

Pokročilé varianty algoritmu PSO v jazyce C

Bc. Tomáš Kadavý

Diplomová práce
2016



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš Kadavý**
Osobní číslo: **A14446**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **prezenční**

Téma práce: **Pokročilé varianty algoritmu PSO v jazyce C**
Téma anglicky: **Advanced Variants of the PSO Algorithm in the C Language**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Naprogramujte vybrané pokročilé varianty algoritmu PSO v jazyce C.
3. Implementujte benchmark knihovnu IEEE CEC 2015.
4. Proveďte testování a statistické vyhodnocení výkonnosti naprogramovaných algoritmů.
5. Graficky srovnajte výsledky jednotlivých algoritmů.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **PLUHACEK, Michal.** PSO Algoritmus v prostředí Mathematica. Zlín, 2011. Diplomová práce. Univerzita Tomáše Bati ve Zlíně.
2. **RIGET, Jacques; VESTERSTROM, Jakob S.** A diversity-guided particle swarm optimizer—the ARPSO. Dept. Comput. Sci., Univ. of Aarhus, Aarhus, Denmark, Tech. Rep, 2002, 2: 2002.
3. **NEPOMUCENO, Filipe V.; ENGELBRECHT, Andries P.** A self-adaptive heterogeneous pso for real-parameter optimization. In: Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 2013. p. 361–368.
4. **ZHAN, Zhi-Hui, et al.** Orthogonal learning particle swarm optimization. Evolutionary Computation, IEEE Transactions on, 2011, 15.6: 832–847.
5. **SHI, Yuhui; EBERHART, Russell.** A modified particle swarm optimizer. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, 1998. p. 69–73.
6. **CHEN, Q., et al.** Problem Definitions and Evaluation Criteria for CEC 2015 Special Session on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization.

Vedoucí diplomové práce:

Ing. Michal Pluháček

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

5. února 2016

Termín odevzdání diplomové práce:

20. května 2016

Ve Zlíně dne 5. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



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


Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne


.....
podpis diplomanta

ABSTRAKT

Cílem práce je implementace pokročilých variant algoritmu PSO v jazyce C a jejich otestování a srovnání za použití knihovny IEEE CEC 2015. Teoretická část popisuje obecnou problematiku hejnových algoritmů, popisuje základní princip PSO algoritmu a poté tři jeho pokročilé varianty, jimiž jsou ARPSO, HPSO a OLPSO. Následně jsou popsány testovací funkce v knihovně IEEE CEC 2015, prostředí Wolfram Mathematica, které se použije pro srovnání výsledků a programovací jazyk C. Praktická část pak obsahuje popis implementace algoritmů, jejich vnitřní strukturu a manuál k jejich ovládní. V závěru praktické části jsou srovnány jejich výsledky v přehledných tabulkách a grafech.

Klíčová slova: hejnové algoritmy, inteligence hejna, PSO, ARPSO, HPSO, OLPSO, jazyk C, IEEE CEC 2015

ABSTRACT

The goal of this work is the implementation of advanced variants of PSO algorithm in C language and further the testing and comparison of implemented methods using the IEEE CEC 2015 benchmark library. The theoretical part describes the general issues of swarm algorithms, describes the basic principle of PSO algorithm and then its three advanced modifications: the ARPSO, HPSO and OLPSO. After that the test functions in the IEEE CEC 2015 benchmark library are described. Also environment Wolfram Mathematica, which is used for the comparison of results and the C programming language are described. The practical part contains a description of the algorithm implementation, description of the internal structure of the algorithms and also contains the manual for users. At the end of the practical part we compare the results in tables and graphs.

Keywords: swarm algorithms, swarm intelligence, PSO, ARPSO, HPSO, OLPSO, C language, IEEE CEC 2015

Tímto bych chtěl poděkovat svému vedoucímu diplomové práce Ing. Michalu Pluháčkovi, Ph.D. za odbornou pomoc a cenné rady při zpracování mé diplomové práce.

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

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 HEJNOVÉ ALGORITMY	11
1.1 GLOBÁLNÍ OPTIMALIZACE	11
1.2 OBCHODNÍ CESTUJÍCÍ	12
2 PARTICLE SWARM OPTIMIZATION	13
2.1 ZÁKLADNÍ PRINCIP	13
2.2 PARAMETRY	16
2.2.1 Dimenze	16
2.2.2 Rozsah	16
2.2.3 Počet jedinců	16
2.2.4 Počet iterací	16
2.2.5 Kognitivní faktor	17
2.2.6 Sociální faktor	17
2.2.7 Maximální rychlost	17
2.2.8 Setrvačnost	18
2.3 DALŠÍ VÝVOJ	18
3 ATTRACTIVE AND REPULSIVE PARTICLE SWARM OPTIMIZATION	20
3.1 ZÁKLADNÍ PRINCIP	20
4 SELF-ADAPTIVE HETEROGENEOUS PARTICLE SWARM OPTIMIZATION	23
4.1 ZÁKLADNÍ PRINCIP	23
5 ORTHOGONAL LEARNING PARTICLE SWARM OPTIMIZATION	27
5.1 ZÁKLADNÍ PRINCIP	27
6 KNIHOVNA CEC 2015	31
7 JAZYK C	36
8 WOLFRAM MATHEMATICA	37
II PRAKTICKÁ ČÁST	38
9 IMPLEMENTACE ALGORITMŮ	39
9.1 POVINNÉ PARAMETRY ALGORITMŮ	39
9.2 VÝSTUP ALGORITMU	40
9.3 PSO.....	41
9.3.1 Nepovinné argumenty	41
9.3.2 Vnitřní struktura algoritmu	42
9.4 ARPSO	46
9.4.1 Nepovinné argumenty	46
9.4.2 Vnitřní struktura	47
9.5 HPSO.....	47
9.5.1 Nepovinné argumenty	47
9.5.2 Vnitřní struktura	49

9.6	OLPSO	49
9.6.1	Nepovinné argumenty	49
9.6.2	Vnitřní struktura	50
10	TESTOVÁNÍ	51
10.1	SOUHRNNÉ VÝSLEDKY TESTOVÁNÍ PRO DESET DIMENZÍ	52
10.2	SOUHRNNÉ VÝSLEDKY TESTOVÁNÍ PRO TŘICET DIMENZÍ.....	60
10.3	ZÁVĚR VÝSLEDKŮ TESTOVÁNÍ	70
	ZÁVĚR	71
	SEZNAM POUŽITÉ LITERATURY.....	72
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	74
	SEZNAM OBRÁZKŮ	75
	SEZNAM TABULEK.....	76
	SEZNAM PŘÍLOH.....	77

ÚVOD

Cílem této práce je implementace pokročilých variant hejnového algoritmu PSO. Hejnové algoritmy umožňují uspokojivě řešit i velmi složité optimalizační problémy v přijatelném čase. Základní varianta PSO je známa už od roku 1995 a bylo vytvořeno nespočet jejích vylepšení, která se týkala jeho různých vlastností.

V teoretické části je čtenáři představena problematika optimalizačních úloh a obecné rysy hejnových algoritmů. Je zde popsána základní varianta PSO, která bude použita pro srovnání s některými jejími modernějšími variantami. Jsou zde představeny konkrétně tři a to jsou Attractive and Repulsive Particle Swarm Optimization (dále jen ARPSO), Heterogenous Particle Swarm Optimization (dále jen HPSO) a Orthogonal Learning Particle Swarm Optimization (dále jen OLPSO). Přičemž každý je svojí vnitřní strukturou unikátní a snaží se PSO algoritmus vylepšit unikátním způsobem. U algoritmu ARPSO je zavedeno sledování rozmanitosti roje tak aby se předcházelo jeho stagnaci. Algoritmus HPSO naopak klade důraz na různé chování svých jedinců a sledování jejich úspěchů při hledání extrémů optimalizační funkce. Poslední zmíněný algoritmus OLPSO se zaměřuje na efektivnější výpočet rychlosti jedinců, a tím i jejich pozice v prohledávaném prostoru, za použití OED metody (ortogonální experimentální design).

V posledních kapitolách je uveden popis knihovny IEEE CEC 2015, jejíž funkce se použijí k otestování a vzájemnému srovnání moderních algoritmů vůči klasickému PSO. Jsou zde i kapitoly popisující použitý programovací jazyk C a prostředí Wolfram Mathematica, který je využíván pro grafické srovnání výsledků testování.

Na začátku praktické části popisujeme vnitřní strukturu vytvořených programů a popis jejich ovládání a řízení chování. Je zde i popsána struktura výstupních souborů a jejich práce s nimi v prostředí Mathematica.

V závěru praktické části jsou popsány spuštěné testy a srovnány jejich výsledky formou přehledných tabulek a také ve formě vybraných grafů některých testů. Je zde i srovnána vzájemná složitost algoritmů podle postupu v knihovně IEEE CEC 2015 a tabulka výsledků Wilcoxonova párového testu pro ověření statistické významnosti výsledků.

I. TEORETICKÁ ČÁST

1 HEJNOVÉ ALGORITMY

Jedná se o algoritmy, které se inspiřují chováním hejn, či rojů zvířecích společenstev v přírodě a napodobují její inteligenci. Proto tedy anglický název swarm intelligence (SI), česky inteligence hejna. Může se jednat například o vlčí smečky, mravenčí kolonie (algoritmus ACO) nebo o pohyb hejna ptáků (algoritmus PSO). Jsou podobné evolučním algoritmům, které svým chováním napodobují přírodu a její zákony. Spadají do nich například genetické algoritmy, které se inspiřují evoluční teorií přirozeného výběru v přírodě, kde přežívají nejsilnější jedinci a vylepšují tak genetickou informaci budoucích generací. [1]

Tyto algoritmy obecně používáme pro nalezení možných optimálních řešení různých složitých problémů, které jdou analyticky vyřešit pouze složitě a zdlouhavě anebo vůbec. Hejnové algoritmy jsou z principu schopné tyto řešení najít v přijatelně dlouhém čase. Nacházejí tudíž uplatnění při řešení složitých optimalizačních problémů jako je například skládání proteinů v chemii, leteckém průmyslu atd. [2]

Základním určujícím rysem těchto algoritmů je, že pracují s populací jedinců, kde každý jedinec reprezentuje kandidátní řešení problému. Tito jedinci se vzájemně ovlivňují podle daných pravidel v opakujících se cyklech, kterým se říká generace (evoluční algoritmy) nebo iterace (hejnové algoritmy). Cílem tohoto procesu je tedy nalezení optimálního řešení. Kvalita řešení jedinců je dána hodnotou účelové funkce (matematický zápis daného problému).

1.1 Globální optimalizace

Velkého významu tyto algoritmy nabývají při řešení problémů, na které nelze uplatnit metodu „hrubé síly“ (otestování všech možných kombinací) z důvodu velkého množství možných kombinací parametrů (patří sem například známý problém obchodního cestujícího nebo problém batohu). Nebo pro případy, kdy máme matematický popis problému (rovnici), ale tento nejde analyticky vyřešit. To může být způsobeno například multimodální (matematická funkce obsahující dvě nebo více lokálních minim či maxim) nebo nediferencovatelnou funkcí (například nespojitá funkce). Analýza problému globální optimalizace ukazuje, že neexistuje obecný deterministický algoritmus řešící obecnou globální optimalizaci v polynomiálním čase (problém globální optimalizace je NP-obtížný). [3]

1.2 Obchodní cestující

Nejznámější optimalizační problém je takzvaný problém obchodního cestujícího, který hledá nejkratší spojnice mezi městy. Množství možných spojení jednotlivých dvojic měst je v tomto případě $n!$, kde n je počet měst. Z této výpočetní složitosti jasně vyplývá, že při klasickém výpočtu možných kombinací se již při relativně malém počtu měst (>15) dostáváme k astronomickým hodnotám, které by i nejmodernější počítače zpracovávaly neúnosně dlouhou dobu. [1]

Hejnové algoritmy zde nabízejí alternativní možnost přijít s relativně kvalitním výsledkem v rozumném čase.

2 PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (česky optimalizace rojem částic, dále jen PSO) spadá do techniky umělé inteligence jako inteligence hejna, který využívá přírodních jevů při pohybu roje zvířat a sociálním chováním jeho jedinců. Tento algoritmus byl vyvinut R. Eberhartem a J. Kennedym v roce 1995. [4]

U genetických algoritmů se využívají operace křížení a mutace, kdy vznikají noví potomci na úkor rodičů, kteří zanikají a do dalších generací se dostávají obecně jen nejlepší jedinci. U PSO nevznikají ani nezanikají v průběhu generací žádní jedinci, ale pohybují se v prostoru řešení daného problému. Každý jedinec má definovanou svoji pozici, rychlost a pamatuje si pozici svého doposud nalezeného nejlepšího řešení (označováno jako pBest; personal best). Dále má každý jedinec přístup ke globálnímu nejlepšímu řešení (gBest; global best). Tyto proměnné udávají směr pohybu a tudíž i novou polohu každé částice v další iteraci. PSO obecně končí po iteraci daného počtu generací předem stanovených uživatelem anebo při dosažení dlouhodobé stagnace. [1,2]

Díky jednoduchým sociálním technikám (pBest a gBest) je dosaženo překvapivě inteligentního chování, kdy se hejno (roj) částic chová jako celek a efektivně se mu daří prohledávat zadanou oblast problému.

2.1 Základní princip

Základní jednotkou PSO je jedinec, který je reprezentován jako souřadnice v n-dimenzionálním prostoru představující hledané parametry účelové funkce pro zadaný optimalizační problém. Algoritmus v každé iteraci aktualizuje polohu každé částice. Tato nová poloha se ověří, zda se nachází v nastavených mezích hledaného problému, ohodnotí se a poté se porovná s pBest a gBest, zda došlo ke zlepšení výsledku (zmenšení účelové funkce). [2,4]

Tento základní princip je znázorněn na obrázku 1.

Nová poloha jedince se počítá z aktuální polohy a vektoru rychlosti, který je pro každého jedince jedinečný a počítá se podle vztahů:

$$v_{i,d}(t+1) = v_{i,d}(t) \cdot w + c_1 \cdot rand \cdot (pBest_{i,d} - x_{i,d}(t)) + c_2 \cdot rand \cdot (gBest_{i,d} - x_{i,d}(t)) \quad (2.1)$$

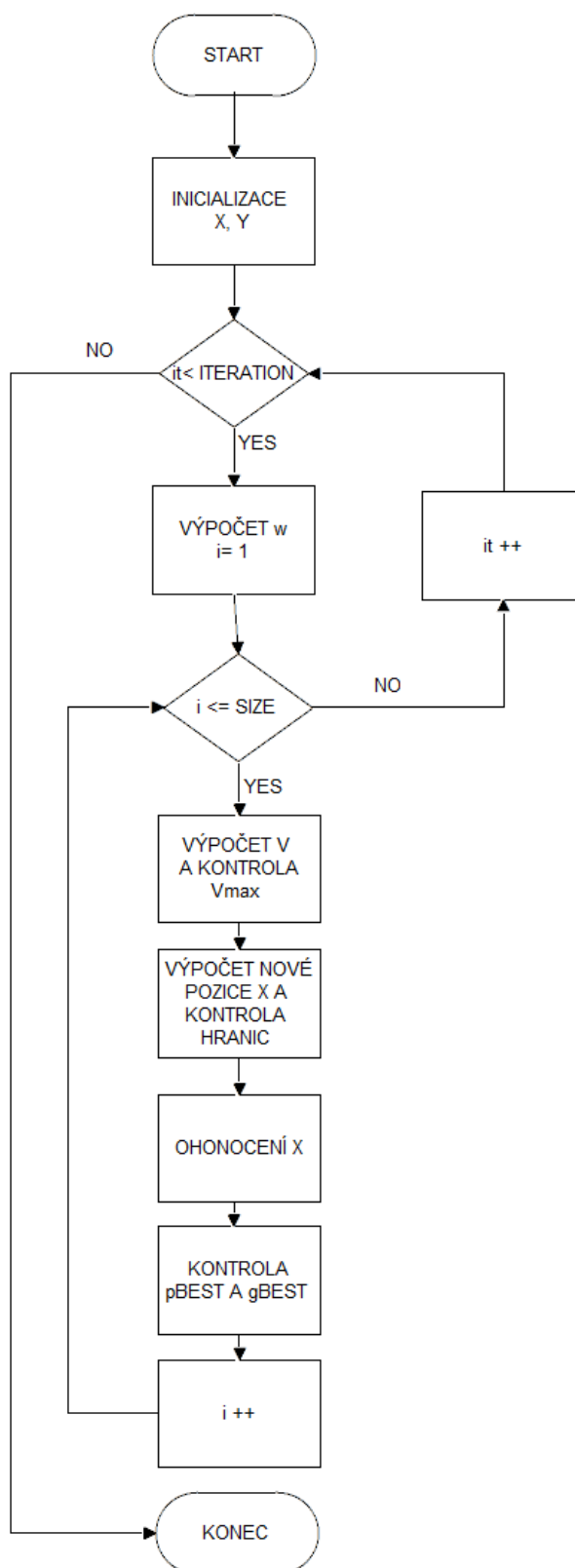
$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1) \quad (2.2)$$

kde	$v_{i,d}(t+1)$: rychlost jedince v následující iteraci,
	$v_{i,d}(t)$: rychlost jedince v aktuální iteraci,
	w	: setrvačnost (popsána v kapitole 2.2.8)
	$x_{i,d}(t+1)$: pozice jedince v následující iteraci,
	$x_{i,d}(t)$: pozice jedince v aktuální iteraci,
	t	: aktuální iterace,
	$pBest_{i,d}$: nejlepší pozice jedince,
	$gBest_d$: nejlepší nalezená pozice v celé populaci,
	$rand$: náhodné číslo rovnoměrného rozložení v intervalu $\langle 0,1 \rangle$,
	c_1, c_2	: kognitivní a sociální faktor, učící faktory.

