TESTOVÁNÍ BASE VERZÍ ALGORITMU .....	41
6.1.1	1. De Jongova funkce .....	42
6.1.1.1	Naměřená data .....	42
6.1.1.2	Vyhodnocení .....	44
6.1.2	2. De Jongova funkce .....	44
6.1.2.1	Naměřená data .....	44
6.1.2.2	Vyhodnocení .....	45
6.1.3	3. De Jongova funkce .....	45
6.1.3.1	Naměřená data .....	46
6.1.3.2	Vyhodnocení .....	47
6.1.4	4. De Jongova funkce .....	48
6.1.4.1	Naměřená data .....	48
6.1.4.2	Vyhodnocení .....	50
6.1.5	Rastriginova funkce .....	50
6.1.5.1	Naměřená data .....	50
6.1.5.2	Vyhodnocení .....	52
6.1.6	Schwefelova funkce .....	52
6.1.6.1	Naměřená data .....	52
6.1.6.2	Vyhodnocení .....	54
6.2	CELKOVÉ VYHODNOCENÍ TESTOVÁNÍ BASE VERZÍ.....	55
6.3	TESTOVÁNÍ ADVANCED VERZÍ ALGORITMU .....	56
6.3.1	Testování PSO_ADV .....	57
6.3.2	Testování PSO_ADV_CONS .....	58
6.3.3	Testování PSO_ADV_WEIGHT .....	59
6.3.4	Testování PSO_ADV_Vmax .....	60
<b>7</b>	<b>SOUHRN PRAKTICKÉ ČÁSTI .....</b>	<b>62</b>
7.1	ČASOVÁ A PROSTOROVÁ OPTIMALIZACE .....	62
7.2	VYTVOŘENÉ SOUBORY .....	63
7.3	SPUŠTĚNÍ A PROHLÍŽENÍ KÓDU .....	65
7.3.1	Wolfram Mathematica .....	65
7.3.2	Wolfram CDF Player .....	65
7.3.3	PDF .....	65
	<b>ZÁVĚR .....</b>	<b>66</b>
	<b>CONCLUSION .....</b>	<b>67</b>
	<b>SEZNAM POUŽITÉ LITERATURY.....</b>	<b>68</b>



<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....</b>	<b>69</b>
<b>SEZNAM OBRÁZKŮ .....</b>	<b>70</b>
<b>SEZNAM TABULEK.....</b>	<b>71</b>
<b>SEZNAM PŘÍLOH.....</b>	<b>74</b>

## ÚVOD

Význam optimalizace je dlouhodobě znám, ale až rozvoj výpočetní techniky umožnil vyvinutí vysoce efektivních optimalizačních algoritmů, které jsou schopny uspokojivě řešit i velmi složité optimalizační úlohy v přijatelném čase. Významnou třídou optimalizačních technik jsou tzv. evoluční algoritmy. Tyto jsou inspirovány evoluční teorií a z přírody odpozorovanými principy.

PSO (Particle Swarm Optimization) je evoluční optimalizační algoritmus inspirovaný chováním ptačího hejna. Cílem této práce je implementace PSO v prostředí Wolfram Mathematica a vytvoření knihovny pro potřeby výzkumu a výuky na Univerzitě Tomáše Bati ve Zlíně.

Teoretická část čtenáře uvádí do problematiky evolučních algoritmů a optimalizačních problémů. Jsou vysvětleny základní principy algoritmů SOMA, DE a ACO a dvou známých optimalizačních problémů. Podrobně je popsán algoritmus PSO a jeho vybrané modifikace.

Připojen je popis několika testovacích funkcí, které se využívají k ověření funkčnosti optimalizačních algoritmů. Poslední kapitola teoretické části představuje vývojové prostředí Wolfram Mathematica

Praktická část popisuje vytvořený programový výstup a zachycuje pomocí tabulek a grafů výsledky testování programu za použití testovacích funkcí. Jsou zmíněna pravidla časové a prostorové optimalizace kódu, dle kterých byl výsledný algoritmus vytvářen.

Velký důraz je kladen na praktickou využitelnost výsledků práce.

## **I. TEORETICKÁ ČÁST**

## 1 EVOLUČNÍ ALGORITMY

Evoluční algoritmy (EA) jsou třídou algoritmu, která svým principem vychází z evoluční teorie přirozeného výběru, tak jak ji popsal Charles Darwin ve své práci „O původu druhů“ (viz [11]). A z navazujících výzkumů v této oblasti.

Typickým rysem pro evoluční algoritmy je, že pracují s tzv. populacemi možných řešení, jimž se říká jedinci. Tito jedinci navzájem ovlivňují svou kvalitu na základě určitých evolučních principů v cyklech, které obvykle nesou jméno „Generace“. Cílem celého evolučního procesu je nalézt nejlepší řešení [7].

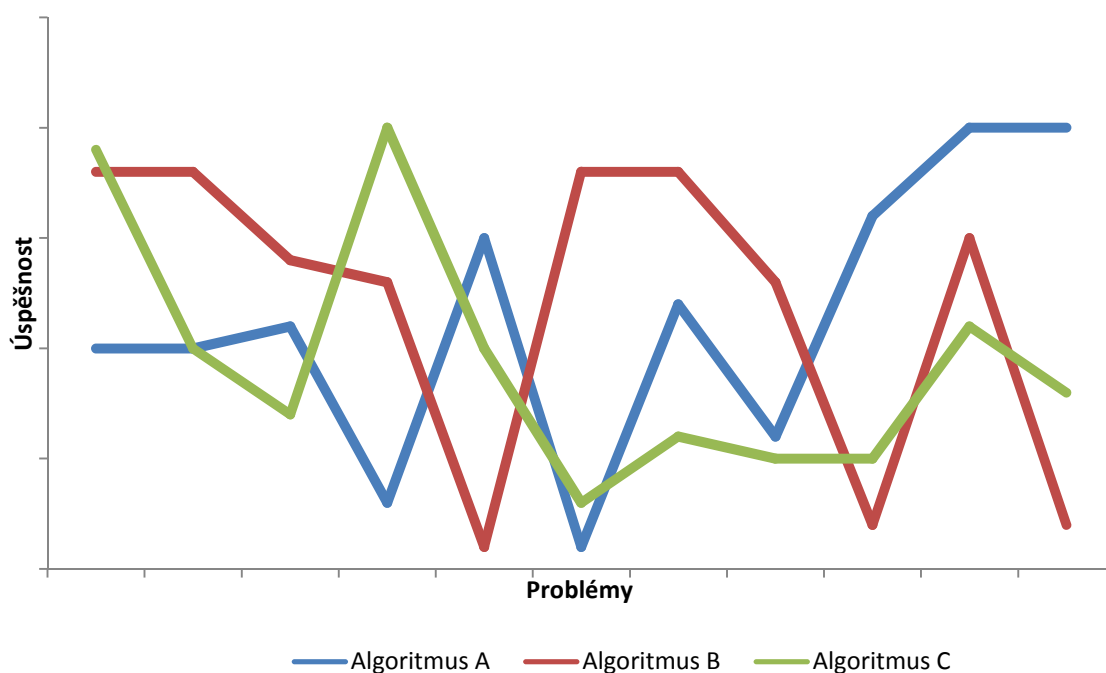
Základní myšlenkou EA je vývoj populace jedinců do nové (kvalitnější) generace s využitím evolučních pravidel známých z přírody. Dle Darwinovy teorie v přírodě přežijí pouze ti nejsilnější jedinci, obdobně v průběhu EA jsou do další generace obvykle připuštěni pouze nejvyšší jedinci. Kvalitu (někdy též vhodnost, fitness) určuje hodnota účelové funkce, což je matematický zápis řešeného problému.

Velké uplatnění nachází evoluční algoritmy v oblasti řešení optimalizačních problémů s vysokou časovou složitostí, které jsou klasickými analytickými metodami neřešitelné v přijatelném čase, nebo neřešitelné zcela, vzhledem ke složitosti nebo fyzikálním omezením výpočetní techniky.

### 1.1 No Free Lunch Teorém

Důležitým pojmem v oblasti optimalizačních algoritmů je tzv. No Free Lunch Teorém (NFL). Je to teorém, ve kterém se tvrdí, že neexistuje algoritmus, který by dokázal řešit všechny problémy lépe než jiné algoritmy, neboli existuje podmnožina problémů, pro které je algoritmus A lepší než algoritmus B [4].

Právě díky tomuto jevu, který vylučuje použití jediného algoritmus pro všechny problémy, existuje velká řada různých optimalizačních (evolučních) algoritmů a jejich modifikací, neboť množina řešených problémů je velmi rozsáhlá a různorodá. Zjednodušené grafické vyjádření NFL je na obr. 1.



Obr. 1 Grafické vyjádření No Free Lunch Teorému

## 1.2 Složité optimalizační problémy

Pro pochopení významu evolučních algoritmů a dalších optimalizačních technik, je třeba vysvětlit, že některé problémy jsou neřešitelné tzv. metodou hrubé síly, tedy otestováním všech možných kombinací parametrů a to ani v případě využití nejlepší výpočetní techniky, protože:

Existují limity omezující výkon jakéhokoliv počítače, které plynou z kvantově-mechanické povahy hmoty. Tyto limity omezují jak výkon počítače, tak jeho paměť. Z těchto omezení je jasné, že k řešení některých problémů, nepomůže žádný počítač [4].

Typickými příklady úloh s vysokou složitostí jsou problémy Obchodního cestujícího (1.2.1) a Batohu (1.2.2).

### 1.2.1 Obchodní cestující

Problém je definován jako hledání nejkratší spojnice  $n$  měst, kde výchozí a koncové město jsou stejné a vzdálenost mezi městy je známá. Množství možných cest pro  $n$  měst je  $n!$ . Tato složitost, je překážkou použití enumerativních postupů pro větší počty měst, neboť by velmi brzy čas potřebný k výpočtu byl neúnosný. Příkladně, pokud výpočet pro 9 měst trvá pouhých 9 sekund, pak by výpočet pro 10 měst zabral 90 sekund a výpočet pro 11 měst by

již trval přibližně čtvrt hodiny. Více jak tři hodiny by pak trval výpočet pro 12 měst. Je vidět enormní nárůst potřebného času, při minimálním zvýšení počtu měst.

Při řešení tohoto problému se s výhodou uplatňují evoluční algoritmy, jako například ACO (optimalizace mravenčí kolonií viz 1.3.1), které sice nezaručí nalezení nejlepšího možného řešení, ale jsou schopny nalézt takové řešení, jehož kvalita je přijatelná, ve velmi krátkém čase.

### 1.2.2 Problém batohu

Další známou optimalizační úlohou je tzv. Balení batohu. Mějme batoh o určité nosnosti a  $n$  předmětů o různých hmotnostech a hodnotách. Umístíme do batohu  $x$  předmětů tak, aby:

- a) Batoh nebyl přetížen.
- b) Hodnota věcí v batohu byla maximální možná.

S rostoucím počtem předmětů  $n$  se stává problém neřešitelný „hrubou silou“ podobně jako problém obchodního cestujícího (viz 1.2.1).

Pozn. V literatuře existuje několik verzí zadání problému balení batohu.

## 1.3 Principy vybraných evolučních optimalizačních algoritmů

Dále budou podrobněji popsány principy, několika významných zástupců evolučních optimalizačních technik. Každá z nich se používá pro řešení určitého okruhu problémů a nelze tvrdit, že některá je celkově lepší či horší, v souladu s principem No Free Lunch Teoremu (kapitola 1.1).

### 1.3.1 Ant Colony Optimization

Podobně jako některé další algoritmy, je optimalizace mravenčí kolonií inspirovaná chováním živočichů v přírodě. V tomto případě algoritmus napodobuje mravence, kteří v přírodě značkují své cesty feromonem. Jestliže cesta vede například k potravě, je označena a další mravenci chodí po této cestě a dále ji značkují, čímž zvyšují její přitažlivost pro další mravence. Jestliže některý mravenec najde kratší cestu k cíli, označí ji a všichni mravenci začnou využívat kratší cesty.

Takovýto způsob hledání nejkratší cesty, je možné s velkým úspěchem využít při řešení problémů typu obchodní cestující (viz 1.2.1). Zájem je též v oblasti transportních systémů (veřejná přeprava osob, zboží...) nebo síťového routování [4].



### 1.3.2 Diferenciální evoluce

Algoritmus diferenciální evoluce (DE) byl vyvinut v roce 1995 dvojicí autorů Ken Price a Rainer Storm. Je založený na klasickém pojetí genetických algoritmů, ale přináší několik změn.

DE používá operace křížení a mutace, typické pro genetické algoritmy, avšak odlišným způsobem. První významnou odlišností je potřeba čtyř jedinců ze staré populace, pro vytvoření jednoho jedince v populaci nové. Ke každému právě kříženému jedinci se náhodně vyberou tři další. Ze dvou těchto náhodných vektorů jedince a vektoru kříženého jedince vznikne sérií matematických operací, které plní funkci mutace, tzv. šumový vektor. Ten se následně kříží s posledním náhodným jedincem a vzniká nový jedinec. Podrobný popis principu DE lze nalézt např. v [4].

Diferenciální evoluce byla již vyzkoušena na mnoha optimalizačních problémech s velmi dobrými výsledky, mnohdy lepšími, než jaké poskytovaly jiné algoritmy [4].

### 1.3.3 SOMA

SOMA neboli SamoOrganizující se Migrační Algoritmus vznikl v roce 1999.

Podobně jako některé jiné algoritmy je inspirován chováním živočišného společenstva při plnění společného cíle v přírodě. Jedinci populace, kteří se pohybují v prostoru, spolu komunikují a vzájemně svůj pohyb ovlivňují s cílem nalézt nejlepší možné řešení problému.

Na rozdíl od jiných algoritmů však SOMA nepoužívá mutaci a křížení, alespoň ne klasickým způsobem. Nový jedinec nevzniká křížením jedinců ve staré populaci. Jedinci se pohybují směrem k nejlepšímu řešení a mění tak svoji pozici (parametry), velmi podobně jako je tomu u PSO algoritmu. Mutace je nahrazena perturbací, která ovlivňuje směr pohybu, ve smyslu odklonu od původně zvolené trasy.

Výraznou odlišností je však způsob pohybu a ohodnocování jedince. Zatímco například u PSO algoritmu jedinec v rámci jednoho migračního kola změní svoji pozici a je ohodnocen pouze jednou, v algoritmu SOMA každý jedinec v jednom migračním kole vykoná několik kroků vypočteným směrem a po každém kroku je ohodnocen. A z těchto mezivýsledků si pamatuje nejlepší hodnotu, se kterou postupuje do dalšího kola.

Z uvedeného vyplývá, že počet ohodnocení účelové funkce, je u SOMA algoritmu, při stejném počtu iterací (migračních kol), několikanásobně vyšší, než u některých jiných

algoritmů, např. PSO. Prostor je však mnohem důkladněji propátrán a SOMA dosahuje často významně lepších výsledků, než ostatní algoritmy. Detailní popis algoritmu SOMA obsahuje publikace autora tohoto algoritmu (viz [4]).

## 2 PSO

PSO (Particle swarm optimization) někdy též pouze Particle Swarm (česky rojení částic) je evoluční optimalizační algoritmus. Tato technika byla vyvinuta Rusellem Eberhartem a Jamesem Kennedym v roce 1995 a inspirované se sociálním chováním živočišných společenstev, například ptačích a rybích hejn [4].

Na rozdíl od genetických algoritmů nepoužívá PSO operace křížení a mutace. Populace jedinců (částic) se pohybuje v prohledávaném prostoru možných řešení dle jednoduchých pravidel. Každý jedinec má definovanou svoji rychlost, zná své nejlepší dosud nalezené řešení (značíme pBest) a také nejlepší dosud nalezené řešení v populaci (gBest). Tyto tři faktory spolurozhodují o tom, kterým směrem se částice (jedinec) vydá v dalším kroku. PSO může končit po určitém počtu iterací, nebo i při jiných, uživatelem nadefinovaných podmínkách.

Před podrobným popisem funkce PSO je nutno ještě zodpovědět otázku, jak funguje inteligence hejna a jaký je její význam. Viz 2.1.

### 2.1 Swarm intelligence – Intelligence hejna

Intelligence hejna je jedním z významných principů, dle kterého se řídí celá třída evolučních algoritmů. Základní myšlenkou je, že pokud umožníme jednoduchým jedincům tvořící populaci spolu komunikovat, byť velmi primitivně, můžeme získat překvapivě inteligentně se chovající celek.

Jedinci v takovém organizovaném hejnu, kde probíhá komunikace, a vzájemné ovlivňování jsou schopni nalézt lepší řešení problému, než by to dokázal kterýkoliv z nich sám. Chování každého jedince se však i v inteligentním hejnu stále bude řídit velmi jednoduchými pravidly. Princip vzájemného ovlivňování se jedinců si můžeme vysvětlit následovně:

Jestliže Váš soused má lepší řešení problému než Vy, snažíte se být více jako on. Je to velmi jednoduchý algoritmus s velkými důsledky[8]. Praktickou aplikaci tohoto principu v PSO algoritmu představuje globální nejlepší nalezené řešení (gBest), které je známo všem jedincům v populaci. Toto globální nejlepší řešení bere v potaz každý jedinec při výpočtu své rychlosti, tzn., že má vliv na jeho následující pozici.

## 2.2 Základní pojmy a princip PSO

Jak bylo uvedeno dříve v 2 a 2.1, PSO je evoluční algoritmus, který využívá inteligenci hejna k nalezení optimálního řešení. K tomuto účelu používá populaci jedinců (částic, ang. particles), kteří spolu v omezené míře komunikují. Každý jedinec je tedy bod v n-dimenzionálním prohledávaném prostoru, jehož souřadnice jsou hledanými parametry účelové funkce pro daný optimalizační problém.

