

Vybrané swarm algoritmy a jejich implementace

Bc. Silvia Panenková

Diplomová práce
2015



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Silvia Panenková**

Osobní číslo: **A13497**

Studijní program: **N3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Forma studia: **kombinovaná**

Téma práce: **Vybrané swarm algoritmy a jejich implementace**

Téma anglicky: **Selected Swarm-based Algorithms and Their Implementation**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma, zaměřte se zejména na tzv. hejnové algoritmy.
2. Popište principy a varianty optimalizačního Bee algoritmu.
3. Naprogramujte zvolenou verzi a strategie optimalizačního Bee algoritmu v prostředí C/C++/C#
4. Otestujte vytvořený algoritmus na sadě testovacích benchmark funkcí.
5. Provedte srovnání s některým jiným hejnovým algoritmem (SOMA/PSO/Firefly)
6. Zobrazte přehledně graficky a tabulkově výsledky testování.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. BEN, 2002, 190 s. ISBN 80-7300-069-5.
2. ZELINKA, Ivan. Evoluční výpočetní techniky: principy a aplikace. 1. vyd. Praha: BEN - technická literatura, 2009, 534 s. ISBN 978-80-7300-218-3.
3. DE JONG, Kenneth A. Evolutionary computation: a unified approach. Cambridge: MIT Press, 2006, ix, 256 s. ISBN 02-620-4194-4.
4. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence, Academia, 1993, ISBN 80-200-0496-3.
5. MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence 4., Academia, 2003, ISBN 80-200-1044-0.
6. ZELINKA, Ivan, Zuzana OPLATKOVÁ a Roman ŠENKEŘÍK. Aplikace umělé inteligence. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2010, 151 s. ISBN 978-80-7318-898-6.
7. KENNEDY, James, Russell C EBERHART a Yuhui SHI. Swarm intelligence. San Francisco: Morgan Kaufmann, c2001, xxvii, 512 s. ISBN 1-55860-595-9.
8. Handbook of Optimization: From Classical to Modern Approach. 2013. vyd. Editor Ivan Zelinka, Václav Snášel, Ajith Abraham. Berlin: Springer, 2013, xii, 1100 s. Intelligent systems reference library, 38. ISBN 978-3-642-30503-0.

Vedoucí diplomové práce:

doc. Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Konzultant:

Ing. Erik Král, Ph.D.

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

6. února 2015

Termín odevzdání diplomové práce:

15. května 2015

Ve Zlíně dne 6. února 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



L.S.



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

Prohlašuji, že

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

Prohlašuji,

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

20.5.2015

Ve Zlíně



.....
podpis diplomanta

ABSTRAKT

Diplomová práca sa zaoberá optimalizačnými algoritmami inšpirovanými správaním sa biologických včiel. V práci je bližšie popísaná rojová inteligencia a s ňou sú predstavené niektoré známe moderné optimalizačné algoritmy. Samostatná kapitola je venovaná optimalizácii rojom včiel. Súčasťou práce sú implementované algoritmy – včelí algoritmus (Bees Algorithm), jeho vylepšená verzia (Improved Bees Algorithm) a umelá včelia kolónia (Artificial Bee Colony), ktoré sú popísané v praktickej časti. Všetky algoritmy tejto práce boli testované a porovnávané s ďalšími rojovými algoritmami – Optimalizácia rojom častíc (Particle Swarm Optimization) a optimalizácia svetluškami (Firefly Algorithm). Výsledky testov sa nachádzajú v poslednej časti tejto práce.

Kľúčové slová: včelí algoritmus, vylepšený včelí algoritmus, umelá včelia kolónia, rojová inteligencia, optimalizácia

ABSTRACT

The thesis deals with optimization algorithms inspired by the behaviour of bees. Swarm intelligence is described in more details and some famous modern optimization algorithms are presented. A separate chapter is devoted to the swarm of bees optimization. Implemented algorithms - Bees Algorithm and its improved version (Improved Bees Algorithm) and Artificial Bee Colony are described in the practical part. All of the algorithms have been tested and compared to the other swarm algorithms - Particle Swarm Optimization and Firefly Algorithm. The test results are listed in the last part of the thesis.