Ohodnocení jedince se provádí dosazením jeho polohy, parametrů, do účelové funkce. Tato výsledná hodnota se srovnává s doposud nejlepší hodnotou jedince $pBest$ a nejlepší hodnotou v celé populaci $gBest$. Pokud je nová poloha jedince lepší tak se jeho parametry uloží jako nová pozice $pBest$ popřípadě $gBest$. [2,4]

Z tohoto popisu je patrné, že jedinci se pohybují, migrují, po prohledávaném prostoru a nedochází k jejich zániku nebo novému vzniku jedinců jako je tomu u různých genetických algoritmů. Výsledná nejlepší hodnota, kombinace parametrů, je uložena v $gBest$. [2,4]

Základní verze PSO algoritmu trpí několika „neduhy“, které často znehodnocují prohledávání oblasti. Na možná zlepšení PSO algoritmu se podíváme v dalších kapitolách.



Obrázek 1: Diagram průběhu algoritmu PSO

2.2 Parametry

Základní verze algoritmu obsahuje několik různých parametrů. Některé jsou dány řešeným problémem jako je velikost dimenze a některé ovlivňují chování částic a jejich různým nastavením se dá dosáhnout různě kvalitních řešení. Různé publikace a autoři nazývají někdy parametry PSO různými názvy a proto zde souhrnně popíšeme parametry základního PSO algoritmu s jejich názvy, které zde budeme nadále používat.

2.2.1 Dimenze

Počet parametrů účelové funkce a tedy i počet parametrů jedince určuje velikost dimenze. Její velikost je dána typem zadané optimalizační úlohy. Velikost dimenze značně ovlivňuje výkon algoritmu, zvyšuje se složitost. Vyšší dimenze znamená větší rozměr jedince, vektor rychlosti jedince, pBest a gBest. Označujeme jako *dim* nebo *d*.

2.2.2 Rozsah

Jedná se o hranice oblasti, kterou jedinci prohledávají. Stanovují tedy číselný rozsah jednotlivých parametrů. Každý parametr přitom může mít rozdílný rozsah, a při řešení reálných úloh je to i běžný jev. Rozsah podobně jako dimenzi je dán zadanou optimalizační úlohou.

2.2.3 Počet jedinců

Určuje počet jedinců, kteří budou prohledávat vymezený prostor. S rostoucím počtem jedinců dosahujeme kvalitnějších řešení, ale narůstá také výpočetní čas, kvůli nutnosti ohodnocení každého jedince v nové iteraci. Označujeme jako *NP*. V literatuře se doporučuje velikost populace $NP = 40\sim 60$. Což pro většinu optimalizačních problémů vystačuje. [1]

2.2.4 Počet iterací

V každé iteraci vypočítáváme novou pozici jedinců. Počet jednotlivých iterací tedy označuje maximální počet těchto výpočtů, obdobně jako počet generací u genetických algoritmů. Tento počet je většinou dostačující ukončovací podmínkou algoritmu. S rostoucím počtem iterací roste přesnost výsledku, do určité míry, kdy poté další navyšování již výsledek nemůže více zpřesnit (dosažení globálního minima nebo uváznutí v lokálním minimu).

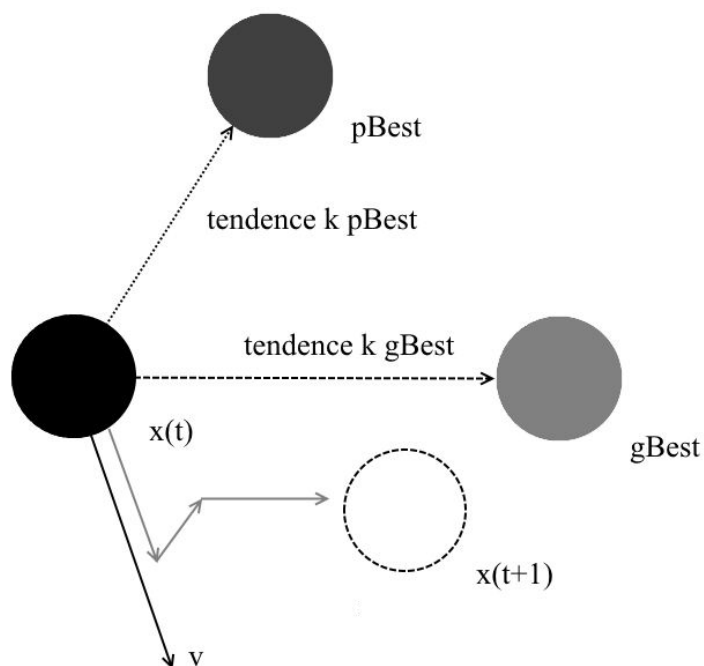
2.2.5 Kognitivní faktor

Jedná se o jeden ze dvou takzvaných učících faktorů, který ovlivňuje význam pBest při výpočtu nové rychlosti jedince. Označujeme jako c_1 a nabývá většinou hodnoty 2. Pro různé problémy a typy úloh se můžou používat i jiné nastavení tohoto faktoru. [1]

2.2.6 Sociální faktor

Podobný jako kognitivní faktor, ale ovlivňuje význam složky gBest a označujeme jej c_2 a jeho hodnota bývá 2. [1]

Rozdílné hodnoty obou faktorů pak určují, které složce dáváme větší přednost. Při vyšší hodnotě c_1 se jedinec pohybuje více směrem k jeho hodnotě pBest, naopak při vyšší hodnotě c_2 se pohybuje ke globálnímu optimu gBest (jak je vidět na obrázku 2). Nicméně díky součinu obou faktorů s náhodným číslem se můžou tyto přednosti měnit v každé iteraci.



Obrázek 2: Pohyb jedince ovlivněný učícími faktory [1]

2.2.7 Maximální rychlost

Označujeme jej jako V_{max} a udává maximální rychlost jedince. Obdobně jako rozsah může být pro každý parametr dimenze jiný, tak i maximální rychlost může mít pro každou dimenzi jinou velikost. Tento parametr PSO byl vytvořen z důvodu zabránění prudké akceleraci jedinců a následné degradaci PSO algoritmu na náhodné prohledávání

vymezeného prostoru. Každá složka rychlosti jedince je porovnávána s touto maximální rychlostí a při jejím případném překročení se nastaví na tuto maximální hodnotu. Obvykle se volí jako 20% rozsahu. [1]

2.2.8 Setrvačnost

Tento parametr stanovuje, jakou měrou se aktuální rychlost jedince promítne ve výpočtu nové rychlosti v další iteraci. Označujeme jej jako w . Udává se jako konstantní číslo nebo jako proměnná závislá na čase, neboli počtu iterací, kdy se lineárně zmenšuje dle vztahu:

$$w(t + 1) = (w_{end} - w_{start}) \cdot \frac{t}{t_{max}} + w_{start} \quad (2.3)$$

kde $w(t+1)$: setrvačnost v následující iteraci,

w_{start} : počáteční velikost setrvačnosti,

w_{end} : koncová hodnota setrvačnosti,

t : pořadí aktuální iterace,

t_{max} : maximální velikost t neboli ukončovací počet iterací algoritmu.

V tomto případě se nejčastěji volí $w_{start} = 0.9$ a $w_{end} = 0.4$. Tato použití umožňují algoritmu v počátečních fázích zmapovat větší prostor a v pozdějších fázích se zaměřit na podrobnější průzkum zajímavých oblastí. [4]

2.3 Další vývoj

U tohoto základního PSO algoritmu dochází často k různým negativním jevům ovlivňujícím kvalitu nalezeného řešení.

K těmto jevům patří například předčasná konvergence roje (jedinci uváznou v lokálním minimu funkce a diverzita jedinců se zmenšuje nebo stagnuje), oscilační fenomén (vzniká kmitáním jedince mezi pBest a gBest) anebo jev zvaný „dva kroky vpřed, jeden zpátky“, který nastává u tradičního učení algoritmu, kdy změna jednoho parametru jedince sice vede ke zlepšení hodnoty účelové funkce v dané dimenzi, ale změny ostatních parametrů vedou ke zhoršení. Tyto jevy se snažíme minimalizovat, nebo jim úplně zabránit. Pro eliminaci bylo již vytvořeno několik modifikací základního PSO algoritmu. [5]

V této práci se budeme zabývat třemi odlišnými modifikacemi, které různými způsoby ovlivňují chování jedinců roje. První z nich se jmenuje ARPSO (Attractive and repulsive PSO), druhý se nazývá Self-adaptive Heterogeneous PSO (self-adaptive HPSO). Třetí a poslední z nich je OLPSO (Orthogonal Learning PSO).

3 ATTRACTIVE AND REPULSIVE PARTICLE SWARM OPTIMIZATION

Ve zkratce ARPSO, česky přitahující a odpuzující PSO, byl představen v roce 2002 J. Rigetem a J. S. Vesterstrømem. Tato změna byla vytvořena z důvodu předčasné konvergence PSO na silně multimodálních problémech. Tento jev vzniká zmenšením diverzity v prohledávaném prostoru, který vede k totální stagnaci hodnoty účelové funkce roje. [6]

3.1 Základní princip

Abychom zabránili této předčasné konvergenci částic, byl zaveden mechanismus pro kontrolu rozmanitosti jedinců a přibyla úprava výpočtu nové rychlosti podle vztahu:

$$v_d(t+1) = v_d(t) \cdot w + dir \cdot (c_1 \cdot rand \cdot (pBest_{i,d} - x_{i,d}(t)) + c_2 \cdot rand \cdot (gBest_{i,d} - x_{i,d}(t))) \quad (3.1)$$

kde dir označuje směr pohybu jedinců a může nabývat pouze hodnot 1 (normální chování jedinců, přitažlivá síla) nebo -1 (odpuzující směr pohybu jedinců).

Hodnota dir se určuje podle:

$$dir = \begin{cases} 1, & \text{pokud rozmanitost} > dHigh \\ -1, & \text{pokud rozmanitost} < dLow \end{cases} \quad (3.2)$$

kde $dHigh$: horní hranice rozmanitosti a je stanovena na hodnotu 0.25,

$dLow$: dolní hranice rozmanitosti a její hodnota je $5 \cdot 10^{-6}$.

Výpočet rozmanitosti je potom dán podle vztahu:

$$\text{rozmanitost} = \frac{1}{NP \cdot |L|} \cdot \sum_{i=1}^{NP} \sqrt{\sum_{d=1}^{\text{dim}} (x_{i,d} - \bar{x}_d)^2} \quad (3.3)$$

kde NP : počet jedinců,

|L| : délka nejdelší diagonály v prohledávaném prostoru,

dim : počet dimenzí problému,

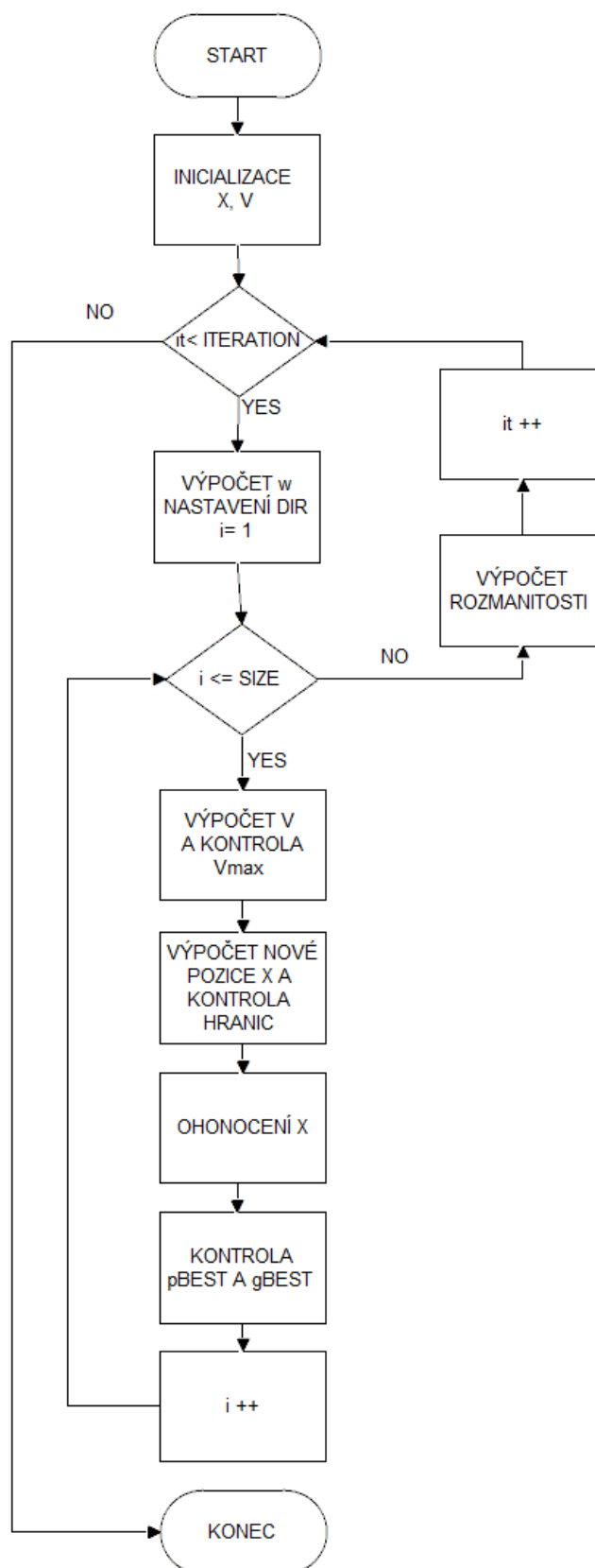
$x_{i,d}$: pozice částice,

\bar{x}_d : průměrná pozice částice v dimenzi.

Algoritmus poté v principu funguje v těchto krocích:

1. Inicializace
2. Nastavení směru
3. Změna rychlosti jedince
4. Změna pozice jedince
5. Výpočet účelové funkce
6. Výpočet rozmanitosti
7. Opakování algoritmu od kroku 2

Grafické znázornění principu tohoto algoritmu je vidět na obrázku 3.



Obrázek 3: Diagram průběhu algoritmu ARPSO

4 SELF-ADAPTIVE HETEROGENEOUS PARTICLE SWARM OPTIMIZATION

Tento algoritmus byl uveden v roce 2013 F. V. Nepomuceno a A. P. Engelbrechtem. Umožňuje částicím měnit jejich chování v průběhu hledání. PSO, které používá stále stejné vztahy pro výpočet pozice a rychlosti jedinců v roji se nazývá homogenní, to znamená, že částice prohledávají prostor se stále stejným chováním. Nicméně toto chování nemusí být pokaždé optimální pro prohledávání celého prostoru zadaného problému. Varianta PSO algoritmu s možností změny svého chování se nazývá heterogenní (heterogenous, HPSO). [7]

Existují tři typy heterogenních PSO:

- **Static HPSO** – každá částice si na začátku algoritmu vybere náhodně určité chování a tímto se pak po celou dobu řídí až do jeho skončení.
- **Dynamic HPSO** – částice můžou své chování změnit v průběhu algoritmu, toto chování se vybírá buď deterministicky, nebo stochasticky.
- **Self-adaptive HPSO** – částice si vybírají další chování na základě úspěšnosti nebo neúspěšnosti chování.