Algoritmus pracuje v cyklech, tzv. migračních kolech či iteracích. V každém kole dojde pro každého jedince k výpočtu nové polohy, kontrola použitelnosti nové polohy (zda se nachází v zadaných mezích), ohodnocení jedince a porovnání s pBest a gBest.

Poloha bodu se mění přičtením vektoru rychlosti k jeho současné pozici. Vektor rychlosti se vypočte dle následujícího vztahu:

$$v(t+1) = v(t) + c_1 \cdot Rand \cdot (pBest - x(t)) + c_2 \cdot Rand \cdot (gBest - x(t)) \quad (1)$$

kde:

$v(t+1)$  – rychlost částice v dalším kroku

$v(t)$  – rychlost částice v aktuálním kroku

$c_1, c_2$  – prioritní nebo též učící faktory

pBest – pozice nejlepšího nalezeného řešení jedince

gBest – pozice nejlepšího nalezeného řešení v celé populaci

$x(t)$  – pozice jedince v aktuálním kroku

Rand – náhodné reálné číslo v intervalu  $<0,1>$

Novou pozici jedince získáme přičtením vektoru rychlosti k jeho současné pozici. Tedy:

$$x(t+1) = x(t) + v(t+1) \quad (2)$$

kde:

$x(t+1)$  – nová pozice jedince v dalším kroku

Ohodnocením jedince rozumíme dosazení parametrů jedince do účelové funkce (Cost function). Výsledná hodnota se porovná s nejlepší nalezenou hodnotou jedince pBest a nejlepší nalezenou hodnotou v populaci gBest. Jestliže se jedná o nové minimum, přepíše

se dosavadní nejlepší hodnota a pozice na aktuální parametry ohodnocovaného jedince. Na rozdíl od řady jiných algoritmů však i jedinec, který je horší než dosavadní nejlepší řešení postupuje do další generace (migračního kola, iterace). Nedochází tedy k zániku jedinců a vzniku nových v pravém slova smyslu, ale pouze k jejich pohybu (migraci). Problémem je však velká akcelerace částic, která vyplývá ze způsobu výpočtu vektoru rychlosti (viz rovnice 1) a překračování vymezeného prostoru. Z tohoto důvodu se při každém výpočtu nové polohy musí provést kontrola, zda je tato v zadaných mezích a pokud ne, opravit ji. Zpravidla se provede omezení seshora, nebo vygenerování nové náhodné polohy.

V extrémním případě může díky tomuto jevu celý algoritmus degradovat na náhodné hledání. Je vhodné tomu předcházet a zabránit extrémní akceleraci částic. Způsoby jak omezit rychlost částic se zabývá řada modifikací PSO algoritmu (viz kapitola 2.4).

Algoritmus se zpravidla ukončí po dosažení stanoveného počtu iterací (migračních kol) a požadovanou výslednou hodnotou je hodnota zapsaná v proměnné gBest.

## 2.3 Parametry PSO

Pro některé parametry uvádí různé zdroje odlišné značení, pro přehlednost a zachování konzistence bude uvedeno značení, tak jak je použito dále v praktické části práce.

### 2.3.1 Dimenze

Značeno dim. Dimenzí rozumíme počet argumentů účelové funkce, tedy rozměr jedince v populaci. Znatelně ovlivňuje náročnost a přesnost algoritmu. Vyšší dimenze, znamená vyšší složitost problému. S rostoucí dimenzí se zvětšuje nejen rozměr jedince, ale i vektoru rychlosti, pBest apod.

### 2.3.2 Rozsah

Prostor přijímaných řešení. Vymezuje oblast, ve které se mohou částice pohybovat, resp. hodnoty, jakých mohou nabývat jednotlivé argumenty účelové funkce. Pro každou dimenzi může být zadán jiný rozsah.

### 2.3.3 Specimen

Pro zadání dimenze a rozsahu se často s výhodou využívá vzorový jedinec, tzv. specimen. Jeho konstrukce se může drobně lišit. Může obsahovat mimo rozsahu pro jednotlivé dimenze i číselný typ, se kterým má algoritmus pracovat.

### 2.3.4 Počet částic

Tento parametr přímo ovlivňuje počet ohodnocení účelové funkce v každém migračním kole. Doporučeně (dle [4]) volíme počet jedinců  $NP = 10 \cdot \text{dim}$ . Pro dvouargumentovou funkci bude tedy prostor prohledávat 20 jedinců a v každém migračním kole dojde k 20-ti ohodnocením účelové funkce. Velký počet jedinců zvyšuje šanci na nalezení kvalitního řešení, ale znamená též velký nárůst výpočetní náročnosti.

### 2.3.5 Počet iterací

Iterace neboli migrační kola zjednodušeně označují jeden průběh algoritmu. Tedy ohodnocení jedinců a úprava jejich pozic. Dosažení zadaného počtu iterací je typickou ukončující podmínkou PSO algoritmu. Příliš malý počet iterací může způsobit, že algoritmus nestihne konvergovat do místa lokálního extrému. Příliš velký pak zvýšenou náročnost, bez vlivu na přesnost výsledku. Proto může být doplněna další ukončující podmínka, např. změna nejlepšího nalezeného řešení v procentech apod.

### 2.3.6 Učící faktory

Učící (někdy též prioritní) faktory  $c_1$  a  $c_2$  ovlivňují význam složky pBest a gBest při výpočtu nového vektoru rychlosti (viz rovnice 1). Rozdílná hodnota  $c_1$  a  $c_2$  značí, že jedné ze složek přikládáme větší vliv a to té, u které je parametr  $c$  větší. Díky součinu s náhodným číslem ale může být priorita i obrácena.

Obvykle  $c_1$  a  $c_2$  nabývají hodnoty 2. Přesto se při různých problémech používají i jiná nastavení. Nejčastěji v rozsahu  $\{0, 4\}$  [4].

## 2.4 Modifikace PSO algoritmu

Stejně jako u jiných evolučních algoritmů, i u PSO existuje velká řada modifikací, které se snaží odstranit některé jeho nedostatky, nebo upravit chování tak, aby pro určitou skupinu problémů dosahoval lepších výsledků. Jelikož známých modifikací je velké množství, budou dále popsány pouze ty, které jsou použity v praktické části práce.

### 2.4.1 Omezení maximální rychlosti $V_{\max}$

Velkým problémem PSO v základní podobě je prudká akcelerace částic, které mají tendenci překračovat hranice prohledávaného prostoru a jelikož takové částici je



vygenerována nová pozice, algoritmus může snadno degradovat z evolučního chování na náhodné prohledávání prostoru.

Proto byla zavedena hodnota  $V_{\max}$ , která omezuje velikost jednotlivých složek ve vektoru rychlosti.  $V_{\max}$  tedy může mít pro každou dimenzi jinou hodnotu. Dle [4] je doporučená hodnota  $V_{\max}$  stanovena na 1/20 rozsahu.

Složky vygenerovaného vektoru rychlosti jsou tedy porovnány s touto maximální přípustnou hodnotou a je-li tato překročena, nastaví se daná hodnota ve vektoru rychlosti na hodnotu  $V_{\max}$ .

### 2.4.2 Setrvačnost $w$

Tato modifikace zavádí proměnnou hodnotu  $w$ , která násobí současnou rychlost jedince, při výpočtu rychlosti nové. Je několik možností, jak se setrvačností pracovat. Typicky volíme počáteční a koncovou hodnotu  $w_{\text{start}}$  a  $w_{\text{end}}$  které slouží pro výpočet hodnoty setrvačnosti v dané iteraci, dle vzorce:

$$w = w_{\text{start}} - \frac{(w_{\text{start}} - w_{\text{end}}) \cdot i}{n} \quad (3)$$

kde:

$w$  – hodnota setrvačnosti

$w_{\text{start}}$  – počáteční hodnota setrvačnosti

$w_{\text{end}}$  – koncová hodnota setrvačnosti

$i$  – aktuální migrační kolo

$n$  – celkový počet migračních kol

Díky zavedení setrvačnosti algoritmus v počáteční fázi prohledává prostor ve větších skocích a tím je posíleno hledání globálního extrému. V závěrečné fázi nízká setrvačnost více omezuje rozsah pohybu částic a povzbuzuje tak schopnost podrobnějšího průzkumu oblasti globálního extrému a nalezení přesnější hodnoty.

### 2.4.3 Constriction faktor

Modifikace spočívá v zavedení parametru Constriction faktor (značíme  $\chi$ ) o konstantní hodnotě, doporučeno 0.729 (dle [4]), jehož použití je velmi podobné setrvačnosti. Rozdílně

se však nenásobí pouze předchozí rychlost jedince, ale celá nová vypočtená rychlost. Tedy zjednodušeně:

$$v(t + 1) = \chi \cdot v(t + 1) \quad (4)$$

Rozepsáno:

$$v(t + 1) = \chi \cdot \{v(t) + c_1 \cdot Rand \cdot (pBest - x(t)) + c_2 \cdot Rand \cdot (gBest - x(t))\} \quad (5)$$

Hodnota parametru se v čase nemění a má na rychlost omezující vliv. Změnou jeho hodnoty lze upravit chování algoritmu pro potřeby řešení konkrétního problému.

### 3 TESTOVACÍ FUNKCE

Jsou to takové funkce, u kterých známe jejich průběh a můžeme je využít k ohodnocení funkčnosti a efektivnosti evolučního algoritmu. Dělíme je na funkce unimodální a multimodální.

Testovacích funkcí existuje velké množství, v dalších částech této kapitoly jsou uvedeny pouze ty, které byly použity pro testování PSO algoritmu v praktické části práce.

#### 3.1 Unimodální funkce

Tyto funkce mají na daném intervalu pouze jeden extrém. Z hlediska optimalizace je nalezení řešení problémů popsaných unimodální funkcí jednodušší.

##### 3.1.1 1. De Jongova funkce

Funkce druhé mocniny. Popsána rovnicí (6).

**Matematický zápis**

$$f(x) = \sum_{i=1}^{dim} x_i^2 \quad (6)$$

**Minimum**

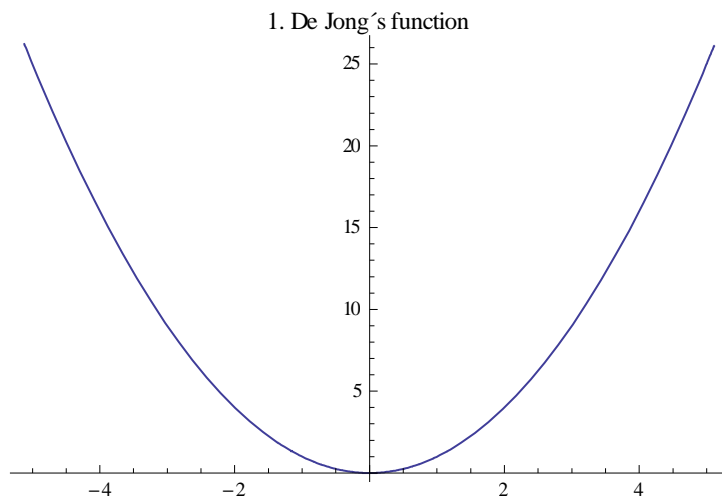
Minimum funkce leží na průsečíku nulové souřadnice všech dimenzí. Hodnota funkce v takovém bodě je nula.

Pozice pro  $E_n$ :  $(x_1, x_2, \dots, x_n) = (0, 0, \dots, 0)$

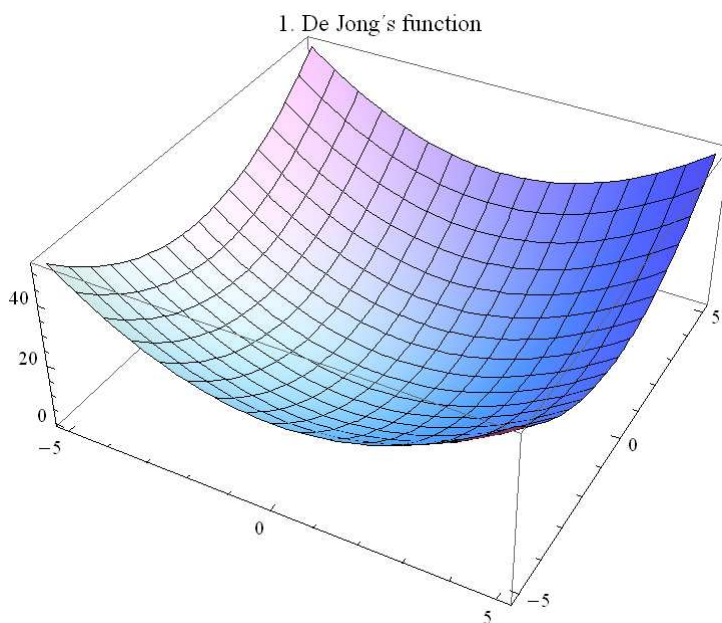
Hodnota pro  $E_n$ :  $y = 0$

**Grafické zobrazení**

Na obrázcích 2 a 3 je zachycen průběh první De Jongovy funkce ve 2D a 3D. Tedy pro jedno a dvouargumentovou účelovou funkci.



Obr. 2 První De Jongova funkce ve 2D



Obr. 3 První De Jongova funkce ve 3D

### 3.1.2 3. De Jongova funkce

Funkce absolutní hodnoty má lineární průběh do bodu extrému (minima). Viz rovnice (7).

**Matematický zápis**

$$f(x) = \sum_{i=1}^{dim} |x_i| \quad (7)$$

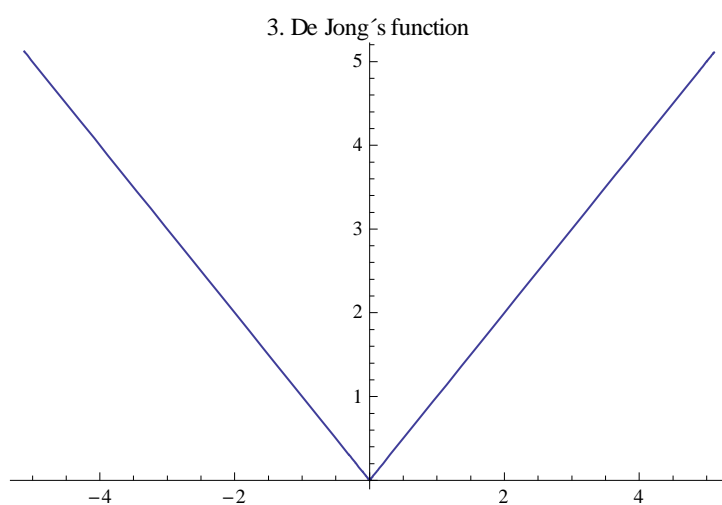
**Minimum**

Pozice pro  $E_n$ :  $(x_1, x_2, \dots, x_n) = (0, 0, \dots, 0)$

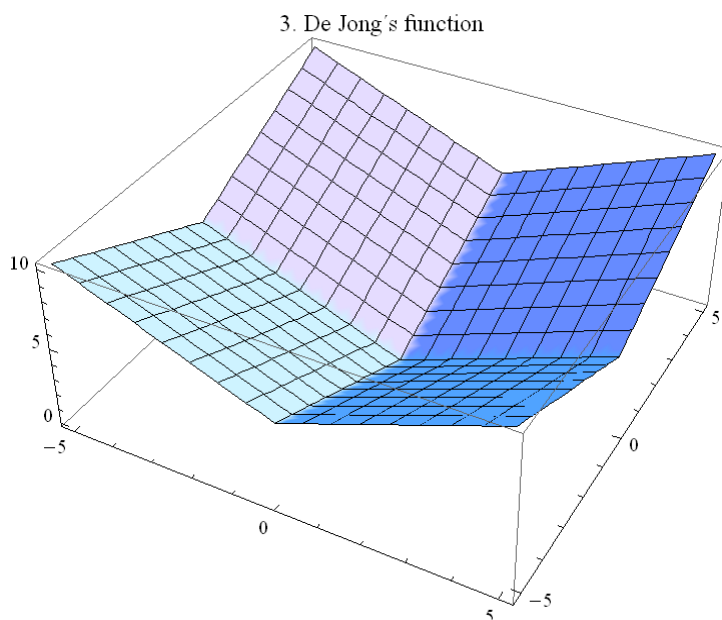
Hodnota pro  $E_n$ :  $y = 0$

**Grafické zobrazení**

Průběh třetí De Jongovy funkce je zobrazen na obrázcích 4 a 5.



Obr. 4 Třetí De Jongova funkce ve 2D



Obr. 5 Třetí De Jongova funkce ve 3D

### 3.1.3 4. De Jongova funkce

Definována vztahem (8). Na zvoleném intervalu  $\langle -5.12, 5.12 \rangle$  funkce nejdříve velmi strmě klesá, pak přechází do velmi pozvolného průběhu k bodu minima.