4.1 Základní princip

Možnosti chování roje jsou uloženy v takzvaném behavior poolu, který by se dal přeložit jako sbírka možností chování. Na začátku algoritmu se každé částici přiřadí náhodné chování z behavior poolu a vynuluje se počítadlo stagnace každé částice (počet iterací po které dané chování nezlepší pBest dané částice). Pokud některá částice dosáhne předem stanovené hranice stagnace, tak se pro ni vybere nové chování podle turnajové selekce (anglicky tournament selection). Ta funguje tak, že náhodně vybere z behavior poolu daný počet chování a z nich se následně zvolí tu, jež po k -iteracích přinesla nejvíc zlepšení pBest částic, které toto chování používají. Tento typ výběru nového chování se používá z důvodu zabránění některého chování v dominanci a umožňuje i výběr ostatních chování, které se projevují zlepšováním hodnoty účelové funkce. Velikost náhodně vybraných chování do turnajové selekce ovlivňuje pravděpodobnost výběru chování, které málo nebo vůbec nezlepšují účelovou funkci. Při výběru jednoho chování se jedná čistě o náhodný výběr a při velikosti rovnající se velikosti behavior poolu se jedná o elitismus, kdy se vždy bude

upřednostňovat jedno chování. Vyšší hodnoty zabraňují tedy výběru nevhodných chování, což někdy znemožňuje vybrat nové chování, pokud je potřeba. Malé hodnoty naopak znemožňují prozkoumat více chování, které si vedou dobře. [7]

Velikost k -iterací, počet předchozích iterací, pro které se počítá úspěšnost chování (vylepšení pBest), má vliv na obměnu strategií. Například, aby se včas změnilo chování částice, pokud se dostane do oblasti, kde již stávající strategie nevyhovuje. Vyšší hodnoty vyústí v častý výběr sub-optimálních strategií, i když v posledních několika iteracích nezlepšují výsledek. Naopak menší hodnoty zabraňují chováním, aby nabraly určitý moment setrvačnosti, a častěji se pak vybírají i ostatní typy. [7]

Maximální hodnota stagnace určuje počet iterací, po které musí mít chování vliv na prohledávání. Vyšší hodnoty umožňují strategiím se dostat do správných oblastí, než začnou vylepšovat kvalitu výsledku a nižší hodnoty zase včasnou změnu chování, pokud je potřeba. [7]

Stručný grafický přehled průběhu algoritmu je na obrázku 4.

Behavior pool obsahuje celkem šest strategií, kdy každá disponuje určitou unikátní vlastností:

- **TVAC-PSO**, pro brzký průzkum velké oblasti a pozdější detailní prohledávání za použití lineární změny učicích faktorů (c_1 se lineárně zmenšuje a hodnota c_2 lineárně vzrůstá),

$$c_1(t+1) = (c_{1,end} - c_{1,start}) \cdot \frac{t}{t_{max}} + c_{1,start} \quad (4.1)$$

$$c_2(t+1) = (c_{2,end} - c_{2,start}) \cdot \frac{t}{t_{max}} + c_{2,start} \quad (4.2)$$

- **TVIW-PSO**, pro brzký průzkum velké oblasti a pozdější detailní prohledávání za využití lineární změny zrychlení (w) a konstantní hodnotě učicích faktorů (tzn. výpočet nové pozice jedince je stejná jako v klasickém PSO, viz rovnice (2.1), (2.2) a (2.3),

- **sPSO**, pro vysokou schopnost detailního prohledávání oblastí gBest,

$$v_d(t+1) = v_d(t) \cdot w + c_2 \cdot rand \cdot (gBest_{i,d} - x_{i,d}(t)) \quad (4.3)$$

- **cPSO**, z důvodu využití kognitivního chování,

$$v_d(t+1) = v_d(t) \cdot w + c_1 \cdot rand \cdot (pBest_{i,d} - x_{i,d}(t)) \quad (4.4)$$

- **modBB-PSO**, jeho schopností je jak dostat se z lokálních minim, tak i možnost prohledat okolí pBest, kdy lineárně roste jeho pravděpodobnost (ep) z 0 na 1, že bude prohledávat ke konci algoritmu více právě pBest,

$$x_{i,d}(t+1) = \begin{cases} pBest_{i,d}, & \text{pokud } U(0,1) < ep \\ N\left(\frac{pBest_{i,d} + gBest_d}{2}, |pBest_{i,d} - gBest_d|\right), & \text{jinak} \end{cases} \quad (4.5)$$

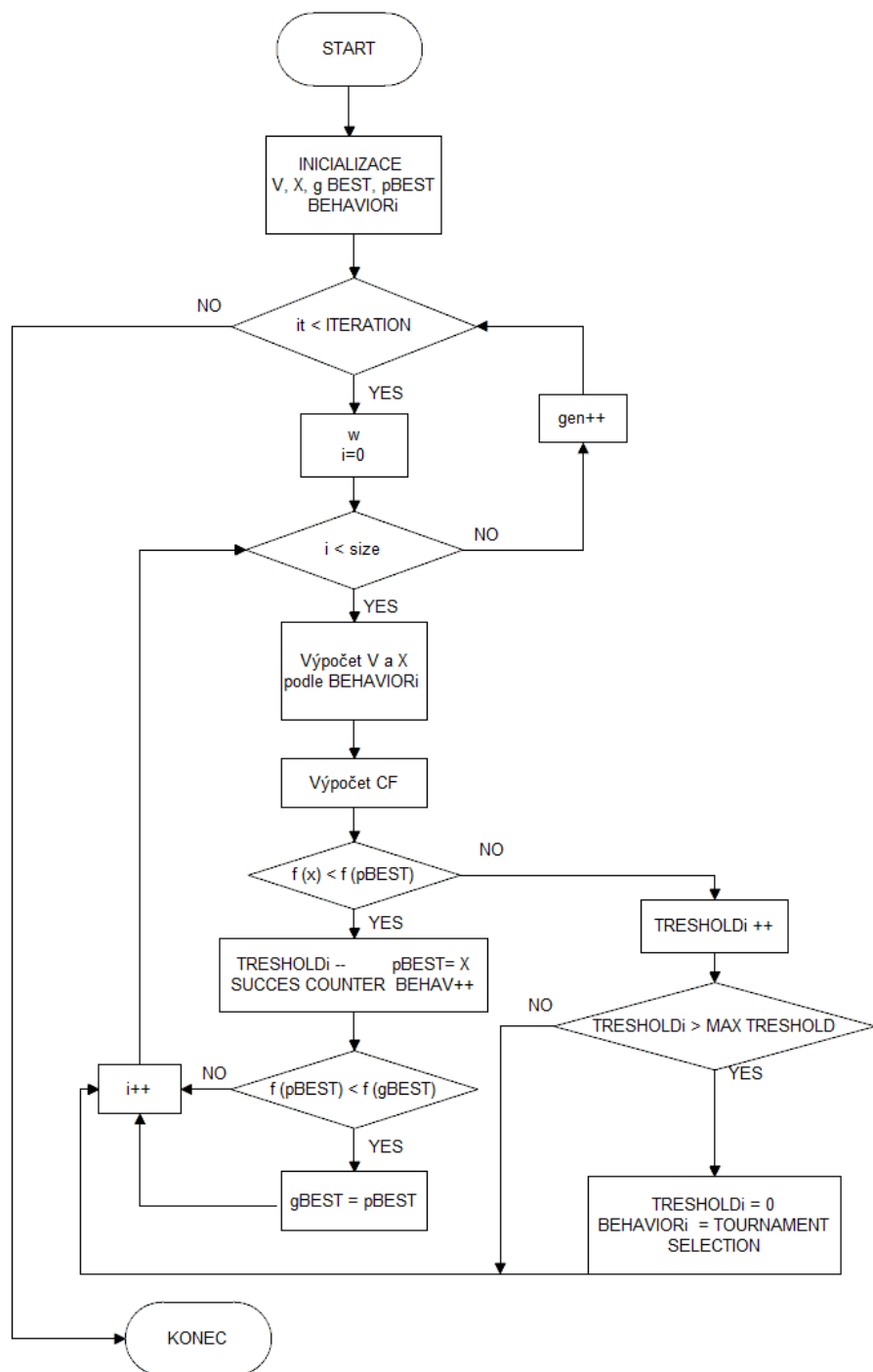
kde N : normální (Gaussovo) rozdělení,

U : rovnoměrné rozdělení,

- **QSO**, u něho se rádius cloudu lineárně zmenšuje z maximálního možného prohledávaného okolí (horní a dolní hranice každé dimenze) k nule. Tím se umožňuje chování na začátku „vyskočit“ z lokálních optim a ke konci se zaměřit na gBest.

$$x_{i,d}(t+1) \sim N(gBest_d, \sigma) \quad (4.6)$$

kde σ : rádius cloudu.



Obrázek 4: Diagram průběhu algoritmu HPSO

5 ORTHOGONAL LEARNING PARTICLE SWARM OPTIMIZATION

Ve zkratce OLPSO, česky přeloženo jako PSO algoritmus s ortogonálním učením. Tento algoritmus byl představen v roce 2010 pány Zhi-Hui Zhan, Jun Zhang, Yun Li a Yu-Hui Shi. [5]

Principem tohoto algoritmu je vytvoření takzvaného guidance vektoru, označovaného jako \mathbf{P}_o . Ten je konstruován z vektorů $pBest$ a $gBest$ pomocí OED metody (Orthogonal Experimental Design) na předdefinované tabulce, která se jmenuje orthogonal array (OA, česky ortogonální pole). Principem OED metody je vybrat pro každou dimenzi hodnotu, která nejvíce pomáhá zlepšit hodnotu účelové funkce. OA se generuje podle pravidla $L_M(2^D)$, kde D je počet dimenzí problému a určuje počet sloupců tabulky, a M určuje počet řádků OA tabulky. [5]

5.1 Základní princip

Postup pro vytvoření guidance vektoru je následující:

- Vytvoření dvou-úrovňové tabulky s počtem řádků M , počtem sloupců N a základním číslem sloupce u .

$$M = 2^{\lceil \log_2(dim+1) \rceil} \quad (5.1)$$

$$N = M - 1 \quad (5.2)$$

$$u = \log_2 M \quad (5.3)$$

- Elementy tabulky vyplníme podle následujícího pravidla

$$L[a][b] = \left(\left\lfloor \frac{a-1}{2^{u-k}} \right\rfloor \right) \bmod 2 \quad (5.4)$$

Kde $a = 1, 2, \dots, M$ určuje řádek tabulky, $b = 2^{k-1}$ je index základního sloupce a $k = 2, \dots, u$.

- Následně prvky ostatních sloupců jsou nastaveny podle

$$L[a][b + s] = (L[a][s] + L[a][b]) \bmod 2 \quad (5.5)$$

Kde $a = 1, 2, \dots, M$ opět určuje řádek tabulky, $b = 2^{k-1}$ je index sloupce, $s = 1, 2, \dots, b-1$ a $k = 2, \dots, u$.

- Pro všechny elementy v tabulce použijeme pravidlo

$$L[a][b] = \begin{cases} 1, & \text{pokud } L[a][b] = 0 \\ 2, & \text{pokud } L[a][b] = 1 \end{cases} \quad (5.6)$$

Zde $a = 1, 2, \dots, M$ značí řádek tabulky a $b = 1, 2, \dots, N$ je sloupec tabulky.

Například pro problém o 2 dimenzích bude výsledná tabulka následující:

$$L_4(2^3) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \quad (5.7)$$

Jelikož máme jen dvoudimenzionální problém tak použijeme pouze prvních d -dimenzí matice, tedy:

$$L_4(2^2) = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 1 \\ 2 & 2 \end{bmatrix} \quad (5.8)$$

- Dalším krokem je vytvoření M testovacích řešení, kdy pro každou dimenzi testovacího řešení použijeme hodnotu buď $pBest$ nebo $gBest$ příslušné dimenze (pokud je index v tabulce 1 použijeme hodnotu $pBest$ a pokud 2 pak $gBest$).
- Vyhodnotíme každé řešení a nejlepší z nich (nejlepší hodnota účelové funkce) uložíme jako vektor \mathbf{X}_b .
- Vypočítáme přínos hodnoty $gBest$ nebo $pBest$ v každé dimenzi pomocí

$$S_{nq} = \frac{\sum_{m=1}^M (f_m \cdot z_{mnq})}{\sum_{m=1}^M z_{mnq}} \quad (5.9)$$

Kde f_m je výsledek M testovacího řešení, n určuje sloupec tabulky a q určuje úroveň (v našem případě 1 nebo 2, čili pBest nebo gBest). Pro každou úroveň (pBest nebo gBest) v každé dimenzi tedy dostaneme aritmetický průměr testů, které byly ovlivněny danou hodnotou. Na základě těchto hodnot vybereme do vektoru \mathbf{X}_p hodnoty s nejlepším ohodnocením (nejlepší hodnotou účelové funkce) v dané dimenzi.

- Jako poslední krok porovnáme účelové funkce dvojice vektorů \mathbf{X}_b a \mathbf{X}_p . Nejlepší z nich se vybere a uloží jako guidance vektor \mathbf{P}_o .

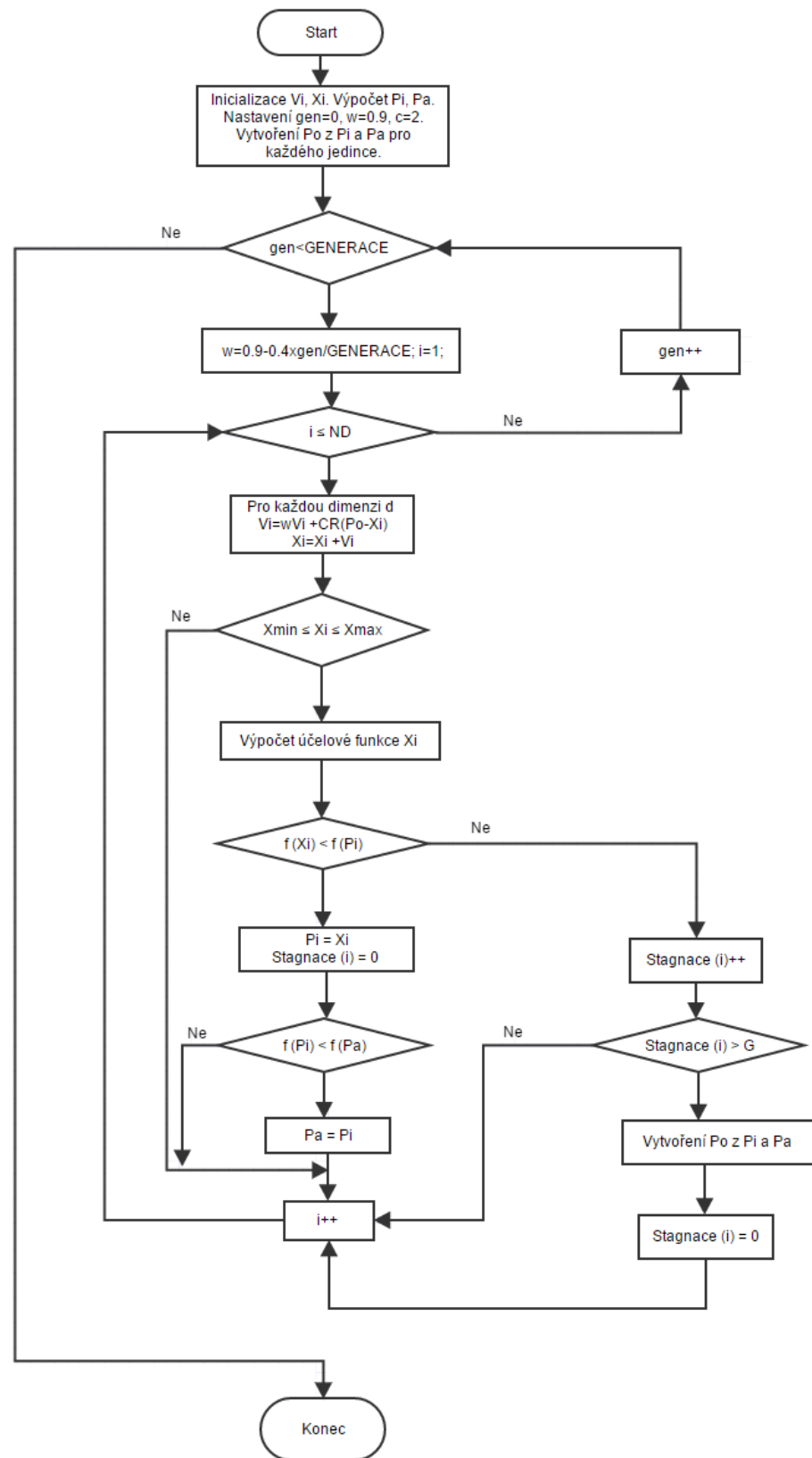
Rychlost částice se poté počítá v každé iteraci podle následujícího vztahu:

$$v_{i,d}(t+1) = v_{i,d}(t) \cdot w + c \cdot rand \cdot (P_{o,i,d} - x_{i,d}) \quad (5.10)$$

kde P_o : guidance vektor.