#### Matematický zápis

$$f(x) = \sum_{i=1}^{dim} ix_i^4 \quad (8)$$

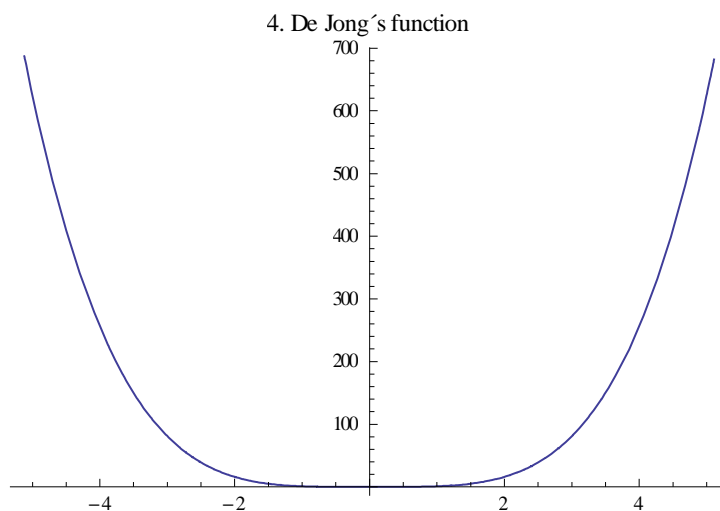
#### Minimum

Pozice pro  $E_n$ :  $(x_1, x_2, \dots, x_n) = (0, 0, \dots, 0)$

Hodnota pro  $E_n$ :  $y = 0$

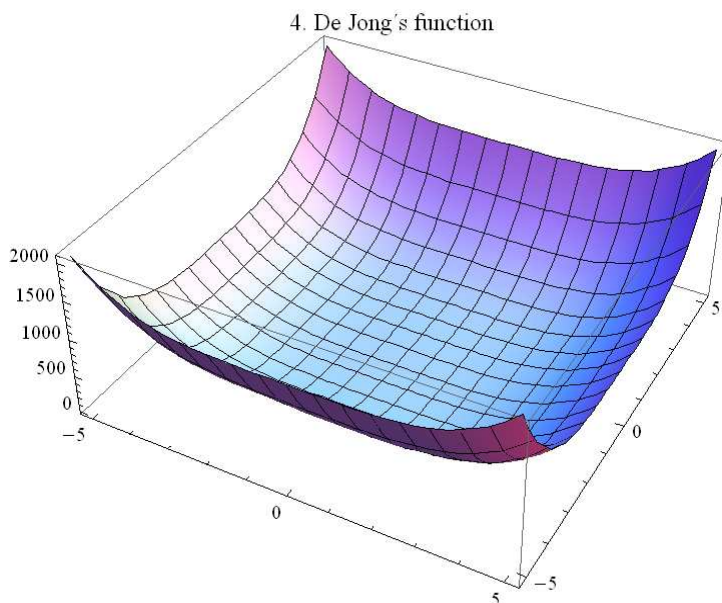
#### Grafické zobrazení

Obrázky 6 a 7 zachycují průběh čtvrté De Jongovi funkce na intervalu  $\langle -5.12, 5.12 \rangle$



Obr. 6 Čtvrtá De Jongova funkce ve 2D





Obr. 7 Čtvrtá De Jongova funkce ve 3D

## 3.2 Multimodální funkce

Multimodální funkce může mít více než jeden extrém různé, nebo i stejné hodnoty. Složité multimodální funkce, s větším počtem extrémů, představují výzvu pro optimalizační algoritmy, především z hlediska překonání předčasné konvergence do lokálního extrému, který však není extrémem globálním.

### 3.2.1 2. De Jongova funkce

Též známá jako Rosenbrokovo sedlo. Používá pro výpočet znalost následující hodnoty a sumu pro  $(dim-1)$  hodnot, proto je pro jednorozměrnou funkci vždy rovna 0. Viz rovnice (9).

#### Matematický zápis

$$f(x) = \sum_{i=1}^{dim-1} 100 (x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \quad (9)$$

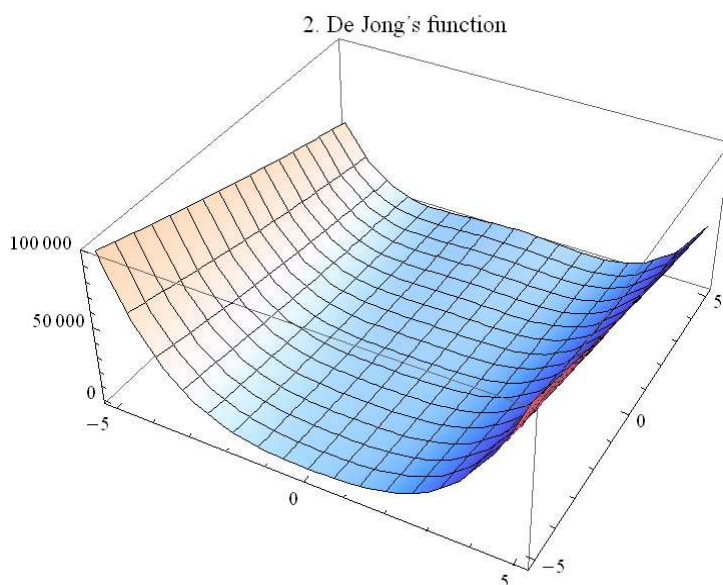
#### Minimum

Pozice pro  $E_n$ :  $(x_1, x_2, \dots, x_n) = (1, 1, \dots, 1)$

Hodnota pro  $E_n$ :  $y = 0$

### Grafické zobrazení

Obrázek 8 zachycuje druhou De Jongovu funkci ve 3D.



Obr. 8 Druhá De Jongova funkce ve 3D

### 3.2.2 Schwefelova funkce

Složitá multimodální funkce. Z grafů v 3.2.2.3 je zjevná členitost funkce, která je pro gradientní metody prakticky neřešitelná. Algoritmus typu PSO by však měl globální extrém této funkce nalézt s vysokou pravděpodobností.

#### Matematický zápis

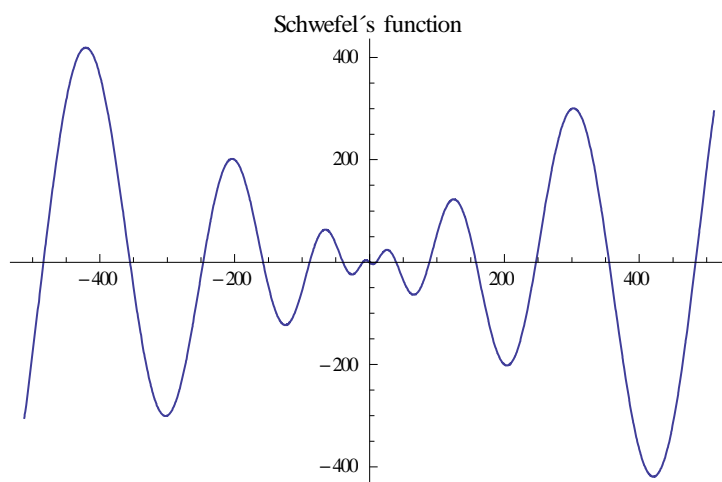
$$f(x) = \sum_{i=1}^{dim} -x_i \sin(\sqrt{|x_i|}) \quad (10)$$

#### Minimum

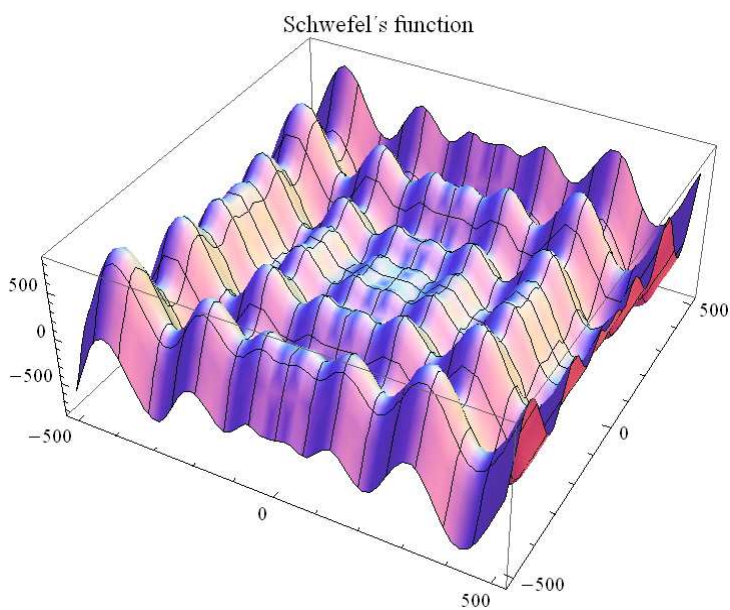
Pozice pro  $E_n$ :  $(x_1, x_2, \dots, x_n) = (420.969, 420.969, \dots, 420.969)$

Hodnota pro  $E_n$ :  $y = -418.983 * dim$

## Grafické zobrazení



Obr. 9 Schwefelova funkce ve 2D



Obr. 10 Schwefelova funkce ve 3D

### 3.2.3 Rastriginova funkce

Velmi členitá funkce, s vysokou hustotou ostrých extrémů. Velké riziko uváznutí algoritmu ve „špatném“ extrému. Definována vztahem (11).

**Matematický zápis**

$$f(x) = 10dim \sum_{i=1}^{dim} x_i^2 - 10\cos(2\pi x_i) \quad (11)$$

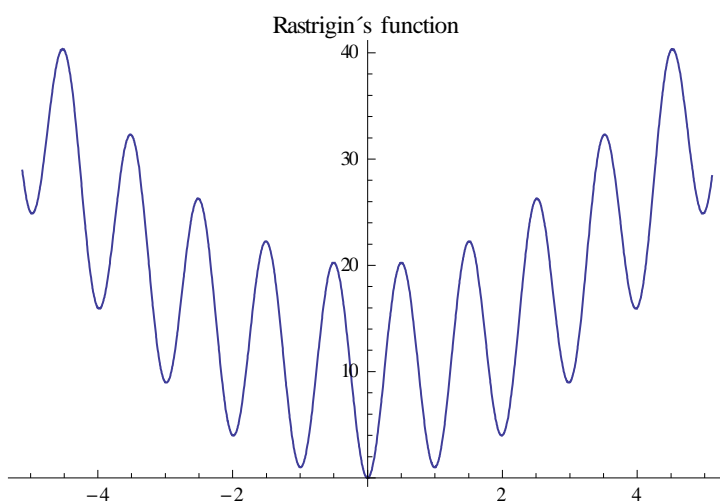
**Minimum**

Pozice pro  $E_n$ :  $(x_1, x_2, \dots, x_n) = (0, 0, \dots, 0)$

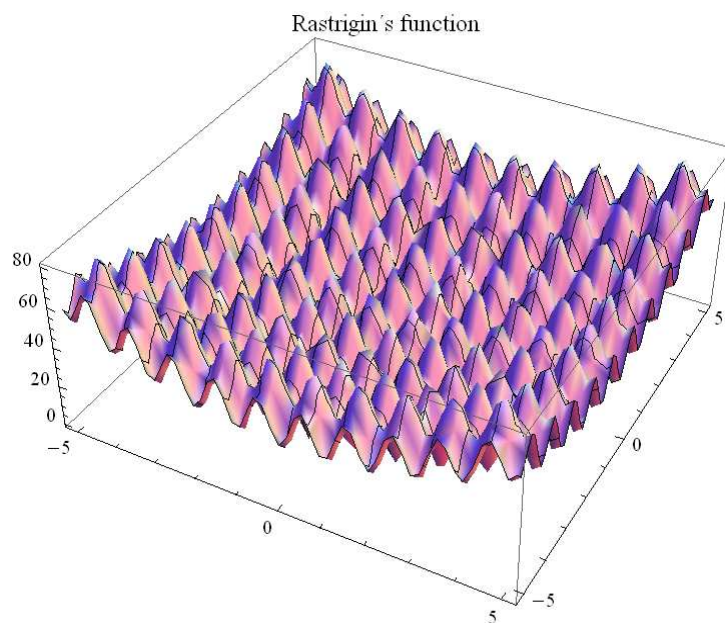
Hodnota pro  $E_n$ :  $y = 0$

**Grafické zobrazení**

Vysoká hustota extrémů je dobře patrna ze zobrazení Rastriginovy funkce ve 2D a 3D na obrázcích 11 a 12.



Obr. 11 Rastriginova funkce ve 2D



Obr. 12 Rastriginova funkce ve 3D

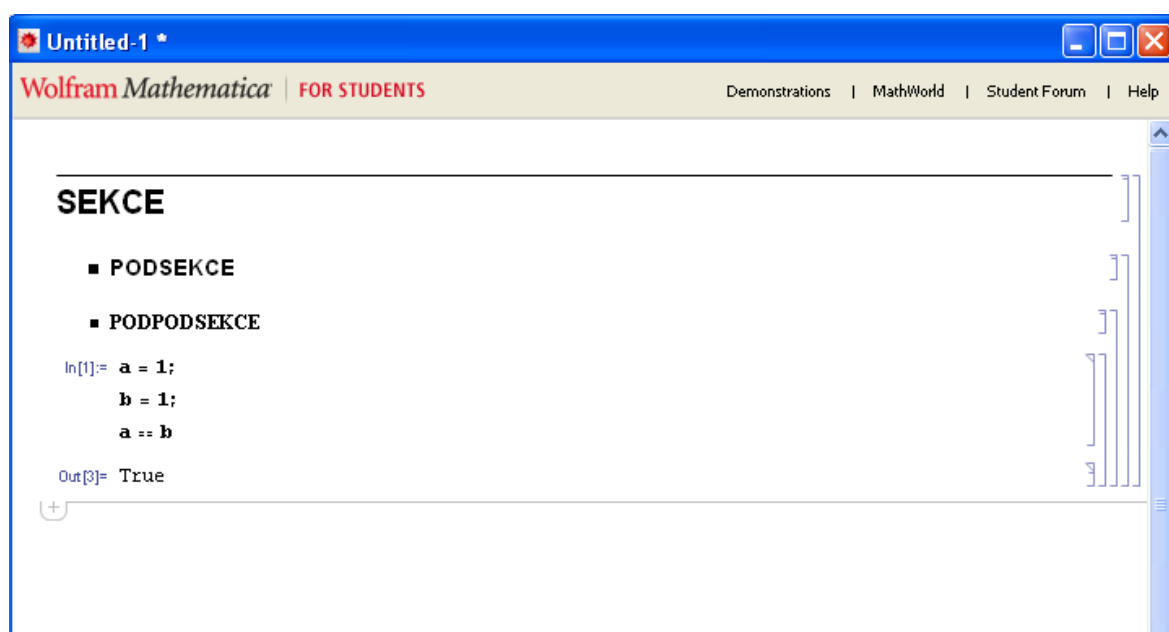
## 4 WOLFRAM MATHEMATICA

Mathematica je světově nejuznávanější aplikaci pro výpočty [9]

Světové renomé získala díky vysoké výkonnosti nejen při numerických operacích, kvalitním možnostem vizualizace výsledků a uživatelskému prostředí na vysoké úrovni. Mathematica je vytvářena přímo pro potřeby vysoce náročných výpočtů a této potřebě odpovídá i velká řada implementovaných funkcí, které umožňují tyto výpočty provádět efektivněji než při použití jiných aplikací či jazyků.

### 4.1 Vývojové prostředí

Základním prvek prostředí je tzv. notebook, který se dále dělí na buňky a sekce. Buňka je nejmenší spustitelnou částí kódu, která může obsahovat jednu i více funkcí, či výrazů. Tyto budou vykonány v jednom cyklu neoddělitelně. Sekce slouží především k zpřehlednění notebooku. Existuje jich několik druhů, liší se svým formátem a především definovanou úrovní. Sekce lze tedy vnořovat. Mimo estetické a organizační plní sekce i další funkci. Notebook lze totiž spouštět celý, jednotlivé buňky anebo právě sekce. Rozdělují tak kód na samostatně spustitelné celky v rámci jednoho notebooku. Buňky jsou označeny In a Out. In buňky jsou vstupní, vyplněné uživatelem. Out buňky označují vypočtený výstup předchozí In buňky. Popsaná struktura je zachycena na Obr. 13.



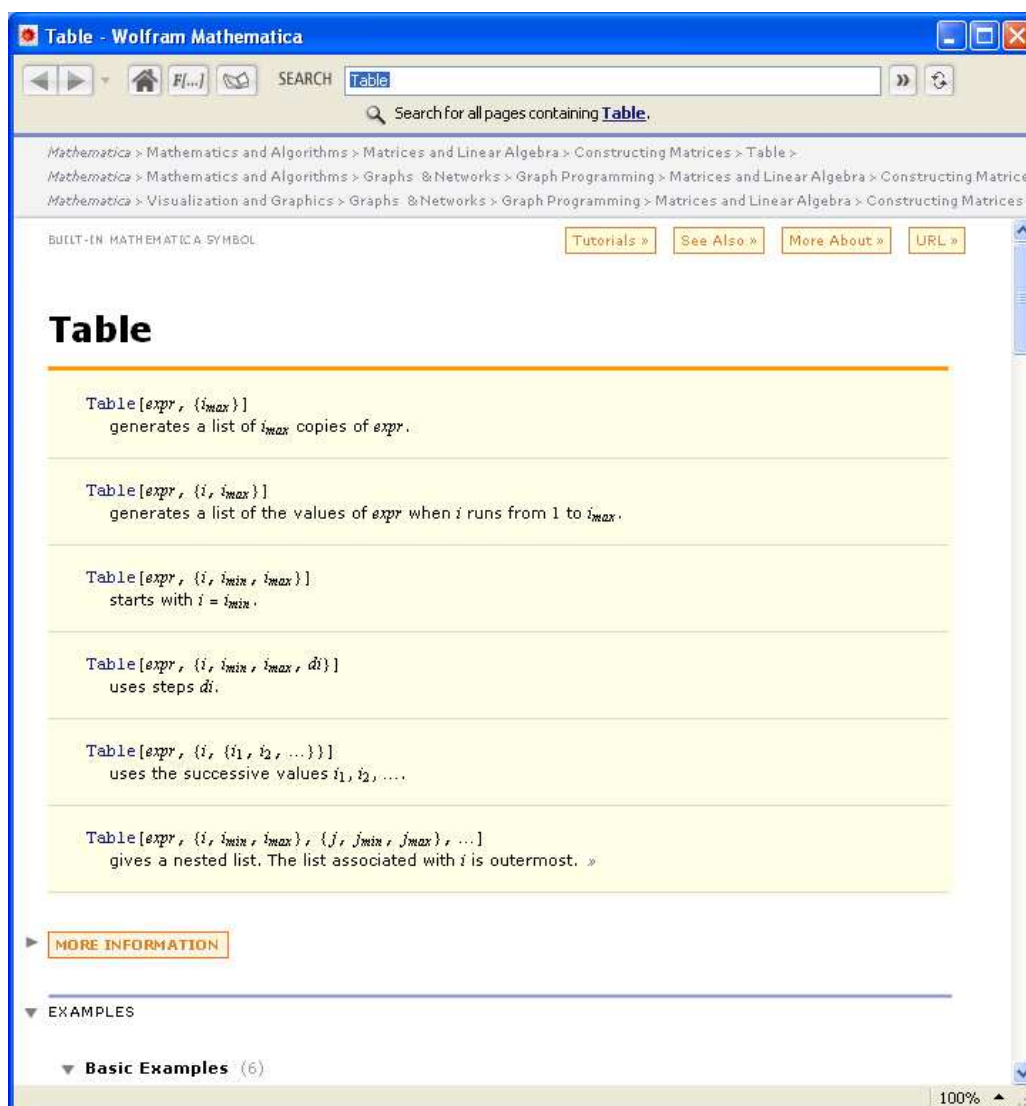
Obr. 13 Notebook v prostředí Mathematica 8 se zobrazením buněk a sekcí



### 4.1.1 Nápopvěda

Jednou z uživatelsky velmi oceňovaných funkcí Mathematicy je přehledně řešená nápověda. Ke každé funkci obsahuje jednoduché i složité příklady použití, seznam možných vstupů a jejich typů. Velmi užitečná je též nabídka příbuzných funkcí. Pokud tedy uživatel nezná přesně název funkce, ale ví přibližně, čeho se týká a co chce dosáhnout, snadno potřebnou funkci (je-li taková k dispozici) v nápovědě nalezne i s podrobným návodem, jak ji aplikovat.

Řada funkcí v Mathematice může být volána s různým počtem a typem vstupních parametrů, které ovlivňují výsledné chování funkce. Všechny varianty vstupních parametrů jsou v nápovědě popsány i s vysvětlením výsledného chování funkce.



Obr. 14 Nápopvěda pro funkci Table v Mathematice v. 8

Příkladem může být funkce Table, která vykonává konečný cyklus (viz Obr. 14). Prvním vstupním parametrem je vždy vykonávaný kód. Druhý parametr může být zadán několika způsoby a stanoví, jak bude cyklus vykonáván, případně jakou hodnotu bude mít iterátor.

Je-li na 2. Pozici zadáno číslo  $n$ , provede se kód  $n$ -krát. V případě zadání pole ve formátu  $\{i, \{n\}\}$  bude kód opět vykonán  $n$ -krát, ale iterátor  $i$  bude nabývat hodnot od 1 do  $n$ .

## 4.2 Historie vývoje

První verze (1.0) aplikace Mathematica pro Macintosh byla vydána v roce 1998. Verze 2.0 následovala v roce 1991 a přinesla, mimo jiné, numerického řešitele pro obyčejné diferenciální rovnice. Tato verze se dočkala v letech 1992 a 93 dvou updatů (2.1 a 2.2), které řešili především rozšíření na další operační systémy. Konkrétně Windows 3.1 a Linux.

O tři roky později, tedy v roce 1996, vychází Mathematica 3 a přináší podporu algebraických čísel a interaktivní systém pro zadávání matematických výrazů. Výrazné zrychlení výpočtů přinesla verze 4 v roce 1999 a dalšího, rekordy lámajícího zrychlení, se dočkali uživatelé v roce 2003 s verzí Mathematica 5. Vysoké rychlosti výpočtů bylo dosaženo optimalizací lineární numerické algebry pro procesor.

Dynamická interaktivita, která umožňuje vytváření sofistikovaných interaktivních rozhraní z jediné vstupní řádky.[10] Jedna z novinek, kterou přinesla verze 6 v roce 2007. Tato verze se v letech 2007 a 2008 dočkala tří malých updatů. Na prudký rozvoj výpočetní techniky v oblasti paralelizace reflektovala Mathematica ve verzi 7 (rok 2008) implementací vysoce výkonných paralelních výpočtů.

Velmi diskutovaným nedostatkem verze 7 byla chybějící podpora CUDA jader, která by umožnila jejich využití pro paralelní výpočty. Toho se, mimo jiná vylepšení, uživatelé dočkali v roce 2010 s příchodem verze 8, která je zatím poslední.

## **II. PRAKTICKÁ ČÁST**

## 5 IMPLEMENTACE PSO ALGORITMU

PSO algoritmus byl implementován ve vývojovém prostředí Wolfram Mathematica 8, které je s výhodou používáno právě při řešení složitých výpočetních úloh. Bylo vytvořeno několik verzí algoritmu, jejichž principy a odlišnosti budou popsány dále v této kapitole.

Každá vytvořená verze je samostatně použitelná ve formě notebooku a dále vložena do vytvořené knihovny (package, m-file).

### 5.1 Přehled verzí

Kromě základní verze PSO, byly vytvořeny tři modifikace. PSO se setrvačností, s constriction faktorem a s omezením maximální rychlosti. Tyto jsou označeny (weight, const, Vmax v uvedeném pořadí). Jednotlivé modifikace jsou blíže popsány v kapitole 2.4.

Základní verze i modifikace byli implementovány dvakrát. Jednou s označením BASE a podruhé s označením ADVANCED. Verze značené ADVANCED vyžadují pokročilé zadávání vstupních parametrů a jsou určeny pro uživatele dobře obeznámené s principem a nastavením PSO algoritmu. Funkce značené BASE jsou pak určeny pro uživatele se základní znalostí a vyžadují minimální množství zadaných hodnot a pro stanovení parametrů nutných pro běh algoritmu používají výpočty či dosazení doporučených hodnot z literatury.

#### 5.1.1 Seznam vytvořených verzí:

**PSO\_BASE** – Základní verze algoritmu

**PSO\_BASE\_CONS** – Modifikace PSO s využitím Constriction faktoru

**PSO\_BASE\_WEIGHT** – Modifikace PSO se setrvačností

**PSO\_BASE\_VMAX** – Modifikace PSO s omezením maximální rychlosti.

**PSO\_ADV** – Základní verze algoritmu s pokročilým zadáváním vstupů

**PSO\_ADV\_CONS** – Modifikace PSO s využitím Constriction faktoru s pokročilým zadáváním vstupů

**PSO\_ADV\_WEIGHT** – Modifikace PSO se setrvačností s pokročilým zadáváním vstupů

**PSO\_ADV\_VMAX** – Modifikace PSO s omezením maximální rychlosti s pokročilým zadáváním vstupů

## 5.2 Základní struktura

Základní strukturu naprogramované funkce tvoří:

- a) Definice vstupů
- b) Definice a inicializace lokálních proměnných
- c) Výpočetní algoritmus
- d) Definice výstupu (návrátové hodnoty)

### 5.2.1 Vstupy

Vstupy se definují pomocí notace `nazev_funkce[vstup1_,vstup2_, ..., vstup_n]`.

Základná dva vstupy, společné pro všechny vytvořené verze jsou:

`Cost_` - Cost function, účelová funkce. Zadává se název předdefinované funkce.

`Spec_` - Specimen, vzorový jedinec. Podle specimen je určen počet dimenzí a meze pro každou z nich.

Specifické vstupy pro jednotlivé modifikace:

`Cs_` - Constiction faktor

`Wx_` – Dělitel pro výpočet maximální rychlosti u `PSO_BASE_VMAX`. Vektor maximálních rychlostí u `PSO_ADV_VMAX`

`Ws_` - Počáteční setrvačnost

`We` – Koncová setrvačnost

Vstupy pro `ADVANCED` verze:

`it_` - Počet iterací, po kterých je algoritmus ukončen

`Np_` - Počet částic

`c` – Učící faktor  $c_1$

`cc` – Učící faktor  $c_2$

Některé vstupy, především u `BASE` verzí, nemusí být uživatelem zadány a pro takový případ je definována implicitní hodnota.

### 5.2.2 Lokální proměnné

Definování lokálních proměnných je velmi výhodné z důvodu předcházení kolizí i uvolnění paměti. K definování lokálních proměnných slouží v Mathematice funkce `Module`. Tato zároveň zašituje zbytek algoritmu a určuje tak hranice definované funkce.

Názvy proměnných byly voleny s ohledem na srozumitelnost, a proto zde není uveden seznam všech proměnných, neboť úloha každé z proměnných vyplývá z jejího názvu. Jednopísmenné názvy jsou použity pro iterátory.

V prostředí Mathematica je proměnná definována pouze svým názvem. Neexistuje zde definice datového typu. Každá proměnná může nabývat libovolné hodnoty ať už číselné či jiné (např. znak, pole, vektor, text, obrázek atd.).

### 5.2.3 Algoritmus

Samotný PSO algoritmus (bez vytváření populace atp.) lze zjednodušeně zapsat dvěma cykly. Vnitřní provádí výpočet rychlosti jedince, kontrolu dosažení nové nejlepší hodnoty a posun v rámci populace, vnější pak řídí posun iterací (migračních kol).

Pro realizaci cyklu slouží v Mathematice funkce `Table`. Přestože jsou definovány i další podobné funkce (např. `For`), dosahuje `Table` nejlepších výsledků z pohledu času i výpočetní náročnosti. Podle testů je v některých případech `Table` vykonávající stejný kód jako `For` rychlejší více než 20x.

Ve výsledných algoritmech se `Table` vyskytuje více než 2x, protože provádí další dodatečné funkce. Například generování populace, nebo kontrolu, zda se částice nachází v povolených hranicích prohledávaného prostoru. Tato kontrola je implementována ve všech verzích.

Struktura implementace jádra PSO je popsána na příkladu verze `PSO_BASE`. Jednotlivé cykly jsou v kódu barevně označeny pro přehlednost. První (modrý) `Table` je vnějším cyklem PSO, který iteruje migrační kola. Druhý (značen fialovou barvou) představuje vnitřní cyklus PSO, ve kterém postupně dochází k výpočtu nové rychlosti jedince, určení nové polohy jedince, kontroly korektnosti polohy v rámci prostoru, ohodnocení jedince, srovnání s osobním a globálním nejlepším výsledkem.

Kontrola, zda nová poloha jedince je v souladu s omezením pro jednotlivé dimenze, je prováděna třetím cyklem (červený `Table`). Ten prochází vektor s polohou jedince a vektory dolních a horních mezí a srovnává jednotlivé položky na stejných pozicích. V případě, že

částice v dané dimenzi překročila vymezený prostor, je jí vygenerována nová pozice, pouze pro danou dimenzi. Nedochází tak k úplné ztrátě informace a evolučního chování. Popsaná struktura je společná pro všechny vytvořené verze.

#### 5.2.4 Výstup

Návratová hodnota je pro všechny verze definována stejně a to jako vektor o dvou prvcích. Prvním je hodnota gBest, tedy nejlepší (v případě hledání minima nejmenší) nalezená hodnota. Druhým prvkem je vektor s pozicí gBest, tedy parametrů účelové funkce, pro které tato nabývá hodnoty gBest.

### 5.3 Specifika

Kromě základní struktury popsané v kapitole 5.2.3 obsahuje kód jednotlivých modifikací specifické příkazy, které ovlivňují chování algoritmu.

#### 5.3.1 PSO\_BASE\_VMAX

Tato verze je specifická zavedením dělicího faktoru pro výpočet maximální rychlosti. Na rozdíl od ADVANCED verze, kde je uživatel nucen zadat přímo hodnoty povolené maximální rychlosti pro každou dimenzi.

S použitím dělicího faktoru se maximální rychlost vypočítá podělením rozsahu (získán součtem absolutních hodnot mezí zadaných ve specimen) hodnotou dělicího faktoru pro každou dimenzi. Implicitně je omezení nastaveno na  $1/20$  rozsahu. Tato hodnota je doporučena z literatury. (viz kapitola 2.4.1).

Poměr maximální rychlosti k velikosti prostoru je tedy stejný ve všech dimenzích, i když konkrétní hodnoty se liší.

Do jádra (kapitola 5.2.3) je v této modifikaci přidán cyklus Table, který kontroluje překročení maximální rychlosti, hned po jejím vygenerování, podobně jako je kontrolována korektnost vypočtené polohy.

#### 5.3.2 PSO\_BASE\_CONS

Zavedena proměnná pro hodnotu constriction faktoru, kterou je násobena vypočtená rychlost jedince. Jedná se o jedinou změnu oproti základní verzi, ale při správné volbě hodnoty constriction faktoru vede ke zpřesnění nalézáných řešení. Stejná úprava je i ve verzi PSO\_ADV\_CONS.

### 5.3.3 PSO\_BASE\_WEIGHT

V kódu této modifikace se objevují tři nově proměnné, které uchovávají hodnotu počáteční, koncové a aktuální setrvačnosti. Mezi vnější a vnitřní cyklus PSO algoritmu je vložen výpočet aktuální setrvačnosti pro dané migrační kolo. Hodnota setrvačnosti ovlivňuje, zda je algoritmus povzbuzován k prohledávání prostoru, nebo zpřesňování nalezeného řešení. Verze PSO\_ADV\_WEIGHT je upravena totožně.

### 5.3.4 PSO\_ADV\_VMAX

Na rozdíl od verze BASE vyžaduje specifické zadání maximální rychlosti pro každou dimenzi, v podobě vektoru. Místo výpočtu je tedy do proměnné Vmax pouze přiřazen vektor zadaný na vstupu. Tato verze tedy umožňuje zadat libovolné omezení rychlosti ke každé dimenzi, bez ohledu na jejich vzájemný poměr. Klade větší nároky na uživatele.



## 6 TESTOVÁNÍ

V kapitole 3 byly představeny testovací funkce. Tyto byly použity pro ověření chování vytvořené implementace PSO algoritmu a jeho vybraných modifikací. Protože vliv náhody je nezanedbatelný, nelze vyvodit závěry pouze z jedné realizace, tedy aplikace algoritmu na testovací funkci. Proto byla vždy provedena řada realizací a pro další vyhodnocení byly uchovány tři hodnoty. Nejhorší nalezené řešení na konci běhu algoritmu (značeno „max“), nejlepší nalezené řešení (značeno „min“) a medián konečných hodnot v daném balíku realizací (značeno „median“). Tučně jsou v tabulce označeny nejlepší hodnoty pro daný balík realizací.

### 6.1 Testování BASE verzí algoritmu

Všechny použité verze, byly volány pouze s parametry CF a Specimen. Specifické parametry pro verze *PSO\_BASE\_CONS*, *PSO\_BASE\_WEIGHT* a *PSO\_BASE\_Vmax* tedy nabývají implicitně zadaných hodnot a stejně tak ostatní (společné) parametry nastavení.

Počty prováděných realizací byli standardně 20 a 50, v případě dvou-dimenzionálního problému pak i 100 a 200. Byl zkoumán i vliv počtu dimenzí na schopnost algoritmu přiblížit se k hodnotě globálního minima a proto jsou testy prováděny pro hodnoty  $dim = 1, 2$  a  $5$ .

Použity byly následující funkce: *PSO\_BASE*, *PSO\_BASE\_CONS*, *PSO\_BASE\_WEIGHT* a *PSO\_BASE\_Vmax*. Cílem bylo ověřit funkcionalitu a přesnost dosahovaných výsledků pro uživatele neznalého nastavení algoritmu PSO, pro kterého jsou právě „BASE“ verze určeny.

Implicitní hodnoty při testu:

$$n = 50$$

$$NP = 10 \cdot dim$$

$$c_1 = c_2 = 2$$

Výchozí hodnota umístění gBest je v  $\frac{1}{2}$  horní meze. Obdobně pBest se nachází v  $\frac{1}{2}$  dolní meze.

V modifikaci *PSO\_BASE\_CONS* je hodnota constriction factoru implicitně nastavena na 0.729. Hodnoty počáteční a konečné setrvačnosti v modifikaci *PSO\_BASE\_WEIGHT* jsou nastaveny následovně:  $w_{start} = 0.9$   $w_{end} = 0.4$ . Maximální rychlost částice pro modifikaci

$PSO\_BASE\_Wmax$  činí  $1/20$  zadaného rozsahu pomocí specimenu a konkrétní hodnota se tedy může pro každou dimenzi lišit.