Tento postup vytváření vektoru je časově zdlouhavý a výpočetně náročný, proto se jeho rekonstrukce odehrává pouze v případě, že hodnota pBest stagnuje po určitý počet předem stanovených iterací algoritmu. [5]

Základní princip algoritmu je na obrázku 5.



Obrázek 5: Diagram průběhu algoritmu OLPSO

6 KNIHOVNA CEC 2015

Pro testování nových verzí PSO je využívána testovací knihovna IEEE CEC 2015, která obsahuje 15 různých testovacích funkcí. Jedná se o směs unimodálních, multimodálních, hybridních a složených funkcí. Každá funkce má stanovený rozsah $[-100,100]^{\text{Dim}}$. Přehled těchto funkcí je vypsán v tabulce 1. [8]

Číslo testovací funkce	Typ funkce	Jméno	Minimum
1	Unimodální	Rotated High Conditioned Elliptic Function	100
2		Rotated Cigar Function	200
3	Multimodální	Shifted and Rotated Ackley's Function	300
4		Shifted and Rotated Rastrigin's Function	400
5		Shifted and Rotated Schwefel's Function	500
6	Hybridní	Hybrid Function 1	600
7		Hybrid Function 2	700
8		Hybrid Function 3	800
9	Složené	Composition Function 1	900
10		Composition Function 2	1000
11		Composition Function 3	1100
12		Composition Function 4	1200
13		Composition Function 5	1300
14		Composition Function 6	1400
15		Composition Function 7	1500

Tabulka 1: Přehled testovacích funkcí v IEEE CEC 2015 [8]

Každá tato funkce je tvořena jednou nebo několika základními funkcemi podle dané definice. Základní funkce jsou v tabulce 2.

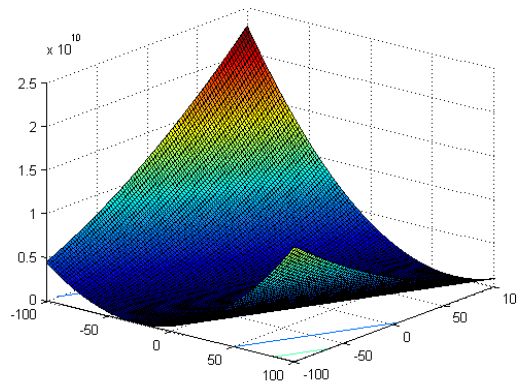
f	Název	Rovnice
1	High Conditioned Elliptic Function	$f_1(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2$
2	Cigar Function	$f_2(x) = x_1^2 + 10^6 \sum_{i=2}^D x_i^2$
3	Discus Function	$f_3(x) = 10^6 x_1^2 + \sum_{i=2}^D x_i^2$
4	Rosenbrock's Function	$f_4(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$
5	Ackley's Function	$f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e$
6	Weierstrass Function	$f_6(x) = \sum_{i=1}^D \left(\sum_{k=0}^{kmax} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{kmax} [a^k \cos(2\pi b^k \cdot 0.5)]$ $a = 0.5, b = 3, kmax = 20$
7	Griewank's Function	$f_7(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
8	Rastrigin's Function	$f_8(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$
9	Modified Schwefel's Function	$f_9(x) = 418.9829 \times D - \sum_{i=1}^D g(z_i), \quad z_i = x_i + 4.209687462275036e + 002$ $g(z_i) = \begin{cases} z_i \sin(z_i ^{1/2}) & \text{if } z_i \leq 500 \\ (500 - \text{mod}(z_i, 500)) \sin(\sqrt{ 500 - \text{mod}(z_i, 500) }) - \frac{(z_i - 500)^2}{10000D} & \text{if } z_i > 500 \\ (\text{mod}(z_i , 500) - 500) \sin(\sqrt{ \text{mod}(z_i , 500) - 500 }) - \frac{(z_i + 500)^2}{10000D} & \text{if } z_i < -500 \end{cases}$
10	Katsuura Function	$f_{10}(x) = \frac{10}{D^2} \prod_{i=1}^D \left(1 + i \sum_{j=1}^{32} \frac{ 2^j x_i - \text{round}(2^j x_i) }{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^2}$
11	HappyCat Function	$f_{11}(x) = \left \sum_{i=1}^D x_i^2 - D \right ^{1/4} + (0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i) / D + 0.5$

12	HGBatFunction	$f_{12}(x) = \left \left(\sum_{i=1}^D x_i^2 \right)^2 - \left(\sum_{i=1}^D x_i \right)^2 \right ^{1/2} + \frac{0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i}{D} + 0.5$
13	Expanded Griewank's plus Rosenbrock's Function	$f_{13}(x) = f_7(f_4(x_1, x_2)) + f_7(f_4(x_2, x_3)) + \dots + f_7(f_4(x_{D-1}, x_D)) + f_7(f_4(x_D, x_1))$
14	Expanded Scaffer's F6 Function	Scaffer's F6 Function: $g(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2 + y^2}) - 0.5)}{(1 + 0.001(x^2 + y^2))^2}$ $f_{14}(x) = g(x_1, x_2) + g(x_2, x_3) + \dots + g(x_{D-1}, x_D) + g(x_D, x_1)$

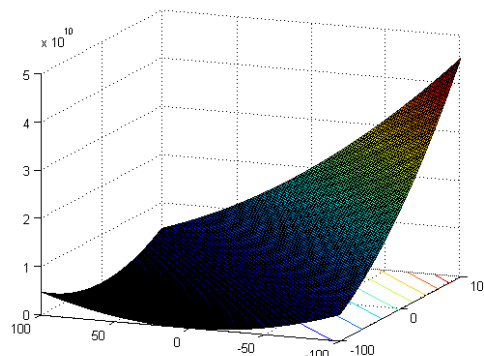
Tabulka 2: Přehled základních funkcí [8]

Přehled typů funkcí použitých k testování:

- **unimodální funkce**, obsahují pouze jeden extrém a jsou zde zastoupeny funkce jako Rotated High Conditioned Elliptic Function (Obrázek 6) a Rotated Cigar Function (Obrázek 7),

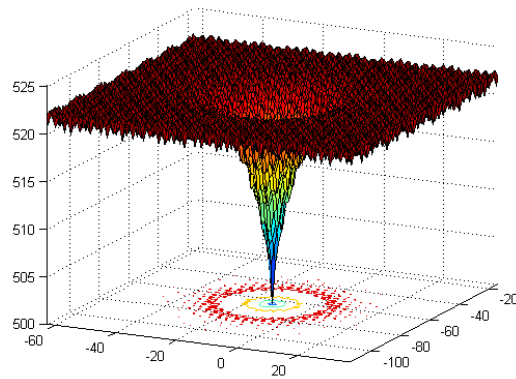


Obrázek 6: Rotated High Conditioned Elliptic Function [8]

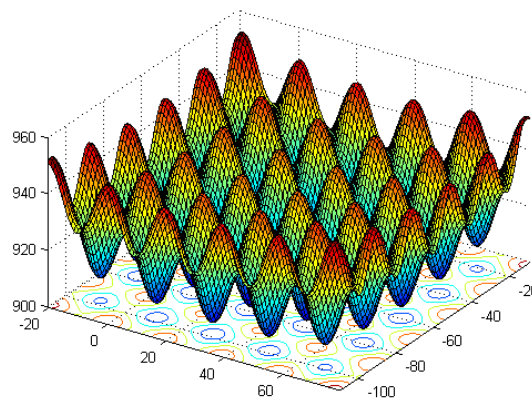


Obrázek 7: Rotated Cigar Function [8]

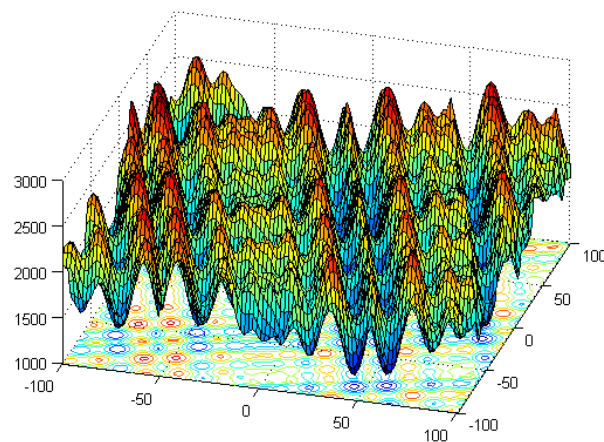
- **multimodální funkce**, obsahují několik extrémů a patří sem Shifted and Rotated Ackley's Function (Obrázek 8), Shifted and Rotated Rastrigin's Function (Obrázek 9) a Shifted and Rotated Schwefel's Function (Obrázek 10),



Obrázek 8: Shifted and Rotated Ackley's Function [8]



Obrázek 9: Shifted and Rotated Rastrigin's Function [8]



Obrázek 10: Shifted and Rotated Schwefel's Function [8]

- **hybridní funkce**, skládají se vždy z několika základních funkcí, které mají různý podíl na celkové výsledné funkci,
- **složené funkce**, mají vlastnost, kdy se proměnná každé dimenze chová podle jiné základní nebo hybridní funkce.

7 JAZYK C

Jedná se o univerzální kompilovaný programovací jazyk nízké úrovně. Má proto i velmi úsporné vyjadřování a relativně malý soubor operátorů a datových struktur. Díky jeho nízké úrovni je mnohem rychlejší a efektivnější než většina ostatních jazyků, hlavně vyšších jako je například JAVA, Python nebo C#. Program v C se rychlostí zpracování kódu téměř vyrovná programu v jazyce assembleru. [9]

První standard tohoto jazyka byl popsán v knize „The C Programming Language“ v roce 1978 od pánů B. W. Kernighana a D. M. Ritchie. Novějším standardem je ANSI C z roku 1988, který přidává přesné specifikace množiny knihovných funkcí a hlavičkových souborů. [9]

Ukládání dat je možné provádět třemi způsoby. Statickou alokací paměti, automatickou alokací na zásobníku nebo dynamickou alokací na haldě pomocí knihoven. S pamětí se pracuje nejčastěji pomocí datového typu ukazatel, jenž obsahuje adresu na začátek paměťového prostoru daného typu proměnné. Jak jsou ukazatele mocným nástrojem pro rychlou práci v polích, tak jsou i jeho slabinou při nesprávném použití a vedou často ke zhroucení programu. Programátor má plnou zodpovědnost za alokaci a dealokaci paměti, což někdy vede k únikům paměti, kdy se paměť už nepoužívá, ale v systému zůstává registrovaná jako obsazená. [9]

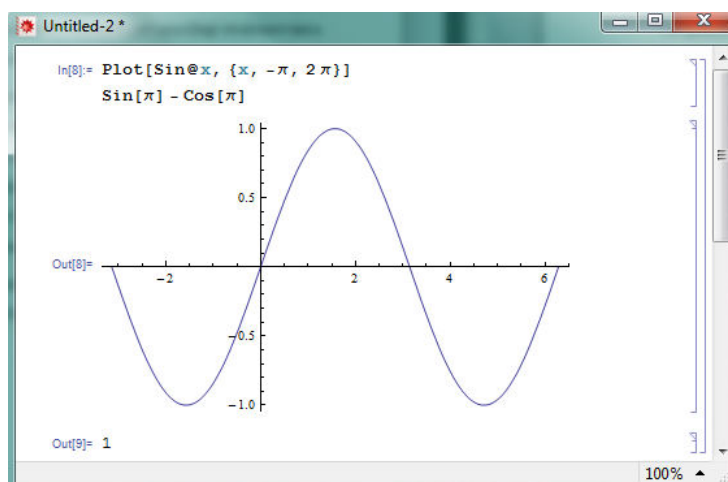
8 WOLFRAM MATHEMATICA

Jde o široce využívaný počítačový program ve vědeckých, matematických a technických oborech. Roku 1988 jeho první verzi uvedl S. Wolfram. Další verze už tvořil společně s týmem matematiků a programátorů. Nejnovější verze je 10.4 vydaná v roce 2016. Mathematica je rozdělena na dva logické celky. Jádro, které interpretuje výrazy a vrací jejich výsledky, a GUI, které tyto výsledky vhodným způsobem zobrazuje. [10]

GUI umožňuje vytvářet dokumenty, které se jmenují notebook a obsahují programový kód, formátovaný text, grafiku, tabulky, zvuky a různé ovládací prvky. Veškeré prvky mohou být generovány pomocí algoritmu nebo interaktivně editovány pomocí dostupných grafických nástrojů. Dokument je strukturovaný do hierarchických buněk, jež umožňují strukturování a oddělení například algoritmu od grafického výstupu. [10, 11]

Pomocí tohoto programu se dá i programovat. Tento multiparadigmatický programovací jazyk se nazývá Wolfram Language. Díky tomu Mathematica zvládá obrovskou škálu úkolů, jako jsou matematické operace, práce s maticemi, práce s komplexními čísly, zobrazení 2D a 3D dat a práci s geometrií, geo vizualizace a animace. Dále umí řešit klasické i diskrétní rovnice. Obsahuje množství různých statistických nástrojů. Přes Mathematicu lze spouštět další programy anebo číst dokumenty na disku a s těmito daty dále pracovat. [11]

Mimo jiné je součástí tohoto softwaru i interaktivní nápověda obsahující příklady funkcí v něm obsažených a dále jednoduché tutoriály pro jejich používání a strukturování dokumentu. [11]



Obrázek 11: Ukázka notebooku v prostředí Wolfram Mathematica

II. PRAKTICKÁ ČÁST

9 IMPLEMENTACE ALGORITMŮ

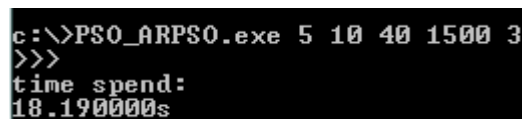
Každý z prezentovaných algoritmů (PSO, ARPSO, HPSO a OLPSO) je tvořen samostatným spustitelným exe souborem. Podmínkou pro spuštění programu je přítomnost složky „input_data“, která obsahuje soubory pro testovací knihovnu IEEE CEC 2015. Výstupní soubor má jméno ve formátu „output_ALGORITMUS.txt“, kde ALGORITMUS označuje jméno vybraného algoritmu, a jeho obsah je připraven pro pohodlné nahrání softwarem Wolfram Mathematica pomocí příkazu **ReadList[]**, kdy se celý obsah nahraje jako tabulka.

9.1 Povinné parametry algoritmů

Každý soubor je třeba spustit v konzoli s parametry, z nichž některé jsou povinné a další nepovinné. Povinné parametry musí být napsány ve správném pořadí a patří sem:

- číslo testu (v rozsahu 1 až 15),
- velikost dimenze (v rozsahu {10, 30, 50, 100}),
- velikost populace,
- počet iterací,
- počet na sobě nezávislých spuštění algoritmu.

Takže například pokud budeme chtít zapnout algoritmus ARPSO a otestovat pátou funkci pro deset dimenzí, s počtem jedinců 40 a počtem iterací 1500, a tento algoritmus s tímto nastavením zapnout třikrát po sobě napíšeme do konzole „PSO_ARPSO.exe 5 10 40 1500 3“.