### 6.1.1 1. De Jongova funkce

Hodnota minima pro  $E_n$ :  $y = 0$

#### 6.1.1.1 Naměřená data

Pro  $dim = 1$

Tab. 1 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a  $dim=1$

20 realizací	max	min	Median
$PSO\_BASE$	0.00141825	5.75E-09	1.52619E-05
$PSO\_BASE\_CONS$	0.00102449	2.23E-07	6.32419E-05
$PSO\_BASE\_WEIGHT$	1.87E-07	<b>3.60E-16</b>	<b>2.14E-09</b>
$PSO\_BASE\_Vmax$	0.000242195	2.46E-08	1.42158E-05

Tab. 2 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací  $dim=1$

50 realizací	max	min	Median
$PSO\_BASE$	0.00143247	1.45E-09	2.16613E-05
$PSO\_BASE\_CONS$	0.00520002	2.95E-09	3.68758E-05
$PSO\_BASE\_WEIGHT$	2.10E-07	<b>6.72E-14</b>	<b>1.18E-09</b>
$PSO\_BASE\_Vmax$	0.00039384	1.21E-09	7.76E-06

Pro  $dim = 2$

Tab. 3 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a  $dim=2$

20 realizací	max	min	median
$PSO\_BASE$	0.903868	0.00390089	0.178456
$PSO\_BASE\_CONS$	1.25983	0.000251605	0.238399
$PSO\_BASE\_WEIGHT$	0.00762494	<b>1.27438E-05</b>	<b>0.000350173</b>
$PSO\_BASE\_Vmax$	0.697232	0.00323999	0.0764814

Tab. 4 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
<i>PSO_BASE</i>	0.0658275	0.000166693	0.00885603
<i>PSO_BASE_CONS</i>	0.0883626	0.000295637	0.0107987
<i>PSO_BASE_WEIGHT</i>	2.93471E-06	<b>8.7384E-10</b>	<b>1.96091E-07</b>
<i>PSO_BASE_Vmax</i>	0.0192718	8.93421E-06	0.00325713

Tab. 5 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2

100 realizací	max	min	median
<i>PSO_BASE</i>	0.0589695	4.67416E-06	0.00794454
<i>PSO_BASE_CONS</i>	0.0673086	8.56486E-05	0.00762554
<i>PSO_BASE_WEIGHT</i>	1.28614E-05	<b>6.45754E-10</b>	<b>1.7339E-07</b>
<i>PSO_BASE_Vmax</i>	0.0365429	5.27311E-06	0.00237412

Tab. 6 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2

200 realizací	max	min	median
<i>PSO_BASE</i>	0.0988516	1.31256E-05	0.00663495
<i>PSO_BASE_CONS</i>	0.101385	0.000079646	0.0083439
<i>PSO_BASE_WEIGHT</i>	1.12762E-05	<b>3.54742E-11</b>	<b>2.55474E-07</b>
<i>PSO_BASE_Vmax</i>	0.0321309	1.04144E-05	0.00248176

Pro dim = 5

Tab. 7 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací dim=5

20 realizací	max	min	median
<i>PSO_BASE</i>	2.00879	0.16785	0.720534
<i>PSO_BASE_CONS</i>	1.48456	0.0881851	0.439349
<i>PSO_BASE_WEIGHT</i>	0.00115471	<b>7.26E-06</b>	<b>5.45971E-05</b>
<i>PSO_BASE_Vmax</i>	1.88487	0.0546216	0.602908

Tab. 8 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací dim=5

50 realizací	max	min	median
<i>PSO_BASE</i>	1.86897	0.104219	0.78615
<i>PSO_BASE_CONS</i>	2.01172	0.0892511	0.655015
<i>PSO_BASE_WEIGHT</i>	0.000477393	<b>5.00E-06</b>	<b>5.59154E-05</b>
<i>PSO_BASE_Vmax</i>	2.82292	0.0727926	0.505786

### 6.1.1.2 Vyhodnocení

Výsledky testování jsou zaznamenány v tabulkách 1 – 8. Jednoznačně nejlepších výsledku dosahovala modifikace PSO se setrvačností, která našla nejpřesnější hodnoty minima a zároveň byla nejspolehlivější, jak vyplývá z ukazatele max a median. Pro dim=5 už je modifikace *PSO\_BASE\_WEIGHT* jedinou, která spolehlivě dosahuje kvalitních výsledků.

## 6.1.2 2. De Jongova funkce

Hodnota minima pro  $E_n$ :  $y = 0$

### 6.1.2.1 Naměřená data

Pro dim = 1 není definována.

Pro dim = 2

Tab. 9 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2

20 realizací	max	min	median
<i>PSO_BASE</i>	1.53546	0.00402712	0.243727
<i>PSO_BASE_CONS</i>	2.57739	0.00401585	0.133752
<i>PSO_BASE_WEIGHT</i>	0.0375226	<b>2.58362E-06</b>	<b>0.000661179</b>
<i>PSO_BASE_Vmax</i>	0.708759	0.000856526	0.0637962

Tab. 10 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
<i>PSO_BASE</i>	1.5984	0.000244547	0.145851
<i>PSO_BASE_CONS</i>	1.35352	0.00275658	0.0902817
<i>PSO_BASE_WEIGHT</i>	0.008065	<b>4.47723E-06</b>	<b>0.000165118</b>
<i>PSO_BASE_Vmax</i>	0.821864	0.000762629	0.0749604

Tab. 11 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2

100 realizací	max	min	median
<i>PSO_BASE</i>	1.73741	0.00257452	0.148463
<i>PSO_BASE_CONS</i>	1.04399	0.000665163	0.136451
<i>PSO_BASE_WEIGHT</i>	0.0331178	<b>1.67019E-07</b>	<b>0.000333119</b>
<i>PSO_BASE_Vmax</i>	1.06746	7.15991E-05	0.0641118

Tab. 12 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2

200 realizací	max	min	median
<i>PSO_BASE</i>	0.924034	0.000865323	0.138166
<i>PSO_BASE_CONS</i>	1.31868	0.00127977	0.163319
<i>PSO_BASE_WEIGHT</i>	0.421316	<b>7.7317E-08</b>	<b>0.000123314</b>
<i>PSO_BASE_Vmax</i>	0.723603	1.95817E-05	0.0430554

Pro dim = 5

Tab. 13 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5

20 realizací	max	min	median
<i>PSO_BASE</i>	382.939	5.7491	56.5293
<i>PSO_BASE_CONS</i>	249.27	8.6564	64.3675
<i>PSO_BASE_WEIGHT</i>	5.01456	<b>0.630194</b>	<b>2.42146</b>
<i>PSO_BASE_Vmax</i>	122.877	14.9963	54.6414

Tab. 14 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5

50 realizací	max	min	median
<i>PSO_BASE</i>	212.026	3.78778	58.6956
<i>PSO_BASE_CONS</i>	341.814	7.05566	61.3682
<i>PSO_BASE_WEIGHT</i>	4.66173	<b>0.00835555</b>	<b>2.035</b>
<i>PSO_BASE_Vmax</i>	185.451	1.74767	47.8875

### 6.1.2.2 Vyhodnocení

Data z testování pomocí 2. De Jongovy funkce jsou uvedena v tabulkách 9 – 14. Vzhledem k průběhu funkce (viz kapitola 2.3.1) lze očekávat menší úspěšnost a přesnost než u 1. De Jongovy funkce. Největší přesnosti opět dosáhla modifikace PSO se setrvačností. Za pozornost však stojí i výsledky dosahované modifikací *PSO\_BASE\_Vmax*, tedy omezení maximální rychlosti pro dvourozměrný problém, kde na rozdíl od předchozího testu dosahuje výrazně lepších výsledků než *PSO\_BASE* či *PSO\_BASE\_CONST*.

### 6.1.3 3. De Jongova funkce

Hodnota minima pro  $E_n$ :  $y = 0$

### 6.1.3.1 Naměřená data

Pro dim = 1

Tab. 15 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2

20 realizací	max	min	median
PSO_BASE	0.0250831	0.00115318	0.00878107
PSO_BASE_CONS	0.0249558	4.04233E-05	0.00395316
PSO_BASE_WEIGHT	0.000783633	<b>2.6151E-06</b>	<b>2.29385E-05</b>
PSO_BASE_Vmax	0.00709911	9.59118E-05	0.00234671

Tab. 16 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
PSO_BASE	0.0427015	0.000213975	0.0062415
PSO_BASE_CONS	0.031678	7.76898E-05	0.00531423
PSO_BASE_WEIGHT	0.000618553	<b>3.585E-07</b>	<b>3.18271E-05</b>
PSO_BASE_Vmax	0.0195358	1.60943E-05	0.00405385

Pro dim = 2

Tab. 17 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2

20 realizací	max	min	median
PSO_BASE	0.300329	0.0448619	0.12848
PSO_BASE_CONS	0.384395	0.0209963	0.103027
PSO_BASE_WEIGHT	0.00281026	<b>0.000111981</b>	<b>0.000671224</b>
PSO_BASE_Vmax	0.128655	0.0073189	0.0474771

Tab. 18 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
PSO_BASE	0.320335	0.0123414	0.129157
PSO_BASE_CONS	0.394556	0.0121081	0.120811
PSO_BASE_WEIGHT	0.00454365	<b>4.32445E-05</b>	<b>0.000548651</b>
PSO_BASE_Vmax	0.182595	0.0101939	0.0594019

Tab. 19 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2

100 realizací	max	min	median
PSO_BASE	0.532917	0.00913497	0.10912
PSO_BASE_CONS	0.378232	0.0175994	0.116789
PSO_BASE_WEIGHT	0.00578183	<b>3.76439E-05</b>	<b>0.000594547</b>
PSO_BASE_Vmax	0.203794	0.00327357	0.0684859

Tab. 20 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2

200 realizací	max	min	median
PSO_BASE	0.385791	0.00936642	0.12317
PSO_BASE_CONS	0.336677	0.00707099	0.117106
PSO_BASE_WEIGHT	0.0061186	<b>1.88388E-05</b>	<b>0.00062765</b>
PSO_BASE_Vmax	0.25082	0.00204488	0.0573214

Pro dim = 5

Tab. 21 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5

20 realizací	max	min	median
PSO_BASE	2.50264	0.665935	1.39042
PSO_BASE_CONS	2.57072	0.344205	1.40044
PSO_BASE_WEIGHT	0.0739593	<b>0.00394756</b>	<b>0.0138665</b>
PSO_BASE_Vmax	2.21812	0.715045	1.17492

Tab. 22 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5

50 realizací	max	min	median
PSO_BASE	2.86961	0.587488	1.59897
PSO_BASE_CONS	2.58114	0.504573	1.40665
PSO_BASE_WEIGHT	0.0442916	<b>0.00327153</b>	<b>0.0129792</b>
PSO_BASE_Vmax	2.87064	0.410159	1.42913

### 6.1.3.2 Vyhodnocení

V tabulkách 15 – 22 jsou uvedeny hodnoty získané při testování pomocí 3. De Jongovy funkce. Tabulka 16, tedy 50 realizací pro dim 1 dobře demonstruje význam mediánu jako statického ukazatele. Přestože *PSO\_BASE\_CONS* dosáhl o řád lepšího minima než *PSO\_BASE*, rozdíl v mediánu a nejhorší hodnotě (max) je mnohem menší. Z toho lze

odvodit, že šlo o velmi řídký jev a tedy nelze na základě minimální nalezené hodnoty tvrdit, že jeden, či druhý algoritmus je lepší.

Při testování pomocí 3. De Jongovy funkce se jako nejlepší projevil PSO se setrvačností, podobně jak tomu bylo v předešlých případech.

#### 6.1.4 4. De Jongova funkce

Hodnota minima pro  $E_n$ :  $y = 0$

##### 6.1.4.1 Naměřená data

Pro  $\dim = 1$

Tab. 23 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a  $\dim=1$

20 realizací	max	min	median
PSO_BASE	4.25318E-06	1.36774E-16	4.48017E-10
PSO_BASE_CONS	2.33913E-06	1.31288E+12	2.64802E-10
PSO_BASE_WEIGHT	2.26868E-13	<b>1.25059E-22</b>	<b>3.84535E-18</b>
PSO_BASE_Vmax	2.33662E-08	1.36012E-16	1.54211E-10

Tab. 24 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a  $\dim=1$

50 realizací	max	min	median
PSO_BASE	1.58732E-06	7.02107E-20	1.67146E-10
PSO_BASE_CONS	3.31783E-06	1.55161E-16	1.83531E-09
PSO_BASE_WEIGHT	8.78926E-15	<b>6.10706E-31</b>	<b>3.03291E-19</b>
PSO_BASE_Vmax	3.61097E-08	6.89646E-19	7.80602E-11

Pro  $\dim = 2$

Tab. 25 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a  $\dim=2$

20 realizací	max	min	median
PSO_BASE	0.000709374	8.35335E-08	7.77804E-05
PSO_BASE_CONS	0.0016163	1.08136E-06	3.23006E-05
PSO_BASE_WEIGHT	3.43438E-12	<b>8.35502E-17</b>	<b>2.84661E-14</b>
PSO_BASE_Vmax	3.04737E-05	4.96583E-09	8.0775E-07



Tab. 26 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
PSO_BASE	0.00783377	5.31417E-07	0.000152864
PSO_BASE_CONS	0.00496224	9.75937E-08	7.11019E-05
PSO_BASE_WEIGHT	1.24094E-10	<b>1.11153E-17</b>	<b>9.09589E-14</b>
PSO_BASE_Vmax	0.000482345	6.60443E-10	6.7018E-06

Tab. 27 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2

100 realizací	max	min	median
PSO_BASE	0.00386763	4.99793E-09	5.86519E-05
PSO_BASE_CONS	0.00619739	6.95197E-09	5.93662E-05
PSO_BASE_WEIGHT	2.15144E-10	<b>8.80395E-18</b>	<b>2.94639E-14</b>
PSO_BASE_Vmax	0.000855654	1.49322E-09	6.47351E-06

Tab. 28 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2

200 realizací	max	min	median
PSO_BASE	0.00805634	1.23743E-09	6.06246E-05
PSO_BASE_CONS	0.0136358	5.2099E-09	7.72699E-05
PSO_BASE_WEIGHT	1.44545E-10	<b>1.37585E-19</b>	<b>6.46011E-14</b>
PSO_BASE_Vmax	0.00152843	5.58936E-11	5.89913E-06

Pro dim = 5

Tab. 29 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5

20 realizací	max	min	median
PSO_BASE	7.03437	0.0487613	0.465759
PSO_BASE_CONS	2.61124	0.00719966	0.402524
PSO_BASE_WEIGHT	9.15032E-08	<b>3.03953E-11</b>	<b>2.20729E-09</b>
PSO_BASE_Vmax	3.0514	0.0434363	1.01974

Tab. 30 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5

50 realizací	max	min	median
PSO_BASE	5.55987	0.000159399	0.701191
PSO_BASE_CONS	11.1075	0.0195809	0.553206
PSO_BASE_WEIGHT	1.30655E-07	<b>2.51961E-11</b>	<b>4.81523E-09</b>
PSO_BASE_Vmax	3.74524	0.00639256	0.651711

#### 6.1.4.2 Vyhodnocení

Výsledky testů provedených pro 4. De Jongovu funkci (viz tabulky 23 – 30) mohou překvapit velkým rozdílem v přesnosti algoritmu PSO se setrvačností oproti ostatním. Je to způsobeno průběhem funkce (viz 3.1.3), který je na většině intervalu velmi pozvolný. Vysoká přesnost i pro  $\text{dim} = 5$  je velmi uspokojivá a ukazuje, že na podobný problém by mohla být verze *PSO\_BASE\_WEIGHT* s výhodou použita i uživatelem bez znalosti nastavení parametrů.

#### 6.1.5 Rastriginova funkce

Hodnota minima pro  $E_n$ :  $y = 0$

##### 6.1.5.1 Naměřená data

Pro  $\text{dim} = 1$

Tab. 31 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 20 realizací a  $\text{dim}=1$

20 realizací	max	min	median
PSO_BASE	4.97527	0.81093	1.63183
PSO_BASE_CONS	5.27583	0.0294067	1.29257
PSO_BASE_WEIGHT	1.14813	<b>1.36927E-05</b>	<b>0.0238963</b>
PSO_BASE_Vmax	2.68384	0.0532782	1.1038

Tab. 32 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 50 realizací a  $\text{dim}=5$

50 realizací	max	min	median
PSO_BASE	3.37422	0.0412169	1.61291
PSO_BASE_CONS	3.37256	0.054577	1.25209
PSO_BASE_WEIGHT	1.03807	<b>2.82585E-06</b>	<b>0.00485538</b>
PSO_BASE_Vmax	2.91235	0.0177057	1.06336

Pro  $\text{dim} = 2$

Tab. 33 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 20 realizací a  $\text{dim}=2$

20 realizací	max	min	median
PSO_BASE	3.5625	0.0198893	1.12155
PSO_BASE_CONS	3.27461	0.0645098	1.34005
PSO_BASE_WEIGHT	0.530383	<b>5.5043E-07</b>	<b>0.0016657</b>
PSO_BASE_Vmax	2.42724	0.116744	1.24848

Tab. 34 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
PSO_BASE	3.59007	0.0013333	1.25497
PSO_BASE_CONS	2.87436	0.204737	1.44513
PSO_BASE_WEIGHT	2.01053	<b>1.22062E-07</b>	<b>0.0258581</b>
PSO_BASE_Vmax	3.00207	0.00526094	1.10509

Tab. 35 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 100 realizací a dim=2

100 realizací	max	min	median
PSO_BASE	3.41731	0.102025	1.28542
PSO_BASE_CONS	3.2904	0.0132271	1.37553
PSO_BASE_WEIGHT	1.99235	<b>5.8435E-06</b>	<b>0.00571544</b>
PSO_BASE_Vmax	3.09526	0.00810496	1.08606

Tab. 36 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 200 realizací a dim=2

200 realizací	max	min	median
PSO_BASE	4.72163	0.00194257	1.34255
PSO_BASE_CONS	4.30154	0.00568607	1.31234
PSO_BASE_WEIGHT	1.98995	<b>8.65E-07</b>	<b>0.00604125</b>
PSO_BASE_Vmax	3.49621	0.00438963	1.04575

Pro dim = 5

Tab. 37 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 20 realizací a dim=5

20 realizací	max	min	median
PSO_BASE	24.0873	12.3348	18.0045
PSO_BASE_CONS	24.6743	6.63814	18.492
PSO_BASE_WEIGHT	13.9573	<b>0.909871</b>	<b>5.18071</b>
PSO_BASE_Vmax	22.7976	4.23284	15.655

Tab. 38 Rastriginova funkce. Testování Base verzí algoritmu  
– hodnoty pro 50 realizací a dim=5

50 realizací	max	min	median
PSO_BASE	28.9273	4.10698	18.4222
PSO_BASE_CONS	28.3025	7.51837	19.0378
PSO_BASE_WEIGHT	13.0736	<b>1.0141</b>	<b>4.38116</b>
PSO_BASE_Vmax	24.7029	7.87697	15.9296

### 6.1.5.2 Vyhodnocení

Extrémy Rastriginovy funkce jsou velmi prudké (viz 3.2.3) a tak se zde naplno uplatní schopnost PSO se setrvačností v počátečních fázích nalézt oblast extrémů a v koncových toto řešení zpřesnit, protože i malá změna jednoho z parametrů u této funkce vede k výrazné změně výsledné hodnoty účelové funkce. Výsledky testu (tabulky 31 – 38) ukazují, že jednoznačně nejkvalitnějších řešení dosahuje algoritmus *PSO\_BASE\_WEIGHT* i když i jeho spolehlivost není absolutní a v některých realizacích našel poměrně špatné výsledky (viz sloupec max v uvedených tabulkách). Přesto ale medián je v porovnání s ostatními velmi nízký. Na Rastriginově funkci dosahuje modifikace *PSO\_BASE\_Vmax* lepších výsledků než zbylé dva algoritmy *PSO\_BASE* a *PSO\_BASE\_CONST*.

### 6.1.6 Schwefelova funkce

Hodnota minima pro  $E_n$ :  $y = -418.983 \cdot \text{dim}$

#### 6.1.6.1 Naměřená data

Pro  $\text{dim} = 1$

Tab. 39 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a  $\text{dim}=1$

20 realizací	max	min	median
PSO_BASE	-416.43	-418.981	-418.94
PSO_BASE_CONS	-418.607	-418.982	-418.892
PSO_BASE_WEIGHT	-418.601	<b>-418.983</b>	-418.966
PSO_BASE_Vmax	-418.9	<b>-418.983</b>	<b>-418.979</b>

Tab. 40 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a  $\text{dim}=1$

50 realizací	max	min	median
PSO_BASE	-414.456	<b>-418.983</b>	-418.929
PSO_BASE_CONS	-416.539	<b>-418.983</b>	-418.878
PSO_BASE_WEIGHT	-415.99	<b>-418.983</b>	-418.972
PSO_BASE_Vmax	-418.629	<b>-418.983</b>	<b>-418.98</b>

Pro dim = 2

Tab. 41 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2

20 realizací	max	min	median
PSO_BASE	-736.956	-834.673	-810.764
PSO_BASE_CONS	-718.520	-836.704	-817.116
PSO_BASE_WEIGHT	-674.897	-837.759	-828.590
PSO_BASE_Vmax	-830.302	<b>-837.947</b>	<b>-835.766</b>

Tab. 42 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2

50 realizací	max	min	median
PSO_BASE	-697.475	-837.149	-798.734
PSO_BASE_CONS	-710.652	-837.937	-806.125
PSO_BASE_WEIGHT	-710.799	<b>-837.966</b>	-826.846
PSO_BASE_Vmax	-824.570	-837.935	<b>-836.955</b>

Tab. 43 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2

100 realizací	max	min	median
PSO_BASE	-687.940	-836.975	-808.173
PSO_BASE_CONS	-672.483	-837.544	-805.150
PSO_BASE_WEIGHT	-690.190	<b>-837.965</b>	-828.078
PSO_BASE_Vmax	-827.569	-837.949	<b>-836.515</b>

Tab. 44 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2

200 realizací	max	min	median
PSO_BASE	-702.748	-837.693	-809.727
PSO_BASE_CONS	-680.765	-837.806	-808.929
PSO_BASE_WEIGHT	-694.744	-837.958	-825.384
PSO_BASE_Vmax	-818.533	<b>-837.960</b>	<b>-836.821</b>

Pro dim = 5

Tab. 45 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5

20 realizací	max	min	median
PSO_BASE	-1153.54	-1697.08	-1455.01
PSO_BASE_CONS	-1292.84	-1757.35	-1476.39
PSO_BASE_WEIGHT	-1250.69	-1832.41	-1624.04
PSO_BASE_Vmax	-1611.6	<b>-2038.13</b>	<b>-1893.11</b>

Tab. 46 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5

50 realizací	max	min	median
PSO_BASE	-1282.39	-1729.53	-1478.04
PSO_BASE_CONS	-1284.98	-1699.13	-1461.4
PSO_BASE_WEIGHT	-1301.63	-1805.27	-1542.4
PSO_BASE_Vmax	-1680.99	<b>-2061.49</b>	<b>-1902.09</b>

### 6.1.6.2 Vyhodnocení

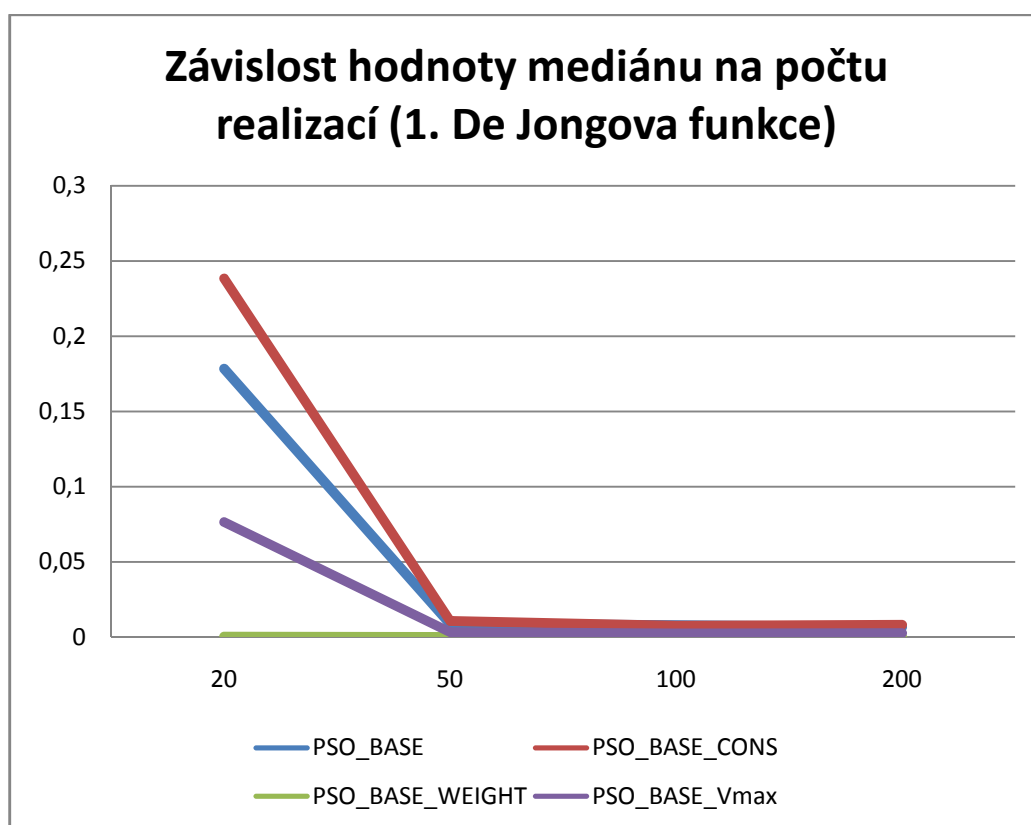
Testování na Schwefelově funkci (tabulky 39 – 46) prokázalo platnost No Free Lunch teorému (viz kapitola 1.1). Zatímco na předchozích pěti testovacích funkcích se nejvíce osvědčila modifikace PSO se setrvačností, na Schwefelově funkci dosahovala nejlepších výsledků ve většině případů modifikace *PSO\_BASE\_Vmax*, tedy PSO s omezením maximální rychlosti. Modifikaci PSO se setrvačností se v některých případech povedlo nalézt lepší minimální hodnotu, ale rozdíl oproti verzi s omezením maximální rychlosti byl minimální. Ovšem medián, tedy ukazatel spolehlivosti, jednoznačně ukazuje sílu verze *PSO\_BASE\_Vmax* pro řešení Schwefelovy funkce. S rostoucím počtem dimenzí a realizací se odstup od ostatních algoritmů zvětšuje.

V tabulce 40 je zachycen další jev potvrzující platnost NFL. Všem čtyřem testovaným algoritmům se povedlo nalézt hodnotu globálního minima a hodnoty mediánu dokazují, že se nejednalo o ojedinělý jev. V danou chvíli bychom mohli považovat všechny použité algoritmy za stejně silné. Další testování již ale prokázalo rozdíly v úspěšnosti.

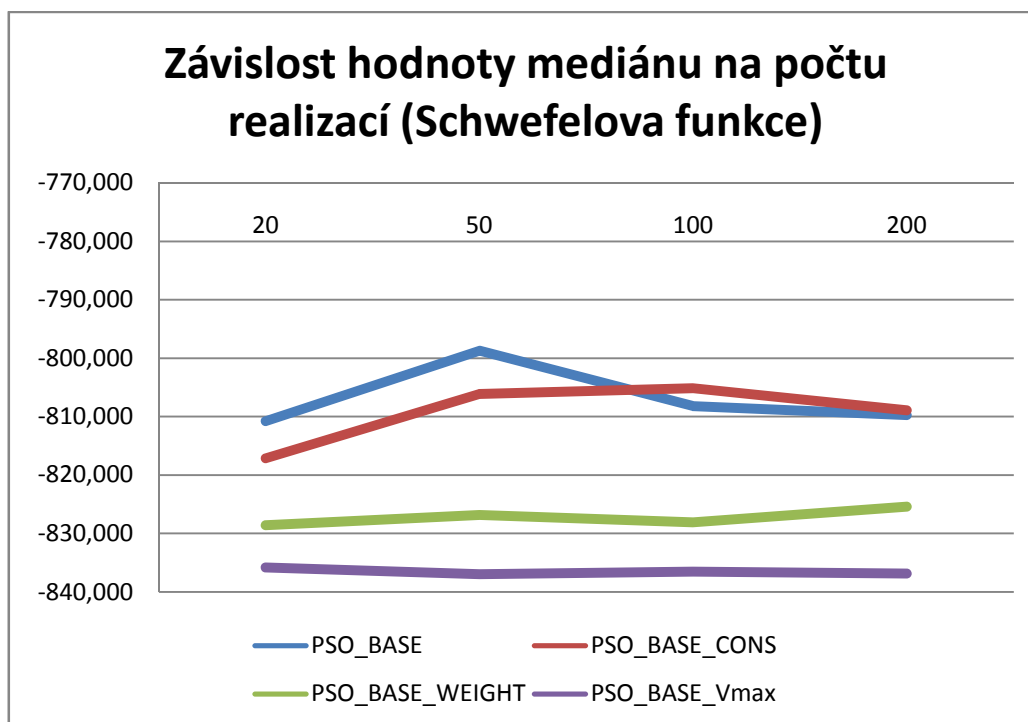
## 6.2 Celkové vyhodnocení testování BASE verzí

Na souboru testovacích funkcí byly testovány naprogramované funkce *PSO\_BASE*, *PSO\_BASE\_CONS*, *PSO\_BASE\_WEIGHT* a *PSO\_BASE\_Vmax*. Testování probíhalo s implicitním nastavením všech parametrů. Pro potřeby vyhodnocení bylo provedeno více realizací pro každý test a uchovány hodnoty nejhoršího a nejlepšího dosaženého výsledku a mediánu všech výsledků v daném balíku realizací (viz tabulky 1 – 46).

Testování bylo provedeno pro dimenzi problému 1, 2 a 5. A balíky realizací, ze kterých byly statistické hodnoty vypočteny, byly tvořeny 20, 50 a příp. 100 a 200 realizacemi. Testování na balíku 100 a 200 realizací nebylo provedeno pro  $\text{dim} = 5$  a  $\text{dim} = 1$  z důvodů časových, ale také protože zkušenost ukazuje, že obecně nejsou výsledky pro 50 realizací dramaticky odlišné od výsledků pro 100 a 200. Zatímco rozdíl mezi výsledky pro 20 a 50 realizací je velký. Tento trend je zachycen na obrázcích 15 a 16. Tyto grafy jsou vytvořeny z reálných dat naměřených při testech.



Obr. 15 Závislost hodnoty mediánu na počtu realizací (1. De Jongova funkce)



Obr. 16 Závislost hodnoty mediánu na počtu realizací (Schwefelova funkce)

Jako obecně nejefektivnější se při implicitním nastavení ukázala modifikace PSO se setrvačností, tedy verze PSO\_BASE\_WEIGHT a tuto lze pro nezkušené uživatele doporučit nejvíce. Ale v souladu s No Free Lunch teorémem (viz kapitola 1.1) nedosáhla ani tato verze nejlepšího výsledku pro všechny testovací funkce. Algoritmy s horšími výsledky nelze označit za špatné, ale je třeba zdůraznit, že algoritmus PSO obsahuje výraznou heuristickou složku a také je významně citlivý na nastavení vstupních parametrů. Problematikou nastavení parametrů PSO algoritmu se zabývají testy v kapitole 6.3.

### 6.3 Testování ADVANCED verzí algoritmu

Jak bylo uvedeno v kapitole 5, byla základní verze PSO algoritmu i 3 zvolené modifikace implementovány ve dvou verzích. Verze ADVANCED (značena ADV) umožňuje pokročilé zadávání vstupních parametrů. Vnitřní funkce samotného PSO algoritmu uvnitř těchto verzí je však totožná, nebo velmi podobná fungování BASE verzí algoritmu. Díky tomu lze testovat, zda při zadání jiných vstupních parametrů do stejné funkce existuje pravděpodobnost dosažení lepšího výsledku.

Pro návrh parametrů použitých při následném testování ADVANCED verzí algoritmu bylo použito PSO algoritmu ve verzi PSO\_BASE. Účelová funkce pro takový návrh byla



definována jako medián z více realizací algoritmu ve verzi ADVANCED, se zadanými parametry. Specimen v takovém případě definuje hranice použitelných hodnot.

Takovémuto jevu, aplikace jednoho evolučního algoritmu na druhý říkáme meta-evoluce. Cílem tohoto testu je dokázat, že stejný algoritmus, který v testech v kapitole 6.1 nepatřil pro danou testovací funkci k nejlepším, je schopen při vhodně zvolených parametrech dosáhnout kvalitnějších výsledků. Aby bylo možno výsledky srovnávat, byl pro testování nalezených parametrů určen balík 200 realizací a zvolen  $\dim = 2$ .

Testy nebyly prováděny na všech testovacích funkcích, především z důvodu vysoké časové náročnosti meta-evoluce, ale na těch funkcích, kde původní algoritmus dosahoval nejhorších výsledků, či byla vybrána jako vhodná pro demonstrování zpřesnění nalézaných řešení.

Hodnoty, které nalezne evoluční algoritmus, který hledá vhodné parametry, jsou ve formátu real. V tomto formátu jsou také následovně zadány do testovaného algoritmu. U počtu iterací a částic však dojde (vzhledem k povaze algoritmu) k zanedbání části za desetinou čárkou. Nedochozí k zaokrouhlení, ale k použití pouze celočíselné části.

### 6.3.1 Testování PSO\_ADV

První test byl proveden pomocí 1. De Jongovi funkce. Hodnoty parametrů určené pomocí meta-evoluce jsou následující (V závorce jsou uvedeny implicitní hodnoty parametrů pro PSO\_BASE) :

$$n = 39 \text{ (50)}$$

$$NP = 29 \text{ (20)}$$

$$c_1 = 0.9025024910681567 \text{ (2)}$$

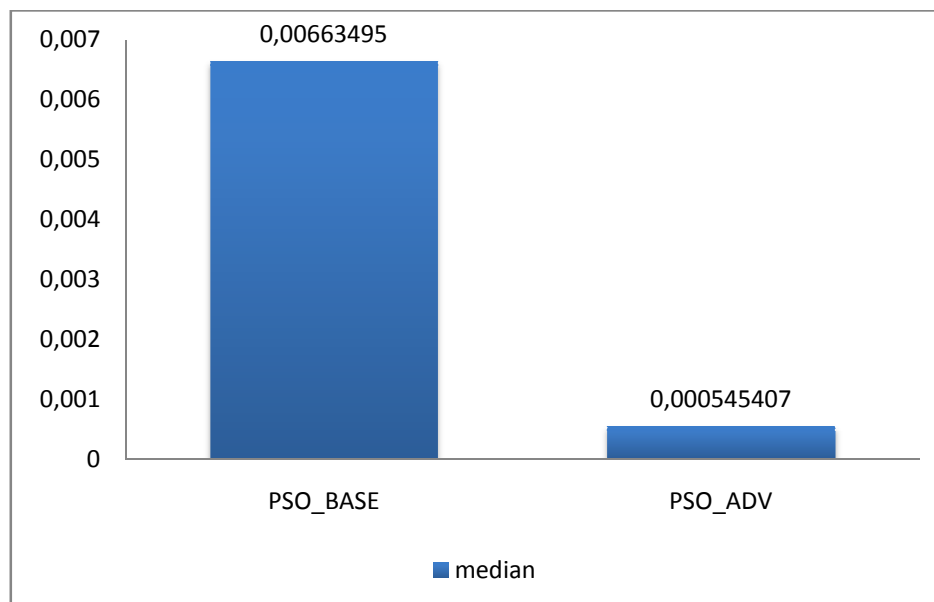
$$c_2 = 0.5446218075847105 \text{ (2)}$$

S použitím těchto hodnot byl proveden test algoritmu PSO\_ADV na 1. De Jongově funkci a výsledky toho testu jsou uvedeny v Tab. 47. Nejdůležitější je hodnota mediánu, která určuje, zda a jak se změnila spolehlivost algoritmu.