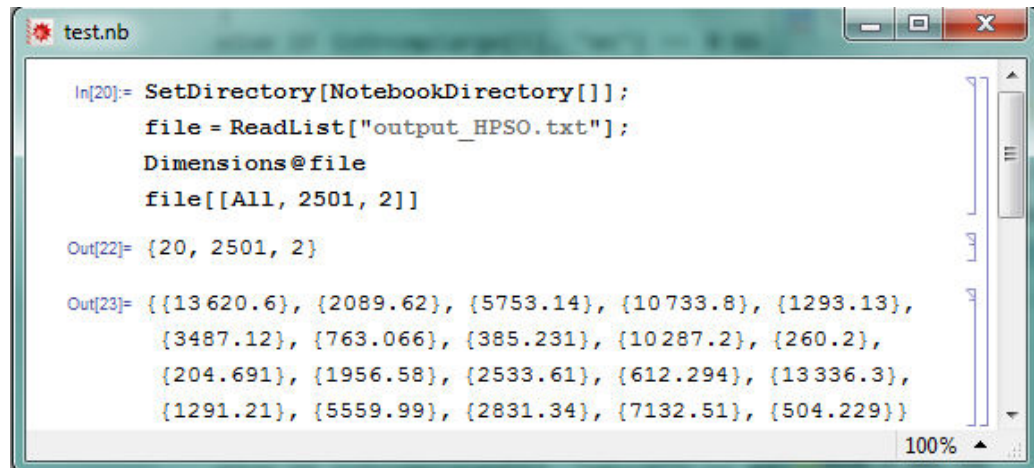


```
c:\>PSO_ARPSO.exe 5 10 40 1500 3
>>>
time spend:
18.190000s
```

Obrázek 12: Náhled na konzoli

Dále máme nepovinné parametry, které se liší svým obsahem mezi jednotlivými algoritmy a budou popsány v samostatné kapitole, stejně tak jako obsah výstupního souboru, který je závislý na zvolených parametrech.

I když všechny vytvořené algoritmy umožňují měnit většinu svých vnitřních proměnných tak se jejich změna nedoporučuje, protože může v některých případech silně ovlivnit



```
In[20]:= SetDirectory[NotebookDirectory[]];
file = ReadList["output_HPSO.txt"];
Dimensions@file
file[[All, 2501, 2]]

Out[22]= {20, 2501, 2}

Out[23]= {{13 620.6}, {2089.62}, {5753.14}, {10 733.8}, {1293.13},
{3487.12}, {763.066}, {385.231}, {10 287.2}, {260.2},
{204.691}, {1956.58}, {2533.61}, {612.294}, {13 336.3},
{1291.21}, {5559.99}, {2831.34}, {7132.51}, {504.229}}
```

Obrázek 14: Náhled na načtený výstupní soubor

V prvním řádku můžeme vidět nastavení složky a na druhém řádku už načítáme vytvořený výstupní soubor. Na třetím řádku je příkaz pro zobrazení dimenzí (první číslo udává počet spuštění algoritmu, druhé číslo udává počet iterací a třetí číslo počet sledovaných proměnných), na dalším je výběr všech posledních hodnot gBestCF z 20 běhů algoritmu.

9.3 PSO

Základní algoritmus PSO má několik nepovinných argumentů, z nichž některé jsou obsaženy i v ostatních programech. Nepovinné argumenty se dělí na bezparametrové (zde patří nastavení, které proměnné se budou ukládat do výstupního souboru) a jednoparametrové (v nich jsou přítomny argumenty pro nastavení vnitřních proměnných algoritmu, určující jeho chování, jako může být například velikost kognitivního faktoru c_1).

9.3.1 Nepovinné argumenty

Tyto argumenty pro spuštění algoritmu se dělí na jednoparametrové, které se zadávají ve formátu „#PARAM #VALUE“. Hodnota #PARAM označuje název proměnné, jejíž hodnotu budeme nastavovat (všechny znaky hodnoty #PARAM musí být psány malým písmem, pokud není uvedeno jinak) a položka #VALUE je pak konkrétní číselná hodnota. Všechny hodnoty jsou ve formě desetinných čísel (typ double), pokud není uvedeno jinak. Patří sem:

- c_1 (kognitivní faktor; defaultní hodnota 2),
- c_2 (sociální faktor; defaultní hodnota 2),
- ws (počáteční velikost setrvačnosti; defaultní hodnota 0.9),

- we (koncová hodnota setrvačnosti; defaultní hodnota 0.4),
- vm (maximální velikost rychlosti; defaultní hodnota 40).

Dalšími argumenty mohou být bezparametrové, zadávané ve tvaru „#VALUE“ malým písmem pokud není uvedeno jinak. Tyto argumenty určují, které proměnné bude algoritmus logovat (zapisovat) do výstupního souboru. Možnosti jsou:

- pop (souřadnice jedinců; parametry řešení; $x_{i,d}$),
- popcf (hodnota účelové funkce jedinců; $f(x_{i,d})$),
- pbest (souřadnice nejlepšího nalezeného řešení jedinců; $pBest_{i,d}$),
- pbestcf (hodnota účelové funkce nejlepšího nalezeného řešení jedinců; $pBestCF_i$),
- gbest (souřadnice nejlepšího nalezeného řešení v populaci; $gBest_d$),
- gbestcf (hodnota účelové funkce nejlepšího nalezeného řešení v populaci; $gBestCF$).

Pokud není zadán žádný argument, tak se budou logovat všechny proměnné v pořadí pop, popSpeed (rychlost jednotlivých jedinců; $popSpeed_{i,d}$), popCF, pBest, pBestCF, gBest, gBestCF. Počet sledovaných proměnných je úměrný době trvání algoritmu.

Příklad nastavení PSO algoritmu pro sledování parametrů gBest a gBestCF, při kognitivním faktoru $c1 = 0$ a maximální rychlosti $v_{max} = 10$ (test = 1, dim = 10, NP = 40, iterací = 2000, počet opakování 1):

„PSO.exe 1 10 40 2000 1 c1 0 vm 10 gbest gbestcf“.

```
c:>PSO.exe 1 10 40 2000 1 c1 0 vm 10 gbest gbestcf
Optional parameter c1: 0.000000
Optional parameter vMax: 10.000000
Optional log add 'gBest'
Optional log add 'gBestCF'
>
time spend:
0.176000s
```

Obrázek 15: Náhled na konzoli s nastavením pro PSO

9.3.2 Vnitřní struktura algoritmu

Základní struktura je stejná pro všechny implementované algoritmy. Proto tuto strukturu popíšeme jen u PSO algoritmu a u ostatních uvedeme pouze změny, přidané proměnné a funkce.

Základní program ve funkci `main()` je zobrazen na obrázku 16. Jako první se inicializuje generátor náhodných čísel, poté se nastaví soubor pro logování sledovaných proměnných. V dalším kroku se parsují argumenty předané při zapínání algoritmu v příkazovém řádku a následně se alokuje paměť všech potřebných proměnných. Poslední částí tohoto úvodního nastavování je zapnutí počítadla pro záznam spotřebovaného času samotným algoritmem, které je pak zastaveno po jeho skončení a výsledný čas je vypsán na konzoli. Po skončení algoritmu se veškerá alokovaná paměť uvolní a zavře se i soubor pro logování.

```
/*  
    Main entry of program  
*/  
int main(int argc, char **argv) {  
    //init rand  
    srand(time(NULL));  
    //log inicialization  
    logInit();  
    //parse arguments  
    argumentParse(argc, argv);  
    //init memory  
    memoryInit();  
    //time spend on algoritm  
    clock_t timeStart, timeEnd;  
    timeStart = clock();  
    //main program  
    //-----  
    //program run  
    ...  
    //-----  
    timeEnd = clock();  
    printf("\ntime spend: \n%fs\n\n", (double)(timeEnd - timeStart) / CLOCKS_PER_SEC);  
    //free memory  
    memoryFree();  
    logFree();  
    return 0;  
}
```

Obrázek 16: Náhled na strukturu `main()`

Samotná struktura algoritmu je pak v části „main program“ a její obsah můžeme vidět na obrázku 17. Na začátku se resetují hodnoty veškerých proměnných. Následně se nastaví jejich logování a zapíší se do souboru jejich inicializační hodnoty. Poté již nastává část algoritmu PSO, kdy se nejdříve počítá setrvačnost, rychlost jedince a jeho nová pozice. Nová pozice se ohodnotí a porovná s `pBest` a `gBest` zda nedošlo ke zlepšení těchto hodnot. Na závěr se po každé iteraci uloží hodnoty všech sledovaných proměnných.

```
for (int r = 0; r < runMax; r++) {
    //reset PSO variables
    reset();
    //start logging new iteration
    logNewRun();
    //init log
    log(0);
    //iterations
    for (int it = 0; it < sizeIt; it++) {
        //calculate w (inertia weight)
        calculateW(it);
        //iterations over population size
        for (int i = 0; i < sizePop; i++) {
            //iterations over dimension size
            for (int j = 0; j < sizeDim; j++) {
                //calculate v
                calculateV(i, j);
                //calculate new position
                calculatePosition(i, j);
            }
            //new fitness
            fitness(pop[i], &popCF[i]);
            //check pBest
            calculatePBest(i);
            //check gBest
            calculateGBest(i);
        }
        log(it);
    }
    //end of run logging
    logEndRun();
    printf(">");
}
```

Obrázek 17: Struktura algoritmu PSO

Globální proměnné používané mezi jednotlivými funkcemi programu jsou uvedeny na začátku před funkcí **main()**. Patří zde například proměnné určující velikost populace, dimenze a počtu iterací. Dále jsou zde například proměnné udržující polohu jedinců (pop), jejich ohodnocení (popCF), nebo i hodnota maximální rychlosti (vMax) a učící faktory (c_1 a c_2). Všechny tyto proměnné jsou okomentovány v patřičném zdrojovém souboru.”

Dále jsou zde deklarovány veškeré používané funkce, které si popíšeme. Jejich deklarace je pak uvedena po funkci **main()**. Jde o obecné funkce, které jsou společné všem algoritmům:

- **argumentParse;**
funkce pro parsování argumentů při zapínání programu,

- **memoryInit;**
funkce pro alokaci dynamických proměnných,
- **memoryFree;**
funkce pro uvolnění paměti alokovaných ve funkci **memoryInit**,
- **reset;**
tato funkce se stará o nastavení defaultních hodnot proměnných při spuštění nového běhu algoritmu,
- **randomNormal;**
funkce vrací náhodné číslo s Gaussovým rozložením v rozsahu stanoveném jejími argumenty,
- **randomUniform;**
funkce vrací náhodné číslo s rovnoměrným rozložením v rozsahu stanoveném jejími argumenty,
- **fitness;**
funkce ohodnotí jedince vloženého jako první argument a toto ohodnocení uloží do proměnné předané jako druhý argument,
- **log;**
logování veškerých vybraných proměnných do souboru pomocí funkce **logVariable**,
- **logVariable;**
tato funkce ukládá do souboru vždy jednu proměnnou předanou argumentem,
- **logNewRun;**
funkce připraví logovací soubor pro ukládání nového spuštění algoritmu,
- **logEndRun;**
ukončení logování aktuálně spuštěného algoritmu,
- **logInit;**
inicializace souboru pro logování vybraných proměnných,
- **logFree;**
uzavření souboru pro logování,
- **logAdd;**
funkce pro přidání nové proměnné ke sledování.

Posledními funkcemi jsou ty, které jsou svým chováním odlišné mezi jednotlivými algoritmy:

- **calculateW;**
funkce počítá hodnotu setrvačnosti pro každou iteraci,
- **calculateV;**
výpočet rychlosti pro každého jedince a jeho dimenzi s následnou kontrolou zda nebyla překročena maximální hodnota této rychlosti,
- **calculatePosition;**
výpočet nové pozice jedince $x_{i,d}$ z původní pozice a nové rychlosti; opět následuje kontrola, zda jedinec svou novou pozicí nepřekročil stanovené hranice,
- **calculatePBest;**
kontrola zda jedinec vylepšil svoji hodnotu pBest,
- **calculateGBest;**
kontrola zda pBest jedince vylepšil gBest populace.

9.4 ARPSO

Tento algoritmus má většinu nepovinných argumentů stejných jako základní PSO algoritmus, ze kterého jeho struktura vychází. Výstupní soubor se jmenuje „output_ARPSO.txt“ a jeho vnitřní struktura je totožná jako u PSO.

9.4.1 Nepovinné argumenty

Tvar argumentů a jejich typy (jednoparametrové a bezparametrové) jsou zachovány jako u PSO algoritmu. Proto v této kapitole uvedeme jen argumenty, které jsou přidáné nebo ty které se změnilly. Opět jsou všechny číselné hodnoty ve formě desetinných čísel (typ double).

Nové jednoparametrové argumenty pro algoritmus ARPSO jsou dva:

- dl (dLow; spodní hranice rozmanitosti; defaultní hodnota je $5 \cdot 10^{-6}$),
- dh (dHigh; horní hranice rozmanitosti; defaultní hodnota je 0.25).

Přibyly také dva nové bezparametrové argumenty pro logování proměnných. Jsou jimi:

- diversity (rozmanitost),
- dir (směr pohybu jedinců; chování; 1 = normální chování, 2 = odpuzující chování).

Pokud nezadáme žádnou proměnou ke sledování tak budou logovány všechny v pořadí pop, popSpeed, popCF, pBest, pBestCF, gBest, gBestCF, diversity, dir.

9.4.2 Vnitřní struktura

Oproti PSO zde přibyly některé globální proměnné (dir = směr pohybu jedinců, diversity = rozmanitost jedinců) a funkce pro jejich výpočet. Struktura programu se od PSO nijak výrazně nezměnila. Nové funkce jsou:

- **setDirection;**
funkce, která nastavuje směr pohybu jedinců podle hodnoty rozmanitosti (diversity),
- **calculateDiversity;**
výpočet rozmanitosti jedinců v populaci.

9.5 HPSO

U tohoto algoritmu přibylo, vzhledem k použitým chováním, několik jednoparametrových argumentů a několik jich bylo oproti PSO odebráno. Výstupní soubor se jmenuje „output_HPSO.txt“.

9.5.1 Nepovinné argumenty

V této kapitole uvedeme všechny přípustné jednoparametrové argumenty z důvodu jejich velkých změn proti PSO, protože struktura toho algoritmu se od něj značně liší. U bezparametrových argumentů uvedeme opět jen nové přírůstky oproti PSO, protože zbytek zde zůstal beze změny.

Nepovinné jednoparametrové argumenty pro HPSO jsou:

- k (počet k-iterací, po které se počítá přínos chování; pouze celé číslo; defaultní hodnota je 10),
- st (stagnationThreshold; počet možných stagnací pBest než dojde ke změně strategie; pouze celé číslo; defaultní hodnota je 5),
- ts (tournamentSize; počet chování náhodně vybraných pro turnajové rozdělení; pouze celé číslo; defaultní hodnota je 2),
- c1_TVAC_s (pouze chování TVAC; počáteční hodnota c_1 ; defaultní hodnota 2.5),
- c1_TVAC_e (pouze chování TVAC; koncová hodnota c_1 ; defaultní hodnota 0),

- $c2_TVAC_s$ (pouze chování TVAC; počáteční hodnota c_2 ; defaultní hodnota 0),
- $c2_TVAC_e$ (pouze chování TVAC; koncová hodnota c_2 ; defaultní hodnota 2.5),
- $c1_TVIW$ (pouze chování TVIW; hodnota c_1 ; defaultní hodnota 1),
- $c2_TVIW$ (pouze chování TVIW; hodnota c_2 ; defaultní hodnota 1),
- $c1_cPSO$ (pouze chování cPSO; hodnota c_1 ; defaultní hodnota 2),
- $c2_cPSO$ (pouze chování cPSO; hodnota c_2 ; defaultní hodnota 0),
- $c1_sPSO$ (pouze chování sPSO; hodnota c_1 ; defaultní hodnota 0),
- $c2_sPSO$ (pouze chování sPSO; hodnota c_2 ; defaultní hodnota 2),
- cr_QSO (pouze chování QSO; počáteční hodnota cloud rádiusu; defaultní hodnota 100),
- ps_modBB (pouze chování modBB; počáteční hodnota ep ; defaultní hodnota 0),
- pe_modBB (pouze chování modBB; koncová hodnota ep ; defaultní hodnota 1),
- ws (počáteční velikost setrvačnosti; defaultní hodnota 0.9),
- we (koncová hodnota setrvačnosti; defaultní hodnota 0.4),
- vm (maximální velikost rychlosti; defaultní hodnota 100).

I když algoritmus umožňuje změnu proměnných jednotlivých strategií, tak je silně doporučeno je nechat beze změny. Jinak se výrazně ovlivní jejich chování, což může vést k jejich nežádoucí degradaci.