Tab. 47 Testování PSO\_ADV na 1. De Jongově funkci

1. De Jongova Funkce	Median	Min
PSO_BASE	0.00663495	1.31256E-05
PSO_ADV	<b>0.00054541</b>	<b>4.64E-08</b>
Rozdíl	0.00608954	1.30792E-05

S použitím parametrů získaných při hledání pomocí evolučního algoritmu, se hodnota mediánu nalézáných řešení snížila o jeden řád. Kvalita nalézáných řešení se tedy s použitím jiných parametrů výrazně zlepšila, přestože vnitřní algoritmy jsou totožné. Rozdíl v hodnotě mediánu je graficky zachycen na Obr. 17.



Obr. 17 Medián nalézáných řešení pro PSO\_BASE a PSO\_ADV s použitím evolučně nalezených parametrů pro 1. De Jongovu funkci.

### 6.3.2 Testování PSO\_ADV\_CONS

Při testování na balíku 200 realizací 2. De Jongovi funkce, vykázala nejhorší hodnotu mediánu modifikace PSO algoritmu s constriction faktorem. Nalezené parametry pomocí meta-evoluce:

$$n = 33 \text{ (50)}$$

$$NP = 29 \text{ (20)}$$

$$c_1 = 0.4444238541653327 \text{ (2)}$$

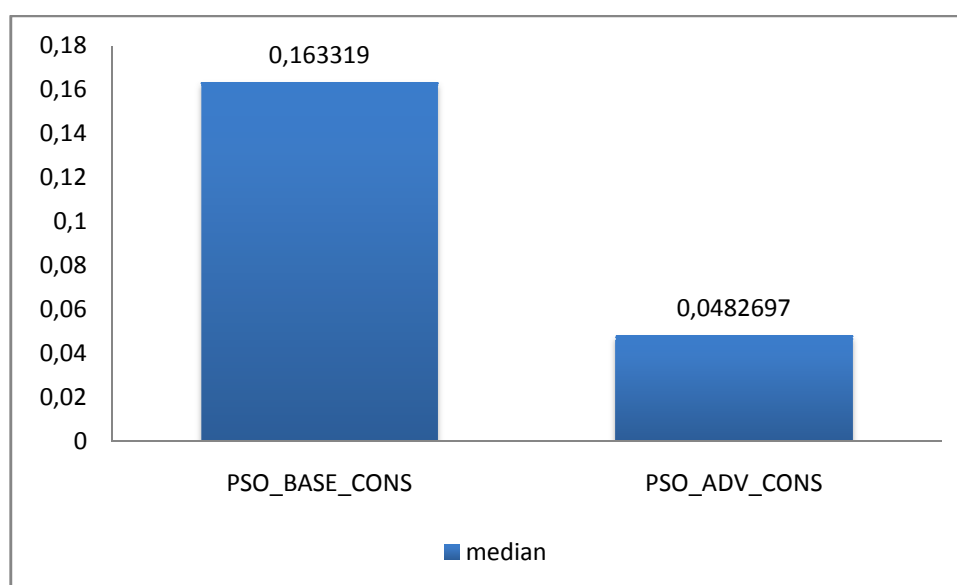
$$c_2 = 1.093756454566524 \text{ (2)}$$

cons = 0.12891914861246834 (0.729)

Jak vyplývá z tabulky 48 a zobrazení na Obr. 18, povedlo se pomocí nalezených parametrů výrazně snížit hodnotu mediánu nalézáných řešení.

Tab. 48 Testování PSO\_ADV\_CONS na 2. De Jongově funkci

2. De Jongova Funkce	median	min
PSO_BASE_CONS	0.163319	0.00127977
PSO_ADV_CONS	<b>0.0482697</b>	<b>0.000375608</b>
Rozdíl	0.1150493	0.000904162



Obr. 18 Medián nalézáných řešení pro PSO\_BASE\_CONS a PSO\_ADV\_CONS s použitím evolučně nalezených parametrů, pro 2. De Jongovu funkci

### 6.3.3 Testování PSO\_ADV\_WEIGHT

Při testování BASE verzí algoritmu a modifikací (viz kapitola 6.1) byla Schwefelova funkce jedinou, kde modifikace PSO se setrvačností nedosáhla nejlepšího výsledku. Pro tuto funkci byla tedy pomocí meta-evoluce navržena sada vstupních parametrů, která překvapivě vedla k hodnotě mediánu pro 200 realizací přímo v globálním minimu této funkce. Díky evolučnímu návrhu parametrů bylo dosaženo výrazného zvýšení přesnosti modifikace se setrvačností pro řešení Schwefelovy funkce. Viz Tab. 49 a Obr. 19.

n = 38 (50)

$$NP = 23 \text{ (20)}$$

$$c_1 = 0.17722922747570546 \text{ (2)}$$

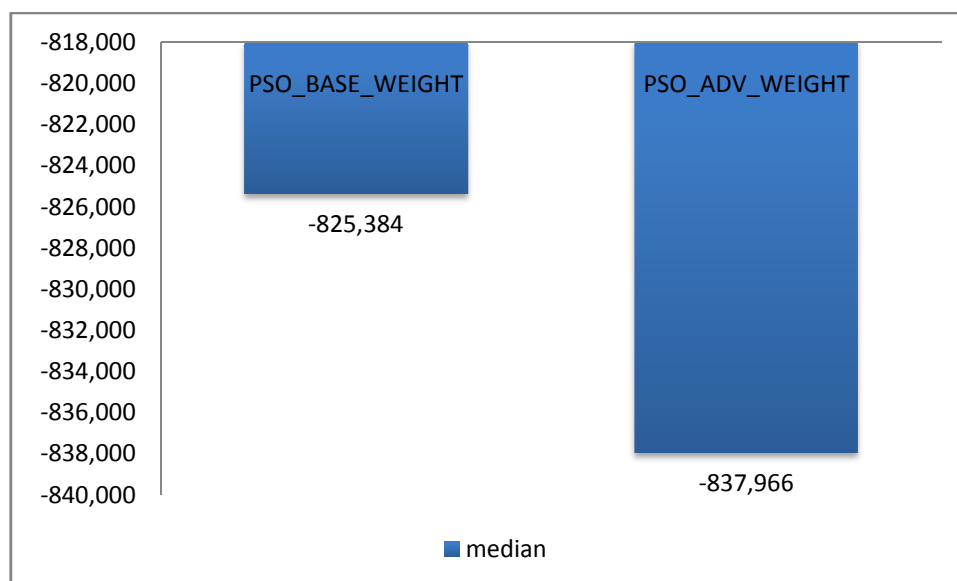
$$c_2 = 1.9573047745747583 \text{ (2)}$$

$$w_{\text{start}} = 0.5401583177347188 \text{ (0.9)}$$

$$w_{\text{end}} = 0.12069187333783049 \text{ (0.4)}$$

Tab. 49 Testování PSO\_ADV\_WEIGHT na Schwefelově funkci

Schwefelova funkce	median	min
PSO_BASE_WEIGHT	-825.384	-837.958
PSO_ADV_WEIGHT	<b>-837.966</b>	<b>-837.966</b>
Rozdíl	12.582	0.008



Obr. 19 Medián nalézaných řešení pro PSO\_BASE\_WEIGHT a PSO\_ADV\_WEIGHT s použitím evolučně nalezených parametrů, pro Schwefelovu funkci

### 6.3.4 Testování PSO\_ADV\_Vmax

Poslední modifikace, PSO s omezením maximální rychlosti byla otestována pro Rastriginovu funkci, která představuje složitý problém s vysokou hustotou hlubokých

extrémů. I v tomto případě vedl evoluční postup navržení parametrů k zlepšení hodnoty mediánu pro 200 realizací. Viz Tab. 50 a Obr. 20. Použité hodnoty jednotlivých parametrů:

$n = 33$  (50)

$NP = 25$  (20)

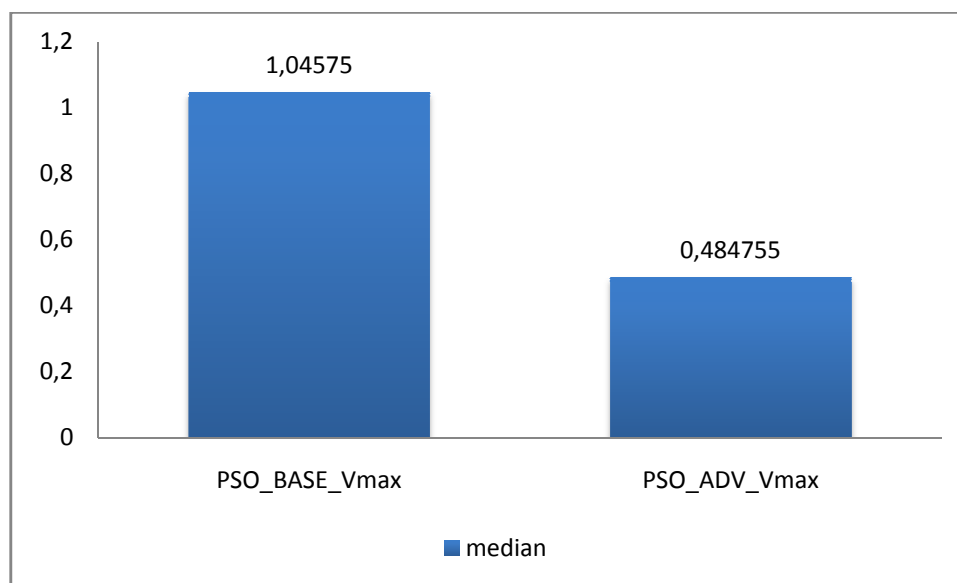
$c_1 = 0.4131872653220504$  (2)

$c_2 = 0.42165824487887704$  (2)

$V_{\max} = \{ 0.4039832131151063, 1.1283206423199865 \}$  (není definováno)

Tab. 50 Tab. 51 Testování PSO\_ADV\_Vmax na Rastriginově funkci

Rastriginova funkce	median	min
PSO_BASE_Vmax	1.04575	0.00438963
PSO_ADV_Vmax	<b>0.484755</b>	<b>6.71892E-05</b>
Rozdíl	0.560995	0.004322441



Obr. 20 Medián nalézáných řešení pro PSO\_BASE\_Vmax a PSO\_ADV\_Vmax s použitím evolučně nalezených parametrů, pro Rastriginovu funkci

## 7 SOUHRN PRAKTICKÉ ČÁSTI

V praktické části práce byl implementován algoritmus PSO a jeho vybrané modifikace v prostředí Mathematica, s důrazem na vytvoření takových prostředků, které budou pohodlné a snadno použitelné pro koncového uživatele. Při vytváření kódu byl dbán důraz na přehlednost a snadnou možnost úpravy. Současně byly prozkoumány možnosti časové a prostorové optimalizace kódu tak, aby výsledný algoritmus dosahoval dobrého výsledku i z hlediska nároků na výpočetní výkon.

Vytvořené implementace byly dále testovány pomocí testovacích funkcí a zpětně aplikovány pro návrh parametrů, jako důkaz funkčnosti a zároveň možnosti zlepšit dosahované přesnosti pomocí pokročilejšího zadání vstupních parametrů, které již vyžaduje znalost principu chování PSO Algoritmu. Pro uživatele bez těchto znalostí byly vytvořeny takové verze, které nevyžadují další vstupní parametry a přitom dosahují kvalitních výsledků.

Výsledky testování jsou tabulkově zaznamenány a slovně i graficky vyhodnoceny.

### 7.1 Časová a prostorová optimalizace

Při vytváření kódu byl brán ohled na časovou a prostorovou (paměťovou) náročnost výsledného algoritmu. Již v přípravné fázi byla kromě studování dokumentace k prostředí Wolfram Mathematica provedena také řada testů různých funkcí s cílem zjistit jejich časovou a paměťovou náročnost a určit jejich vhodnost pro implementaci jednotlivých složek PSO algoritmu.

Z hlediska časové optimalizace například využití funkce Table místo funkce For, zrychlí daný úsek kódu až 20x, přestože výsledek je totožný. Podobně byli testovány různé způsoby práce s vektory a další předpokládané zdroje vyšší časové náročnosti. Výsledná implementace je i díky těmto poznatkům velmi rychlá a režie nezabírá nepřijatelné množství času.

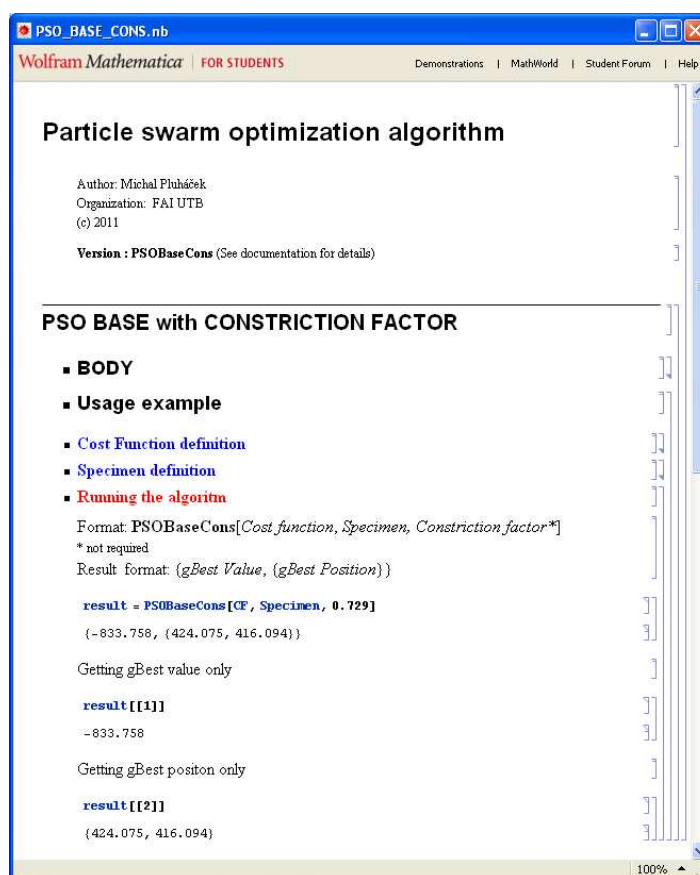
Paměťové nároky byli též brány v potaz a především díky využití lokálních proměnných (funkce Module) nezatěžují paměť data, se kterými se aktuálně nepracuje a maximální výpočetní kapacita je tak stále přístupná pro probíhající výpočet. Stejně tak v případě ekvivalence více funkcí, byly voleny ty, jejichž paměťové nároky jsou menší (pokud toto nebylo v přímém rozporu s časovou optimalizací).

## 7.2 Vytvořené soubory

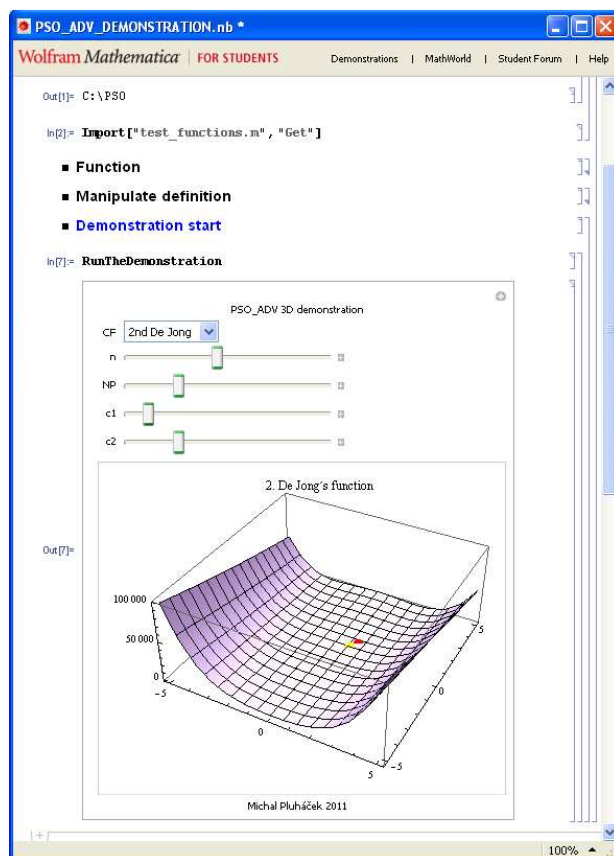
Všechny soubory jsou obsaženy ve složce PSO na přiloženém CD.

Celkem bylo vytvořeno osm verzí PSO algoritmu a tyto jsou jako samostatné notebooky (viz Obr. 21) ve složkách „en“ a „cz“ v anglické resp. české jazykové mutaci. Ve složce „demo“ se nachází notebook s 3D demonstrací algoritmu PSO\_ADV (viz Obr. 22).