Bezparametrové argumenty zůstaly stejné jako u PSO a přibyly jen tři nové proměnné ke sledování:

- $poptresholds$ (počet stagnací hodnoty pBest jedince; $\{s_1, s_2, \dots, s_{NP}\}$),
- $succescounter$ (počet vylepšení hodnot pBest po k-iteraci; $\{\{behav1, behav2, \dots\}_1, \{behav1, behav2, \dots\}_2, \dots, \{behav1, behav2, \dots\}_k\}$),
- $popbehaviors$ (aktuální chování každého jedince; $\{behavX_1, behavX_2, \dots, behavX_{NP}\}$).

Pokud opět nezadáme žádné proměnné ke sledování, budou se logovat v pořadí jako u PSO a na konci přibudou tři nově zmíněné $popTreshold$, $succesCounter$ a $popBehavior$.

9.5.2 Vnitřní struktura

Ač tento algoritmus hodně mění základní PSO tak struktura zůstala nezměněna a veškeré změny se projevují v již probraných funkcích. Přibylo zde také několik nových globálních proměnných, které mají vliv na jednotlivé chování. Pro úplnost zde uvedeme funkce které HPSO nějak upravuje nebo přidává:

- **calculateV;**
výpočet rychlosti je jedinečný pro každou strategii,
- **calculatePosition;**
výpočet nové pozice jedince se také liší od toho, jak se počítala u PSO,
- **calculatePBest;**
při kontrole pBest se kontroluje zlepšení u jednotlivých chování a v případě nutnosti se vybere pomocí **tournamentSelection** nové,
- **tournamentSelection;**
vrací nové chování pro jedince z behavior poolu,
- **nextIteration;**
posun počítadla k-iterací vylepšení pBest,
- **sumOfSuccesCounter;**
součet vylepšení pBestu pro každé chování v k-iteracích.

9.6 OLPSO

I když tento algoritmus přináší zásadní změnu, jak se vypočítává nová rychlost jedinců, tak jeho vnitřní struktura zůstala podobná základnímu PSO. Z toho důvodu se nepovinné argumenty uvedou jen ty, které se změnily, nebo přibyly. Výstupní soubor se jmenuje „output_OLPSO.txt“

9.6.1 Nepovinné argumenty

U jednoparametrových argumentů oproti PSO ubyla možnost nastavit sociální faktor c_2 , který se zde nepoužívá. A zůstal tak jen argument pro nastavení c_1 , který se nastavuje stejně jako u PSO. Přibyl tak jediný jednoparametrový argument:

- **g** (reconstruction gap; počet stagnací pBest, než bude nutné znovu sestavit vektor \mathbf{P}_0 ; pouze celá čísla; defaultní hodnota je 5).

K bezparametrovým argumentům přibyly oproti PSO následující proměnné:

- **matrixl** (matice L ; počet řádků M , počet sloupců dim),
- **gvector** (guidance vector \mathbf{P}_o ; $\{\mathbf{P}_{o,1,d}, \mathbf{P}_{o,2,d}, \dots, \mathbf{P}_{o,NP,d}\}$),
- **stagnation** (počítadlo stagnací pBest každého jedince; $\{s_1, s_2, \dots, s_{NP}\}$).

Tyto proměnné se budou logovat v tomto pořadí následující za těmi z PSO, pokud nebudeme žádné ke sledování.

9.6.2 Vnitřní struktura

U toho algoritmu přibylo několik globálních proměnných a přibyly dvě funkce. Základní struktura programu se od PSO nijak výrazně nezměnila. Nové funkce tedy jsou:

- **calculateOLPSOVars;**
v této funkci vypočítáváme veškeré proměnné, které pak zůstávají v celém programu konstantní, jako je počet řádků a sloupců matice L a samotná matice,
- **calculateVector;**
zde se počítá guidance vector \mathbf{P}_o pokud je potřeba na základě dosažení hranice neúspěchů při vylepšení pBestu jedince.

10 TESTOVÁNÍ

Každý algoritmus byl otestován na všech 15 funkcích v knihovně CEC 2015. Testy byly spuštěny pro 10 a 30 dimenzí. Každé testování bylo spuštěno 51x a výsledky byly zprůměrovány. Zaznamenávala se pouze hodnota $gBest$ v každé iteraci. Populace byla stanovena $NP = 40$ pro obě velikosti dimenze. Ukončovací podmínkou byl počet iterací, který vychází z maximálního počtu ohodnocení účelové funkce, spočítaný podle:

$$maxIT = \frac{10000 \cdot dim}{NP} \quad (10.1)$$

kde $maxIT$: maximální počet iterací,

dim : velikost dimenze,

NP : velikost populace.

Srovnány byly vždy nejnižší dosažené průměrné hodnoty v každém testu a také absolutně nejnižší dosažená hodnota. Dále se pomocí Wilxonova párového testu porovnali výsledné hodnoty, zda jsou statisticky významné (čím více je hodnota tohoto blízká jedné, tím více jsou na sobě výsledky závislé). Hodnoty jsou statisticky významné, pokud je výsledek testování menší než 5%. [12].

Na závěr byla srovnána časová složitost algoritmů na základě dokumentace pro CEC 2015. Výsledky tohoto srovnání jsou vidět v tabulce 3. Platí, že čím vyšší číslo, tím náročnější je algoritmus. Mezi algoritmy platí závislost. Například algoritmus OLPSO je třikrát náročnější než PSO a jen dvakrát náročnější než HPSO pro $dim = 10$.

Algoritmus	Náročnost	
	$dim = 10$	$dim = 30$
PSO	1.311	3.038
ARPSO	1.230	3.526
HPSO	1.946	5.086
OLPSO	4.002	27.507

Tabulka 3: Srovnání náročnosti
vybraných algoritmů

10.1 Souhrnné výsledky testování pro deset dimenzí

V tabulce 4 jsou zobrazeny jednotlivé průměrné hodnoty výsledků každého testu pro 10 dimenzí. Tučně je potom vyznačena nejnižší dosažená hodnota v daném testu, která je statisticky významná oproti výsledku PSO algoritmu.

Z výsledků pro 10 dimenzí je patrné, že pro unimodální funkce (f1, f2) si nejlépe vedl algoritmus HPSO. Tento algoritmus si obstojně vedl i v dalších funkcích (f3, f9, f11 a f13). U multimodální funkce f3 dokázal najít nejmenší hodnotu, ale ostatní algoritmy se zde se svými výsledky moc zřetelně neliší. U funkce f9, která je prvním zástupcem složených funkcí má sice opět podle tabulky nejlepší průměrný výsledek, nicméně algoritmus ARPSO se této hodnotě také hodně přiblížil.

Z moderních algoritmů se potom dařilo lépe algoritmu ARPSO než OLPSO. Je poněkud nečekané jak skvělých výsledků dokázal dosáhnout klasický PSO, který je svojí úspěšností hned po algoritmu HPSO.

Zajímavý je i výsledek posledního testu kdy se všechny algoritmy „zasekly“ na stejné hodnotě.

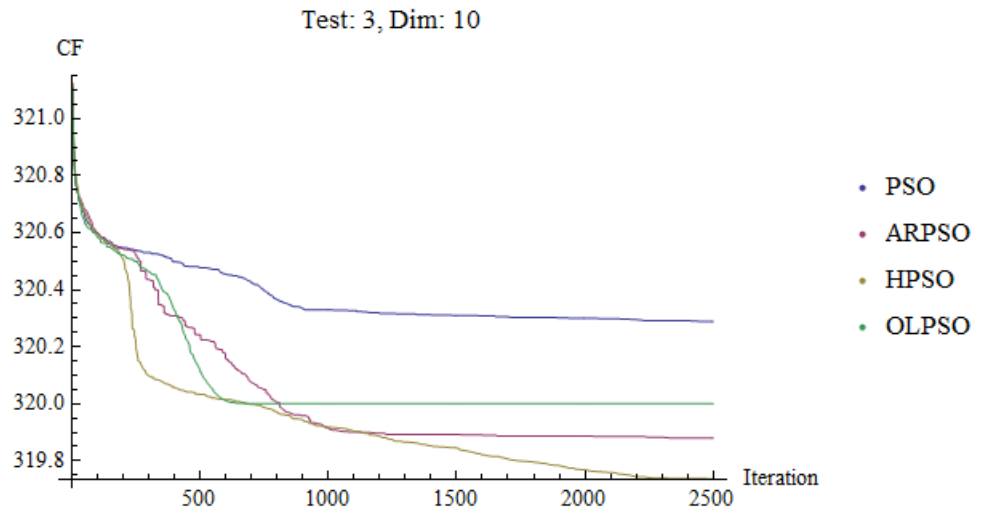
Je nicméně je třeba ještě podotknout, s jakou rychlostí se algoritmy dokázaly těmto výsledkům přiblížit. Z grafického srovnání vždy jako první dosáhne nejnižších výsledků algoritmus HPSO následovaný algoritmem OLPSO. Kvůli značné podobnosti základního PSO algoritmu s ARPSO dosahují tyto algoritmy výsledků často v podobnou dobu.

f	PSO	ARPSO	HPSO	OLPSO
1	131942	196164.	68349.6	184489.
2	8466.86	7843.1	5866.76	6803.53
3	320.288	319.879	319.735	320.
4	407.804	407.921	409.68	412.661
5	723.73	710.503	775.757	819.508
6	1574.82	1873.19	2163.58	2455.25
7	700.718	700.779	700.732	700.624
8	1435.55	1738.75	2148.22	1458.64
9	1000.22	1000.209	1000.205	1000.35
10	1766.15	1876.53	1907.07	1996.15
11	1353.78	1324.68	1307.51	1325.37
12	1302.06	1302.22	1302.5	1302.73
13	1332.38	1333.17	1330.32	1334.02
14	4425.93	4189.75	5773.92	5036.94
15	1600.	1600.	1600.	1600.

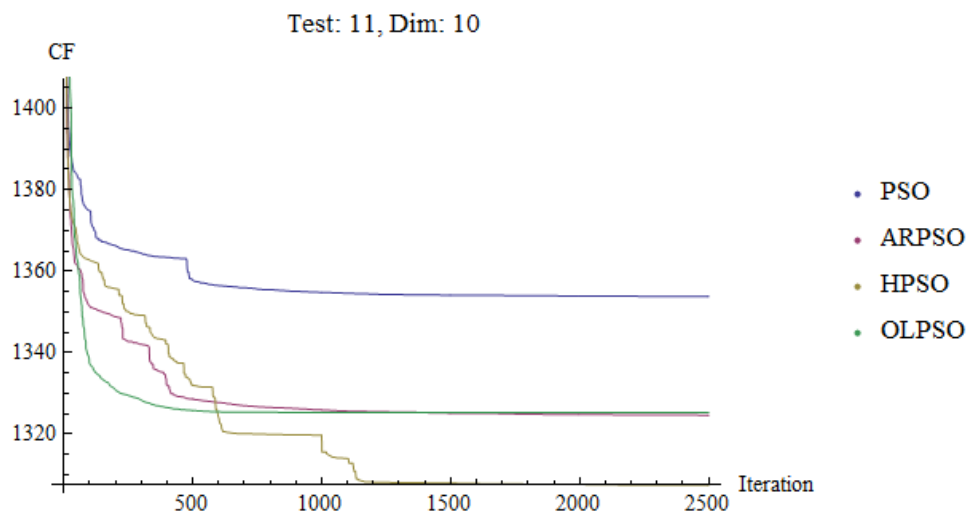
Tabulka 4: Srovnání průměrných výsledků pro 10 dimenzí

U grafického srovnání výsledků testu f3 je tato skutečnost jasně vidět (Obrázek 18). U grafu pro test f11 (Obrázek 19) je patrná změna dominantních chování u algoritmu HPSO, které jsou zde vidět jako skoková změna vylepšení účelové funkce. Graf (Obrázek 21) testu f14 v tomto tvoří výjimku. Pro poslední test f15 je zde uveden graf (Obrázek 22) pro srovnání rychlosti dosažení výsledku jednotlivých algoritmů.

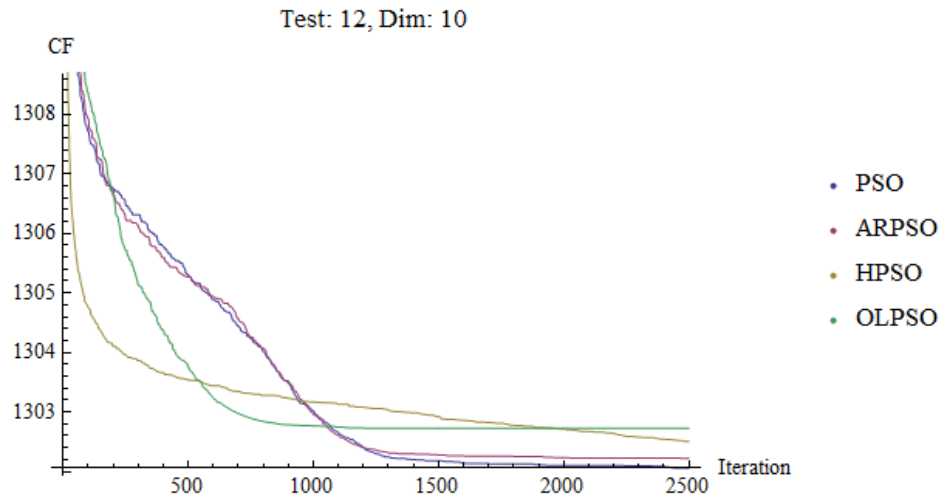
Z těchto grafických výsledků je i vidět důvod proč často základní algoritmus PSO dosahoval lepších výsledků, velice často se totiž stávalo, že i když algoritmy HPSO a OLPSO začali stagnovat, tak PSO a ARPSO měli pořád tendence se dále zlepšovat (jak je vidět na obrázku Obrázek 20: Graf testu f12 pro dim=10).



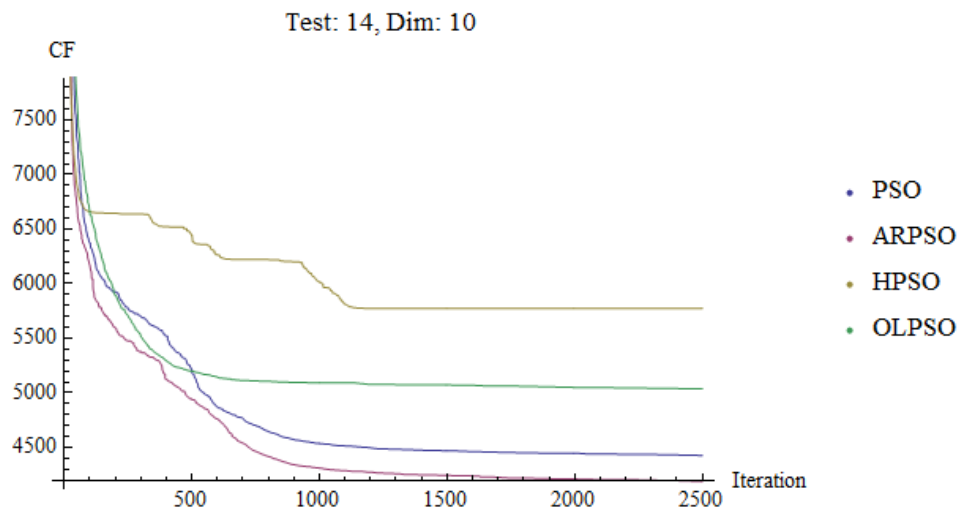
Obrázek 18: Graf testu f3 pro dim=10



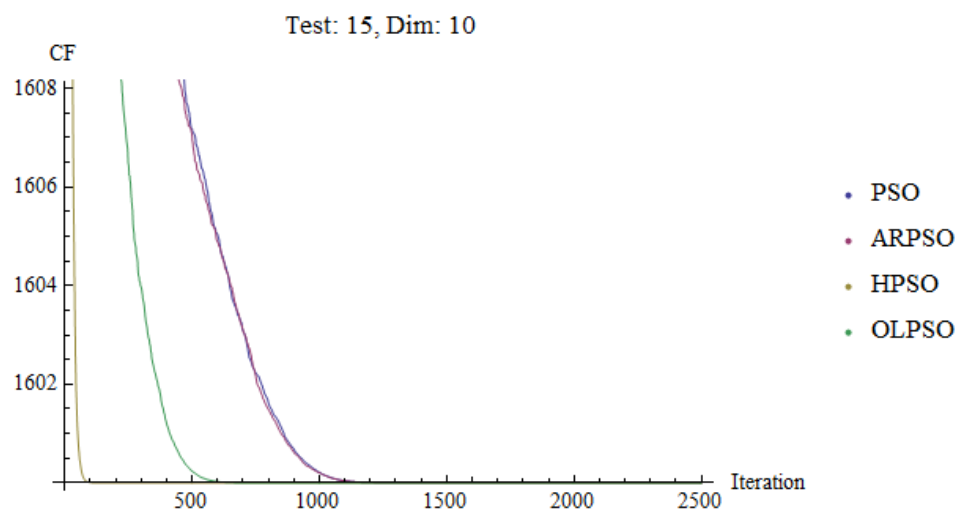
Obrázek 19: Graf testu f11 pro dim=10



Obrázek 20: Graf testu f12 pro dim=10



Obrázek 21: Graf testu f14 pro dim=10



Obrázek 22: Graf testu f15 pro dim=10

Dále ještě uvedeme tabulku 5, srovnání výsledků Wilcoxonova párového testu. Hodnoty algoritmů (ARPSO, HPSO a OLPSO) byly vždy srovnány s výsledky základního algoritmu PSO. Čím menší je číslo v tabulce, tím více jsou výsledky rozdílné. U testu f15 je vidět že všechny výsledky jsou stejné, tzn. výsledek Wilcoxonova párového testu je rovna jedné.

V tabulkách 6, 7, 8 a 9 jsou uvedeny detailní statistické analýzy výsledků každého algoritmu (nejhorší a nejlepší dosažená hodnota, aritmetický průměr hodnot, medián a směrodatná odchylka).

f	ARPSO	HPSO	OLPSO
1	4.804E-03	2.413E-02	5.158E-04
2	1.895E-02	7.517E-02	2.262E-02
3	2.294E-02	4.740E-02	5.224E-04
4	9.902E-03	7.976E-03	2.522E-05
5	4.838E-03	3.978E-02	8.979E-03
6	1.480E-05	3.946E-02	3.505E-04
7	4.177E-05	6.104E-04	7.365E-02
8	1.701E-03	4.790E-02	9.743E-04
9	1.490E-02	5.792E-02	1.749E-6
10	5.562E-02	1.468E-05	1.641E-02
11	1.341E-01	3.103E-02	1.650E-02
12	8.853E-04	1.136E-01	4.944E-03
13	1.379E-03	2.250E-02	6.344E-02
14	1.223E-03	1.356E-02	1.240E-03
15	1.000E+00	1.000E+00	1.000E+00

Tabulka 5: Výsledky Wilcoxonova párového testu pro 10 dimenzí

f	Min	Max	Medián	Průměr	Std
1	844.274	836515	44184.9	131942	183943
2	200.351	100967.	3976.82	8466.86	15203.5
3	320.083	320.436	320.312	320.288	0.079
4	401.99	417.909	406.965	407.804	3.534
5	506.892	1114.14	655.988	723.73	155.722
6	628.413	6368.22	1190.42	1574.82	1146.36
7	700.02	701.194	701.016	700.718	0.449
8	801.203	4725.37	1122.33	1435.55	819.916
9	1000.13	1000.34	1000.22	1000.22	0.055
10	1225.41	7756.65	1494.52	1766.15	1006.98
11	1101.95	1400.83	1400.19	1353.78	108.833
12	1301.02	1303.89	1302.05	1302.06	0.619
13	1321.16	1343.35	1332.14	1332.38	4.296
14	1500.	12856.1	4408.38	4425.93	3415.14
15	1600.	1600.	1600.	1600.	0.

Tabulka 6: Přehled výsledků testování algoritmu PSO
pro deset dimenzí

f	Min	Max	Medián	Průměr	Std
1	666.382	1.709E+6	68463.1	196164.	357157.
2	262.275	30966.5	4662.18	7843.1	7847.99
3	300.	320.446	320.281	319.879	2.840
4	402.985	414.924	407.96	407.921	3.083
5	503.727	1048.17	688.877	710.503	132.477
6	696.495	6321.28	1463.15	1873.19	1390.54
7	700.037	701.257	701.013	700.779	0.414
8	826.121	4692.83	1360.73	1738.75	1044.12
9	1000.11	1000.37	1000.2	1000.21	0.053
10	1243.77	7160.1	1588.07	1876.53	1073.05
11	1101.67	1401.02	1400.14	1324.68	130.57
12	1300.89	1303.4	1302.22	1302.22	0.648
13	1326.89	1340.84	1332.98	1333.17	3.500
14	1500.	12862.7	4396.09	4189.75	3308.82
15	1600.	1600.	1600.	1600.	3.566E-6

Tabulka 7: Přehled výsledků testování algoritmu
ARPSO pro deset dimenzí

f	Min	Max	Medián	Průměr	Std
1	333.168	280357.	56232.7	68349.6	60750.4
2	200.106	22022.4	2948.28	5866.76	6062.65
3	302.317	320.202	320.097	319.735	2.488
4	401.99	422.884	408.955	409.68	5.353
5	503.665	1252.59	756.672	775.757	171.222
6	616.156	7582.82	1403.67	2163.58	1912.73
7	700.028	702.538	701.009	700.732	0.529
8	801.262	9310.22	1535.36	2148.22	1784.61
9	1000.11	1000.39	1000.21	1000.21	0.046
10	1237.11	4567.84	1683.48	1907.07	694.457
11	1102.66	1400.65	1400.13	1307.51	138.503
12	1301.38	1303.82	1302.52	1302.5	0.563
13	1324.07	1340.86	1329.45	1330.32	3.286
14	1500.	13256.5	4380.4	5773.92	3046.32
15	1600.	1600.	1600.	1600.	0.

Tabulka 8: Přehled výsledků testování algoritmu
HPSO pro deset dimenzí

f	Min	Max	Medián	Průměr	Std
1	123.264	1.418E+6	3252.02	123.264	334621.
2	202.324	24656.	3674.21	202.324	7374.95
3	319.997	320.	320.	319.997	4.35E-04
4	402.985	430.844	410.945	402.985	5.65317
5	511.83	1387.36	759.029	511.83	201.168
6	601.67	16890.9	1509.32	601.67	2879.3
7	700.027	701.184	700.691	700.027	0.432
8	802.457	3532.32	1256.42	802.457	672.225
9	1000.17	1000.6	1000.34	1000.17	0.111
10	1246.86	6675.38	1695.35	1246.86	982.955
11	1101.02	1400.92	1400.22	1101.02	129.516
12	1300.91	1306.69	1302.38	1300.91	1.058
13	1323.72	1341.67	1333.68	1323.72	4.170
14	1500.	9009.83	4401.47	1500.	2800.78
15	1600.	1600.	1600.	1600.	0.

Tabulka 9: Přehled výsledků testování algoritmu OLPSO pro deset dimenzí

10.2 Souhrnné výsledky testování pro třicet dimenzí

V tabulce 10 jsou zobrazeny jednotlivé průměrné hodnoty výsledků každého testu pro 30 dimenzí. Tučně je potom vyznačena nejnižší dosažená hodnota v daném testu, která je statisticky významná oproti výsledku PSO algoritmu.

Při této vyšší velikosti dimenze se více projevují moderní verze PSO algoritmu. Například algoritmus OLPSO dosahuje lepších výsledků jak při deseti dimenzích. To by mohlo být

způsobeno tím, jak se počítá rychlost jedinců a má tak lepší podmínky dosáhnout zlepšení. Naopak základní verze PSO již těchto dobrých výsledků nedosahuje.

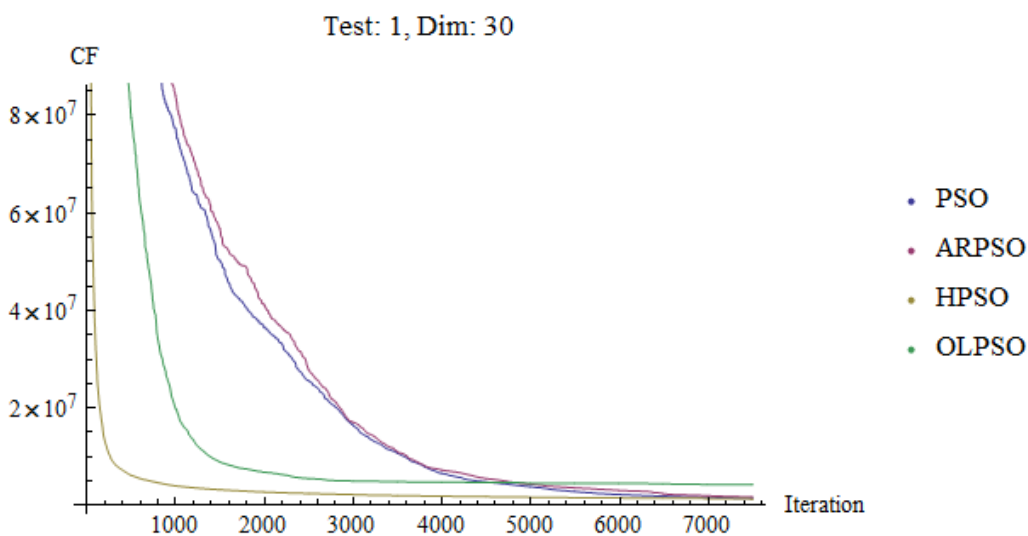
V unimodálních funkcích (f1 a f2) jednoznačně převažují HPSO a OLPSO algoritmus. U multimodálních funkcí se algoritmu OLPSO dařilo při funkci Ackley (f3).

Pro hybridní funkce jsou jednotlivé algoritmy rovnoměrně zastoupeny a u složených s přehledem vede algoritmus HPSO až na funkci f11, kde byl lepší algoritmus OLPSO. Funkce f15 opět pro všechny použité algoritmy znamenala stagnaci na hodnotě 1600.

Co se týče grafického srovnání, tak se ve většině případů podobají grafům pro deset dimenzí. Naopak se ale většinou nejrychleji stagnaci přiblíží algoritmus OLPSO, který je následován algoritmem HPSO (kromě unimodálních funkcí, Obrázek 23) jak je vidět na grafech (Obrázek 24, Obrázek 25 a Obrázek 26). ARPSO a PSO mají často podobný průběh.

Na grafu Obrázek 27 je vidět, že i při této velikosti dimenze mají algoritmy PSO a ARPSO dlouhodobé tendence se zlepšovat a naopak HPSO a OLPSO začnou brzo stagnovat (nejnižší hodnoty dosáhl HPSO).

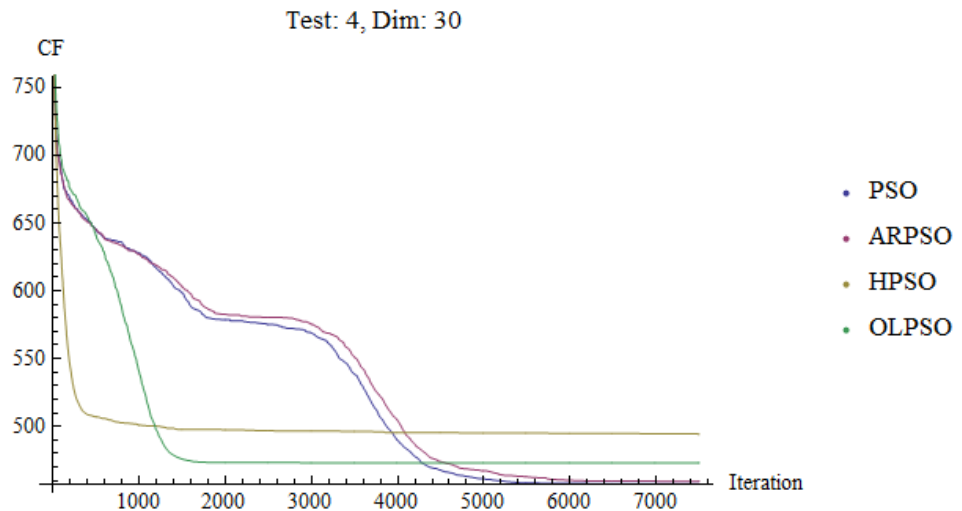
V tabulkách 12, 13, 14 a 15 jsou uvedeny detailní statistické analýzy výsledků každého algoritmu (nejhorší a nejlepší dosažená hodnota, aritmetický průměr hodnot, medián a směrodatná odchylka).



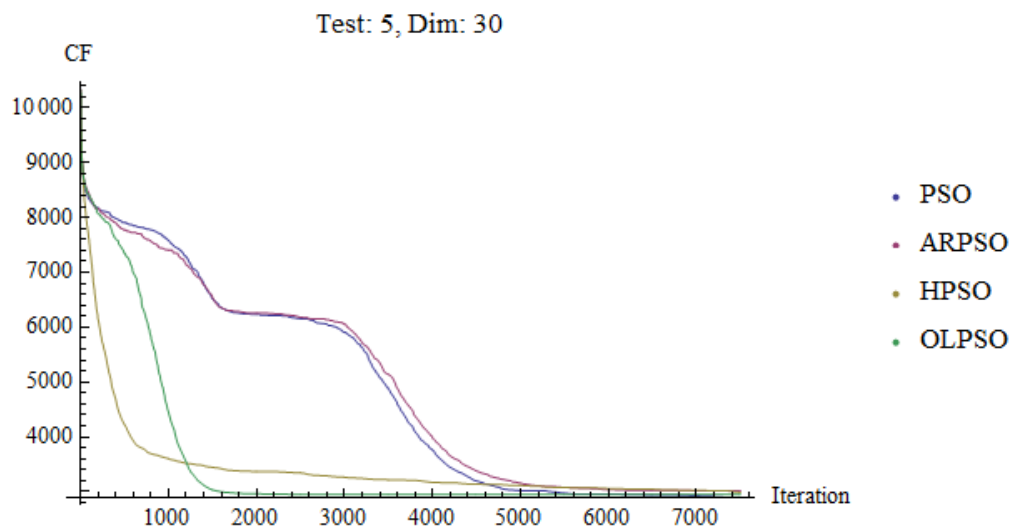
Obrázek 23: Graf testu f1 pro dim=30

f	PSO	ARPSO	HPSO	OLPSO
1	1.441E+6	1.717E+6	1.299E+6	4.285E+6
2	3004.41	3563.18	5240.34	2607.1
3	320.852	320.848	320.395	320.
4	457.493	459.483	494.108	473.098
5	2896.46	3018.38	2982.88	2946.53
6	132158.	95536.6	195614.	362442.
7	708.631	708.345	708.066	707.6
8	52077.1	62965.5	83981.2	56196.7
9	1015.73	1028.67	1013.55	1014.17
10	165873.	138832.	135415.	685166.
11	1587.44	1600.64	1868.02	1549.52
12	1308.01	1307.9	1307.64	1309.86
13	1417.88	1417.66	1406.11	1423.17
14	36602.5	36690.9	34436.5	36221.5
15	1600.	1600.	1600.	1600.