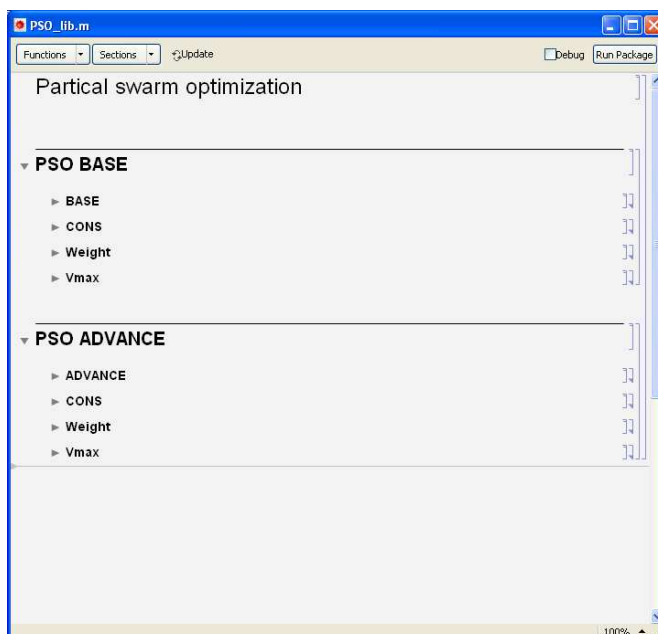
Knihovny (package, m-file) jsou ve složce „lib“. Knihovna PSO\_lib obsahuje všechny naprogramované verze (viz obr. 23). Knihovny PSO\_ADV\_lib a PSO\_BASE\_lib obsahují pouze příslušné verze. Navíc je umístěna ve složce „lib“ knihovna „test\_functions.m“, která obsahuje všechny použité testovací funkce. Tato knihovna je použita například v 3D demonstraci PSO\_ADV algoritmu.



Obr. 21 Ukázka demonstrativního notebooku pro verzi PSO\_BASE\_CONS



Obr. 22 3D demonstrace algoritmu PSO\_ADV vytvořená v prostředí Mathematica



Obr. 23 Knihovna PSO\_lib.m obsahuje všech 8 vytvořených verzí PSO algoritmu



## 7.3 Spuštění a prohlížení kódu

### 7.3.1 Wolfram Mathematica

Všechny soubory .nb a .m jsou vytvořeny v Mathematice v 8. Otestováno i ve verzi 7. Kompatibilita s verzí 6 a starší není zaručena. Při spouštění ukázkového notebooku je doporučeno využít možnost „Evaluate notebook“ v záložce „Evaluation“. Tato volba postupně spustí všechny buňky a zaručí, že nedojde k vynechání některé z nutných inicializací.

Návod na použití knihoven m-file (.m) je přiložen v adresáři lib, který tyto knihovny obsahuje.

### 7.3.2 Wolfram CDF Player

Pro prohlížení notebooků či knihoven vytvořených ve Wolfram Mathematice lze využít aplikaci Wolfram CDF Player, která je zdarma k dispozici na internetových stránkách společnosti Wolfram, na adrese:

<http://www.wolfram.com/products/player/>

Tato aplikace neumožňuje vykonávat kód. Slouží pouze k prohlížení.

### 7.3.3 PDF

Pro případ, že není k dispozici žádná z aplikací schopných otevřít soubor typu Mathematica notebook (.nb) jsou ve složkách „cz“ a „en“, které obsahují demonstrační notebooky pro naprogramované verze PSO algoritmu, pdf soubory s vytištěnou verzí těchto notebooků.

## ZÁVĚR

Hlavním cílem práce bylo vytvoření knihovny pro prostředí Wolfram Mathematica, která obsahuje funkce pro optimalizaci pomocí PSO algoritmu a jeho vybraných modifikací. Všechny vytvořené verze byly testovány na zvolené sadě testovacích funkcí a výsledky testů zpracovány a vyhodnoceny.

Mimo knihovny byly vytvořeny také notebooky, které uživatele seznamují s funkcí jednotlivých vytvořených verzí PSO algoritmu a mohou být samostatně využity pro řešení optimalizačních úloh. Tyto jsou, stejně jako uživatelské návody k vytvořené knihovně, v české a anglické jazykové mutaci.

Při vytváření všech programů bylo dbáno zásad časové a prostorové optimalizace kódu a kladen důraz na přehlednost a uživatelskou přívětivost. Veškeré naprogramované soubory, vytvořené a popsané v praktické části, jsou obsaženy na přiloženém CD.

Knihovna pro optimalizaci za použití algoritmu PSO bude umístěna na univerzitním serveru pro rychlé výpočty a bude tak přístupná pro využití v rámci výuky a výzkumu.

V teoretické části práce byla zpracována literární rešerše, popisující principy evolučních algoritmů a optimalizačních problémů, představující vybrané testovací funkce a také zvolené vývojové prostředí Wolfram Mathematica 8.

Postupy pro použití programových výstupů práce jsou popsány v závěru praktické části, stejně jako v přiloženém uživatelském návodu a komentářích v kódu.

## CONCLUSION

The main goal of the work was to create a Wolfram Mathematica library, that would contain functions to optimize using the PSO algorithm and its selected modifications. All the version created have been tested on the specified set of test functions and results of the tests have been processed and evaluated.

Aside from the library, several notebooks have been created. These notebooks introduce the user to the individual modifications of the PSO algorithm and can be separately used to solve optimization tasks. These notebooks, as well as user documentation for the library, exist in both Czech and English languages.

During the development of all programs, the principles of space and time code optimization have been honored. The code is highly readable and user-friendly. All the resulting files, created and described in the practical part, are included on the attached CD.

The PSO optimization library will be hosted on the university fast-computation server, therefore being accessible for use in research and education.

The theoretical part contains a literature search, describing the principles of evolutionary algorithms and optimization problems, selected test functions and the Wolfram Mathematica 8 environment.

The instructions for using the program outputs of the work are described in the last chapter of the practical part, as well as in the attached user manual and source code comments.

**SEZNAM POUŽITÉ LITERATURY**

- [1] DORIGO M., Ant Colony Optimization and Swarm Intelligence, Springer, 2006, ISBN 35402267292.
- [2] HOSTE, Jim. Mathematica DeMYSTiFied. McGraw-Hill Professional, 2008. 408 s. ISBN 978-0071591447.
- [3] ZELINKA, I. Umělá inteligence - hrozba nebo naděje. BEN - technická literatura, 2003, ISBN 80-7300-068-7.
- [4] ZELINKA I., OPLATKOVÁ Z., ŠEDA M., OŠMERA P., VČELAŘ F., Evoluční výpočetní techniky - principy a aplikace, BEN, Praha, 2008, ISBN 80-7300-218-3.
- [5] RUSKEEPAA, Heikki. Mathematica Navigator: Mathematics, Statistics and Graphics, Third Edition. Academic Press, 2009. 1136 s. ISBN 978-0123741646.
- [6] ZELINKA, Ivan. Aplikovaná Informatika. Zlín. UTB, 1999. 183 s. ISBN 80-214-1423-5.
- [7] ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. BEN, 2002, 190 s. ISBN 80-7300-069-5.
- [8] EBERHART R., KENNEDY J., Swarm Intelligence, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, 2001, ISBN 15586059593.
- [9] <http://www.wolfram.com/mathematica/>
- [10] <http://www.wolfram.com/mathematica/quick-revision-history.html>
- [11] <http://darwin-online.org.uk/>

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

ACO	Ant Colony Optimization
DE	Diferenciální evoluce
dim	Dimenze, rozměr problému
EA	Evoluční algoritmus, evoluční algoritmy
NFL	NO Free Lunch (theorem)
PSO	Particle swarm optimization.
SOMA	SamoOrganizující se, Migrační Algoritmus

**SEZNAM OBRÁZKŮ**

Obr. 1 Grafické vyjádření No Free Lunch Teorému .....	13
Obr. 2 První De Jongova funkce ve 2D .....	24
Obr. 3 První De Jongova funkce ve 3D .....	24
Obr. 4 Třetí De Jongova funkce ve 2D .....	25
Obr. 5 Třetí De Jongova funkce ve 3D .....	25
Obr. 6 Čtvrtá De Jongova funkce ve 2D .....	26
Obr. 7 Čtvrtá De Jongova funkce ve 3D .....	27
Obr. 8 Druhá De Jongova funkce ve 3D .....	28
Obr. 9 Schwefelova funkce ve 2D .....	29
Obr. 10 Schwefelova funkce ve 3D .....	29
Obr. 11 Rastriginova funkce ve 2D .....	30
Obr. 12 Rastriginova funkce ve 3D .....	31
Obr. 13 Notebook v prostředí Mathematica 8 se zobrazením buněk a sekcí.....	32
Obr. 14 Návoděda pro funkci Table v Mathematice v. 8 .....	33
Obr. 15 Závislost hodnoty mediánu na počtu realizací (1. De Jongova funkce).....	55
Obr. 16 Závislost hodnoty mediánu na počtu realizací (Schwefelova funkce) .....	56
Obr. 17 Medián nalézáných řešení pro PSO_BASE a PSO_ADV s použitím evolučně nalezených parametrů pro 1. De Jongovu funkci. ....	58
Obr. 18 Medián nalézáných řešení pro PSO_BASE_CONS a PSO_ADV_CONS s použitím evolučně nalezených parametrů, pro 2. De Jongovu funkci.....	59
Obr. 19 Medián nalézáných řešení pro PSO_BASE_WEIGHT a PSO_ADV_WEIGHT s použitím evolučně nalezených parametrů, pro Schwefelovu funkci .....	60
Obr. 20 Medián nalézáných řešení pro PSO_BASE_Vmax a PSO_ADV_Vmax s použitím evolučně nalezených parametrů, pro Rastriginovu funkci.....	61
Obr. 21 Ukázka demonstrativního notebooku pro verzi PSO_BASE_CONS.....	63
Obr. 22 3D demonstrace algoritmu PSO_ADV vytvořená v prostředí Mathematica .....	64
Obr. 23 Knihovna PSO_lib.m obsahuje všech 8 vytvořených verzí PSO algoritmu.....	64

**SEZNAM TABULEK**

Tab. 1 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=1 .....	42
Tab. 2 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací dim=1 .....	42
Tab. 3 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2 .....	42
Tab. 4 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	43
Tab. 5 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2 .....	43
Tab. 6 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	43
Tab. 7 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací dim=5 .....	43
Tab. 8 První De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací dim=5 .....	43
Tab. 9 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	44
Tab. 10 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	44
Tab. 11 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2 .....	44
Tab. 12 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	45
Tab. 13 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5 .....	45
Tab. 14 Druhá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5 .....	45
Tab. 15 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2 .....	46
Tab. 16 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	46

Tab. 17 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2 .....	46
Tab. 18 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	46
Tab. 19 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2 .....	47
Tab. 20 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	47
Tab. 21 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5 .....	47
Tab. 22 Třetí De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5 .....	47
Tab. 23 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=1 .....	48
Tab. 24 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=1 .....	48
Tab. 25 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2 .....	48
Tab. 26 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	49
Tab. 27 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2 .....	49
Tab. 28 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	49
Tab. 29 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5 .....	49
Tab. 30 Čtvrtá De Jongova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5 .....	49
Tab. 31 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=1 .....	50
Tab. 32 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5 .....	50



Tab. 33 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2 .....	50
Tab. 34 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	51
Tab. 35 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2 .....	51
Tab. 36 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	51
Tab. 37 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5 .....	51
Tab. 38 Rastriginova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5 .....	51
Tab. 39 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=1 .....	52
Tab. 40 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=1 .....	52
Tab. 41 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=2 .....	53
Tab. 42 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=2 .....	53
Tab. 43 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 100 realizací a dim=2 .....	53
Tab. 44 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 200 realizací a dim=2 .....	53
Tab. 45 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 20 realizací a dim=5 .....	54
Tab. 46 Schwefelova funkce. Testování Base verzí algoritmu – hodnoty pro 50 realizací a dim=5 .....	54
Tab. 47 Testování PSO_ADV na 1. De Jongově funkci .....	58
Tab. 48 Testování PSO_ADV_CONS na 2. De Jongově funkci .....	59
Tab. 49 Testování PSO_ADV_WEIGHT na Schwefelově funkci .....	60
Tab. 50 Tab. 51 Testování PSO_ADV_Vmax na Rastriginově funkci .....	61

## SEZNAM PŘÍLOH

- P I Uživatelský manuál ke knihovně PSO\_lib v anglickém jazyce.
- P II Uživatelský manuál ke knihovně PSO\_lib v českém jazyce.

# **PŘÍLOHA P I: UŽIVATELSKÝ MANUÁL KE KNIHOVNĚ PSO\_LIB V ANGLICKÉM JAZYCE.**

## **USER GUIDE**

### **PSO\_lib.m**

For Wolfram Mathematica 8

Created by Michal Pluháček, FAI UTB, 2011

### **How to import?**

First, set the active directory, to where the m-file is located

#### **Example:**

**SetDirectory**["C:/PSO"]

Then, use following import function:

**Import**["PSO\_lib.m", "Get"]

Warning: Import::nfil: "File not found during Import.", means that the active directory or file name wasn't set correctly.

### **How to use the functions?**

Each function is called by its name and proper format of input parameters.

**PSOBase**[Cost function, Specimen]

**PSOBaseCons**[*Cost function, Specimen, Constriction factor\**]

**PSOBaseWeight**[*Cost function, Specimen, Starting weight\*, Ending weight\**]

**PSOBaseVmax**[*Cost function, Specimen, Velocity factor \**]

- Velocity factor is single number, used to calculate the max velocity in each dimension, using formula:  $V_{max} = (Range\ To - Range\ From) / Velocity\ Factor$

**PSOAdv**[*Cost function, Specimen, Iterations, Number of particles,  $c_1$ ,  $c_2$* ]

**PSOAdvCons**[*Cost function, Specimen, Iterations, Number of particles,  $c_1$ ,  $c_2$ , Constriction factor*]

**PSOAdvWeight**[*Cost function, Specimen, Iterations, Number of particles,  $c_1$ ,  $c_2$ , Starting weight, Ending weight*]

**PSOAdvVmax**[*Cost function, Specimen, Iterations, Number of particles,  $c_1$ ,  $c_2$ , Vector of max. velocities*]

Input marked '\*' is not required. Default value is used.

### **Specimen format**

Correct specimen format:  $\{\{Type, \{From, To\}\}, \{Type, \{From, To\}\}, \dots, \{Type, \{From, To\}\}\}$

Only Real or Integer type and values are accepted.

### **Cost function format**

When using self-created cost function, following rules must be obeyed.

- a) Cost function has one input – array of numbers.
- b) Output is single number, with no vector brackets.

### **Result format**

Return value has following format: {gBest value, gBest position} where:

- gBest value is single number.
- gBest position is vector of similar length as is the dimension. Contains parameters to cost function, which will return the gBest value.

Use [[1]] or [[2]] to get only value or position of gBest.

### **Example:**

**PSOBase**[Cost function, Specimen][[1]] – only value of gBest is returned.

## PŘÍLOHA P II: UŽIVATELSKÝ MANUÁL KE KNIHOVNĚ PSO\_LIB V ČESKÉM JAZYCE.

# UŽIVATELSKÝ MANUÁL

### PSO\_lib.m

Pro Wolfram Mathematicu 8

Vytvořil: Michal Pluháček, FAI UTB, 2011

### Načtení knihovny

Nejdříve, je třeba nastavit cestu ke složce, kde se nachází knihovna PSO\_lib.m

#### Příklad:

**SetDirectory["C:/PSO"]**

Poté lze použít následující funkci:

**Import["PSO\_lib.m", "Get"]**

Warning: Import::nfil: "File not found during Import. Tato chyba znamená, že se soubor nepovedlo nalézt. Zkontrolujte správnost zadání cesty a názvu souboru.

### Použití funkcí

Funkce jsou volány pomocí jména a požadovaných parametrů.

**PSOBase**[Účelová funkce, Specimen]

**PSOBaseCons**[Účelová funkce, Specimen, Constriction factor\*]

**PSOBaseWeight**[Účelová funkce, Specimen, Počáteční setrvačnost\*, Konečná setrvačnost\*]

**PSOBaseVmax**[Účelová funkce, Specimen, Faktor rychlosti \*]

- Faktor rychlosti je jedno číslo, které určuje omezení rychlosti v každé dimenzi dle následujícího vzorce:  $V_{max} = (\text{horní mez} - \text{dolní mez}) / \text{faktor rychlosti}$

**PSOAdv**[Účelová funkce, Specimen, Iterace, Počet částic,  $c_1$ ,  $c_2$ ]

**PSOAdvCons**[Účelová funkce, Specimen, Iterace, Počet částic,  $c_1$ ,  $c_2$ , Constriction factor]

**PSOAdvWeight**[*Účelová funkce, Specimen, Iterace, Počet částic,  $c_1$ ,  $c_2$ , Počáteční setrvačnost, Konečná setrvačnost*]

**PSOAdvVmax**[*Účelová funkce, Specimen, Iterace, Počet částic,  $c_1$ ,  $c_2$ , Vector of max. velocities*]

Vstupy označené pomocí '\*' nejsou povinné, v případě nezadání, bude použita implicitní hodnota

### Formát specimenu

Správný formát speicmenu:  $\{\{Typ, \{Dolní\ mez, Horní\ mez\}\}, \{Typ, \{Dolní\ mez, Horní\ mez\}\}, \dots, \{Typ, \{Dolní\ mez, Horní\ mez\}\}\}$

Only Real or Integer Typ and values are accepted.

### Formát účelová funkce

Účelová funkce musí splňovat dvě pravidla:

- c) Účelová funkce má jeden vstup – pole hodnot.
- d) Výstupem je jedno číslo bez vektorových závorek.

### Formát výsledku

Vrácená hodnota má následující formát: {gBest hodnota, gBest pozice} kde:

- gBest hodnota je jedno číslo.
- gBest pozice je vektor o délce dimenze problému a obsahuje parametry, který při zadání do účelové funkce vrátí hodnotu gBest (první buňka).

Pro získání pouze hodnoty nebo pozice gBest použijte volání [[1]] resp. [[2]]

### Příklad:

**PSOBase**[*Účelová funkce, Specimen*][[1]] – vrátí pouze hodnotu gBest.

\*gBest – nejlepší nalezené řešení v populaci