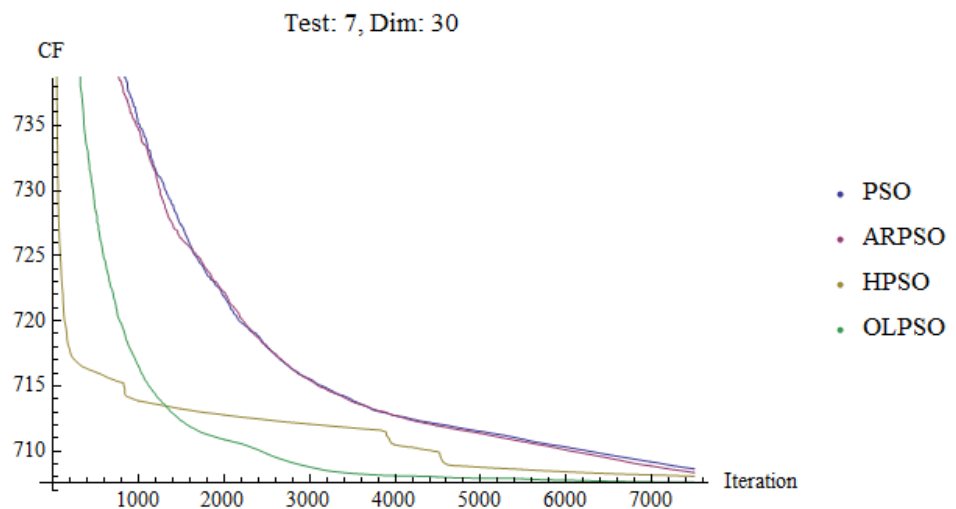
Tabulka 10: Srovnání průměrných výsledků pro 30 dimenzí



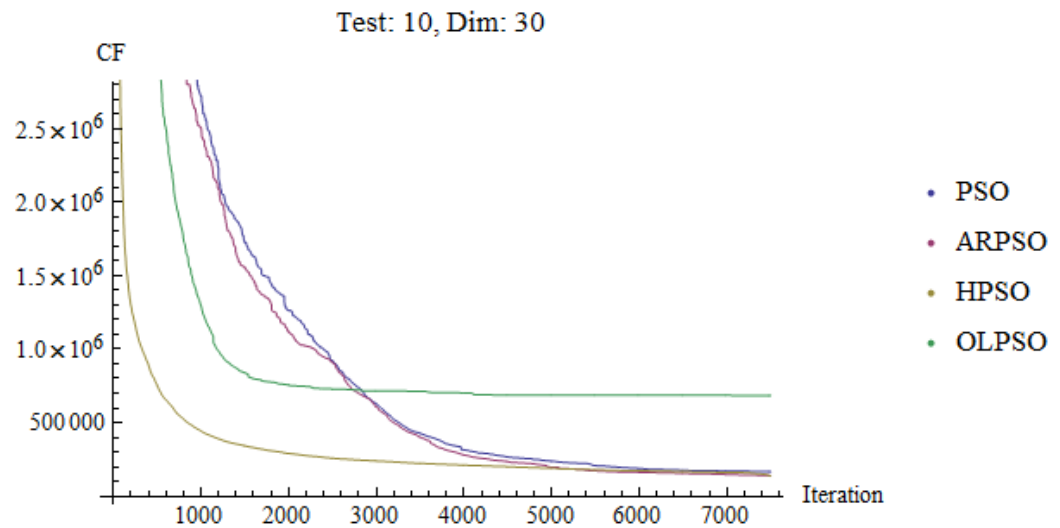
Obrázek 24: Graf testu f4 pro dim=30



Obrázek 25: Graf testu f5 pro dim=30



Obrázek 26: Graf testu f7 pro dim=30



Obrázek 27: Graf testu f10 pro dim=30

Na závěr ještě znovu uvedeme tabulku 11, srovnání výsledků Wilcoxonova párového testu, tentokrát pro $\text{dim} = 30$. Hodnoty algoritmů (ARPSO, HPSO a OLPSO) byly vždy srovnány s výsledky základního algoritmu PSO. Čím menší je číslo v tabulce, tím více jsou výsledky rozdílné. U testu f15 je opět vidět že všechny výsledky jsou stejné.

f	ARPSO	HPSO	OLPSO
1	4.77E-02	2.315E-02	1.411E-03
2	2.766E-03	1.918E-02	8.625E-05
3	7.294E-02	4.999E-02	1.877E-02
4	2.521E-03	5.178E-03	5.801E-02
5	3.216E-03	6.877E-03	2.669E-02
6	3.720E-8	5.090E-04	3.127E-02
7	5.561E-03	2.104E-03	1.629E-02
8	7.598E-02	2.905E-03	5.724E-03
9	5.749E-03	5.167E-03	5.233E-03
10	7.351E-02	1.840E-03	8.364E-03
11	3.160E-02	5.925E-05	2.466E-02
12	1.804E-02	1.204E-02	1.023E-02
13	2.969E-02	2.034E-02	9.227E-05
14	9.812E-02	8.466E-02	2.831E-02
15	1.000E+00	1.000E+00	1.000E+00

Tabulka 11: Výsledky Wilcoxonova párového testu pro 30 dimenzí

f	Min	Max	Medián	Průměr	Std
1	387057.	8.008E+6	1.074E+6	1.441E+6	1.455E+6
2	211.957	13217.4	1122.38	3004.41	3592.52
3	320.65	320.997	320.857	320.852	0.088
4	428.854	492.531	453.728	457.493	13.801
5	1958.	4046.3	2929.19	2896.46	496.418
6	9003.18	681531.	77361.9	132158.	132557.
7	703.974	714.143	708.535	708.631	2.347
8	10033.6	174190.	41386.	52077.1	34014.8
9	1002.44	1184.7	1003.02	1015.73	44.408
10	13085.8	530357.	129127.	165873.	129471.
11	1402.33	1831.56	1632.34	1587.44	158.704
12	1306.11	1311.2	1307.93	1308.01	1.178
13	1389.66	1435.05	1418.82	1417.88	9.886
14	33901.4	41477.4	36002.1	36602.5	2039.05
15	1600.	1600.	1600.	1600.	0.

Tabulka 12: Přehled výsledků testování algoritmu PSO pro třicet dimenzí

f	Min	Max	Medián	Průměr	Std
1	108345.	7.039E+6	1.066E+6	1.717E+6	1.744E+6
2	205.656	12709.1	1696.26	3563.18	3744.57
3	320.529	321.008	320.867	320.848	0.089
4	424.874	499.496	458.702	459.483	15.925
5	1920.04	4329.41	2950.02	3018.38	532.535
6	15130.4	459460.	75249.1	95536.6	82875.7
7	704.655	715.845	707.956	708.345	2.149
8	3889.29	194473.	55838.6	62965.5	42818.7
9	1002.46	1220.09	1002.97	1028.67	61.033
10	19762.9	961543.	112587.	138832.	145970.
11	1402.93	1921.07	1641.59	1600.64	170.331
12	1305.31	1311.61	1307.68	1307.9	1.324
13	1388.56	1430.4	1419.25	1417.66	9.368
14	33717.	42341.	36905.2	36690.9	2130.36
15	1600.	1600.	1600.	1600.	1.540E-05

Tabulka 13: Přehled výsledků testování algoritmu ARPSO pro třicet dimenzí

f	Min	Max	Medián	Průměr	Std
1	136749.	3.732E+6	1.109E+6	1.299E+6	757500.
2	200.015	15135.3	3665.29	5240.34	4446.65
3	320.006	320.614	320.428	320.395	0.123
4	440.793	561.182	492.531	494.108	28.610
5	2019.66	4266.34	2940.72	2982.88	456.394
6	21731.7	464446.	178973.	195614.	118066.
7	704.467	727.766	707.228	708.066	3.377
8	9402.43	337812.	67704.1	83981.2	62725.9
9	1002.56	1202.77	1003.04	1013.55	42.823
10	9695.66	365110.	99565.8	135415.	94686.1
11	1402.45	2169.92	1927.39	1868.02	224.388
12	1305.55	1309.91	1307.77	1307.64	0.939
13	1389.11	1416.13	1406.1	1406.11	5.551
14	32558.7	37462.3	34401.5	34436.5	1353.24
15	1600.	1600.	1600.	1600.	0.

Tabulka 14: Přehled výsledků testování algoritmu HPSO
pro třicet dimenzí

f	Min	Max	Medián	Průměr	Std
1	7712.6	2.797E+7	2.098E+6	7712.6	5.624E+6
2	200.162	9427.32	1420.2	200.162	2530.06
3	320.	320.	320.	320.	3.162E-05
4	421.889	515.415	471.637	421.889	22.102
5	2090.37	4009.24	2975.19	2090.37	481.315
6	3050.44	4.153E+6	36722.9	3050.44	711095.
7	703.304	716.399	706.482	703.304	3.494
8	1799.52	773597.	24350.3	1799.52	135546.
9	1003.14	1205.27	1003.69	1003.14	42.642
10	13013.3	1.388E+7	338975.	13013.3	1.926E+6
11	1403.92	1941.64	1544.74	1403.92	141.498
12	1306.63	1315.9	1309.3	1306.63	1.962
13	1406.53	1440.73	1423.96	1406.53	8.106
14	33152.4	40846.4	35692.	33152.4	2063.32
15	1600.	1600.	1600.	1600.	0.

Tabulka 15: Přehled výsledků testování algoritmu OLPSO
pro třicet dimenzí

10.3 Závěr výsledků testování

Z výsledků testování je vidět, že moderní algoritmy se více uplatňují při vyšší dimenzi. Toto zlepšení je nejvíce zřetelné u algoritmu OLPSO, který se vylepšuje na základě tvorby guidance vektoru \mathbf{P}_0 . Celkově nejlépe vyšel z testování algoritmus HPSO, který exceluje díky svému přístupu, kdy se každý jedinec chová podle odlišné strategie. To umožňuje rychle reagovat na případné oblasti grafu, kde ostatní algoritmy upadají do stagnace. Algoritmus ARPSO se svojí vnitřní strukturou podobá PSO a proto mají hodně výsledků podobných a kvůli jeho tendenci k rozkmitání kolem globálního minima i mnohdy horších. Nicméně díky výsledkům Wilcoxonova testování je zřejmé, že výsledky jsou na sobě nezávislé.

ZÁVĚR

Hlavním cílem práce byla implementace moderních verzí PSO algoritmu v jazyce C. Všechny vytvořené programy byly otestovány pomocí knihovny IEEE CEC 2015 jak z hlediska výkonosti tak i z hlediska výpočetní náročnosti. Graficky byly srovnány průběhy vybraných testů.

Základní princip práce s programy je vždy stejný a každý je popsán přehledným manuálem v příslušné kapitole. Je zde popsána i vnitřní struktura vytvořených programů pro jednoduchou orientaci ve zdrojových kódech. Stejně tak je popsána i struktura výstupních souborů a jak se s nimi dá pracovat v prostředí Wolfram Mathematica, kde byly výsledky graficky vzájemně srovnávány. Veškeré zdrojové kódy jsou navíc i přehledně uspořádané a okomentované.

V teoretické části práce byla zpracována literární rešerše na téma těchto algoritmů a byly zde i implementované varianty podrobně probrány.

Přínos těchto moderních algoritmů je znatelný. Například algoritmus HPSO dosahoval pravidelně kvalitních výsledků při všech testovaných dimenzích. Algoritmus OLPSO své přednosti nejvíce uplatňuje při vyšších dimenzích, kde roste jeho úspěšnost. Naopak algoritmus ARPSO byl svými výsledky překvapivě podobný základnímu PSO algoritmu a v některých případech dosahoval i horších výsledků.

Výsledky této práce naleznou praktické uplatnění ve výuce a výzkumu na FAI UTB ve Zlíně.

SEZNAM POUŽITÉ LITERATURY

- [1] ZELINKA, Ivan. Evoluční výpočetní techniky: principy a aplikace. 1. vyd. Praha: BEN - technická literatura, 2009, 534 s. ISBN 978-80-7300-218-3.
- [2] PLUHACEK, Michal. PSO Algoritmus v prostředí Mathematica. Zlín, 2011. Diplomová práce. Univerzita Tomáše Bati ve Zlíně.
- [3] TVRDÍK, Josef. Učební texty ostravské univerzity: Evoluční algoritmy. Ostrava: Ostravská univerzita, 2004.
- [4] SHI, Yuhui; EBERHART, Russell. A modified particle swarm optimizer. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. IEEE, 1998. p. 69-73.
- [5] ZHAN, Zhi-Hui, et al. Orthogonal learning particle swarm optimization. Evolutionary Computation, IEEE Transactions on, 2011, 15.6: 832-847.
- [6] RIGET, Jacques; VESTERSTROM, Jakob S. A diversity-guided particle swarm optimizer-the ARPSO. Dept. Comput. Sci., Univ. of Aarhus, Aarhus, Denmark, Tech. Rep, 2002, 2: 2002.
- [7] NEPOMUCENO, Filipe V.; ENGELBRECHT, Andries P. A self-adaptive heterogeneous pso for real-parameter optimization. In: Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 2013. p. 361-368.
- [8] CHEN, Q., et al. Problem Definitions and Evaluation Criteria for CEC 2015 Special Session on Bound Constrained Single-Objective Computationally Expensive Numerical Optimization.
- [9] HEROUT, Pavel. Učebnice jazyka C. Praha: Kopp, 2004, 271 s. ISBN 8072322206. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence, Academia, 1993, ISBN 80-200-0496-3.
- [10] TROTT, Michael. The Mathematica guidebook for programming. New York: Springer, c2004, xxxvii, 1028 s. ISBN 0-387-94282-3.
- [11] Wolfram Language & System: Documentation Center. *Wolfram: Computation meets knowledge* [online]. 2016 [cit. 2016-05-01]. Dostupné z: <http://reference.wolfram.com/language/?source=nav>

- [12] ANDĚL, Jiří. Statistické metody. 4., upr. vyd. Praha: Matfyzpress, 2007.
ISBN 978-80-7378-003-6.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

PSO Particle swarm optimization

ARPSO Attractive and Repulsive Particle Swarm Optimization

HPSO Heterogenous Particle Swarm Optimization

OLPSO Orthogonal Learning Particle Swarm Optimization

OED Orthogonal Experimental Design

ACO Ant Colony Optimization

SEZNAM OBRÁZKŮ

Obrázek 1: Diagram průběhu algoritmu PSO.....	15
Obrázek 2: Pohyb jedince ovlivněný učitelskými faktory [1].....	17
Obrázek 3: Diagram průběhu algoritmu ARPSO	22
Obrázek 4: Diagram průběhu algoritmu HPSO	26
Obrázek 5: Diagram průběhu algoritmu OLPSO	30
Obrázek 6: Rotated High Conditioned Elliptic Function [8].....	33
Obrázek 7: Rotated Cigar Function [8].....	33
Obrázek 8: Shifted and Rotated Ackley's Function [8].....	34
Obrázek 9: Shifted and Rotated Rastrigin's Function [8].....	34
Obrázek 10: Shifted and Rotated Schwefel's Function [8]	34
Obrázek 11: Ukázka notebooku v prostředí Wolfram Mathematica	37
Obrázek 12: Náhled na konzoli	39
Obrázek 13: Náhled na konzoli s nastavením pro HPSO	40
Obrázek 14: Náhled na načtený výstupní soubor	41
Obrázek 15: Náhled na konzoli s nastavením pro PSO	42
Obrázek 16: Náhled na strukturu main()	43
Obrázek 17: Struktura algoritmu PSO	44
Obrázek 18: Graf testu f3 pro dim=10.....	54
Obrázek 19: Graf testu f11 pro dim=10.....	54
Obrázek 20: Graf testu f12 pro dim=10.....	55
Obrázek 21: Graf testu f14 pro dim=10.....	55
Obrázek 22: Graf testu f15 pro dim=10.....	55
Obrázek 23: Graf testu f1 pro dim=30.....	61
Obrázek 24: Graf testu f4 pro dim=30.....	63
Obrázek 25: Graf testu f5 pro dim=30.....	63
Obrázek 26: Graf testu f7 pro dim=30.....	63
Obrázek 27: Graf testu f10 pro dim=30.....	64

SEZNAM TABULEK

Tabulka 1: Přehled testovacích funkcí v IEEE CEC 2015 [8].....	31
Tabulka 2: Přehled základních funkcí [8].....	33
Tabulka 3: Srovnání náročnosti vybraných algoritmů.....	51
Tabulka 4: Srovnání průměrných výsledků pro 10 dimenzí.....	53
Tabulka 5: Výsledky Wilcoxonova párového testu pro 10 dimenzí.....	56
Tabulka 6: Přehled výsledků testování algoritmu PSO pro deset dimenzí.....	57
Tabulka 7: Přehled výsledků testování algoritmu ARPSO pro deset dimenzí.....	58
Tabulka 8: Přehled výsledků testování algoritmu HPSO pro deset dimenzí.....	59
Tabulka 9: Přehled výsledků testování algoritmu OLPSO pro deset dimenzí.....	60
Tabulka 10: Srovnání průměrných výsledků pro 30 dimenzí.....	62
Tabulka 11: Výsledky Wilcoxonova párového testu pro 30 dimenzí.....	65
Tabulka 12: Přehled výsledků testování algoritmu PSO pro třicet dimenzí.....	66
Tabulka 13: Přehled výsledků testování algoritmu ARPSO pro třicet dimenzí.....	67
Tabulka 14: Přehled výsledků testování algoritmu HPSO pro třicet dimenzí.....	68
Tabulka 15: Přehled výsledků testování algoritmu OLPSO pro třicet dimenzí.....	69

SEZNAM PŘÍLOH

PŘÍLOHA P I: Obsah vloženého CD-ROM disku

PŘÍLOHA P I: OBSAH VLOŽENÉHO CD-ROM DISKU

Obsah disku:

- Elektronická verze diplomové práce
- Zdrojové soubory implementovaných algoritmů
- Spustitelné programy implementovaných algoritmů