Keywords: bees algorithm, improved bees algorithm, artificial bee colony, swarm intelligence, optimization

Chcela by som sa poďakovať vedúcemu mojej diplomovej práce doc. Ing. Romanovi Šenkeříkovi, PhD. za jeho ochotu, ústretovosť a odborné rady.

Ďakujem aj svojmu manželovi, jeho rodičom, mame a deťom za to, že mi vytvorili priaznivé podmienky, pomáhali a mali trpezlivosť nielen počas tvorby tejto práce, ale aj v čase celého môjho štúdia.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČASŤ.....	10
1 EVOLUČNÉ ALGORITMY	11
1.1 VŠEOBECNÝ CYKLUS EVOLUČNÝCH ALGORITMOV	11
1.2 ZÁKLADNÉ POJMY	12
1.2.1 Jedinec.....	12
1.2.2 Populácia	12
1.2.3 Účelová funkcia a vhodnosť	13
1.2.4 Selekcia	14
1.2.5 Genetické operátory	14
1.3 CHARAKTERISTIKA EVOLUČNÝCH ALGORITMOV	15
1.4 OPTIMALIZAČNÉ METÓDY	15
2 ROJOVÁ INTELIGENCIA	17
2.1 SOMA	18
2.2 PARTICLE SWARM OPTIMIZATION	20
2.3 ANT COLONY OPTIMIZATION	22
2.4 FIREFLY ALGORITHM	25
2.5 CUCKOO SEARCH	26
2.6 BAT ALGORITHM.....	28
3 OPTIMALIZÁCIA ROJOM VČIEL.....	31
3.1 VČELY V PRÍRODE	31
3.1.1 Svadobný let.....	33
3.1.2 Hľadanie potravy a včelí tanec.....	33
3.1.3 Správanie sa včiel.....	34
3.2 PREHLAD ALGORITMOV INŠPIROVANÝCH VČELAMI.....	36
3.3 BEES ALGORITHM	37
3.4 IMPROVED BEES ALGORITHM.....	39
3.4.1 Zmenšovanie okolia <i>ngh</i>	39
3.4.2 Opustenie prehľadávaného miesta	39
3.5 ARTIFICIAL BEE COLONY ALGORITHM	40
3.6 HONEY BEE MATING OPTIMIZATION.....	42
3.7 ARTIFICIAL BEEHIVE ALGORITHM	44
3.8 BEE COLONY OPTIMIZATION.....	46
II PRAKTICKÁ ČASŤ	48
4 CIELE PRAKTICKEJ ČASTI PRÁCE	49
5 POPIS A OVLÁDANIE PROGRAMU PRE BEES ALGORITHM.....	50
5.1 PROGRAM <i>BEEAPP</i>	50
5.1.1 Trieda <i>FormBees</i>	51
5.2 PROGRAM <i>BEESALGORITHM</i>	52
5.2.1 Trieda <i>Parameters</i>	52
5.2.2 Trieda <i>Bee</i>	53

5.2.3	Trieda Population	54
5.2.4	Trieda Program.....	55
6	POPIS A OVLÁDANIE PROGRAMU PRE ABC ALGORITMUS.....	56
6.1	PROGRAM <i>ABCAPP</i>	56
6.1.1	Trieda Form1	56
6.2	PROGRAM <i>ABCALGORITHM</i>	57
6.2.1	Trieda Parameters.....	58
6.2.2	Trieda Bee	58
6.2.3	Trieda Population	58
6.2.4	Trieda Program.....	59
7	TESTOVACIE FUNKCIE	60
8	VÝSLEDKY TESTOVANIA	61
8.1	BEES ALGORITHM A IMPROVED BEES ALGORITHM.....	61
8.1.1	Bees Algorithm	62
8.1.2	Improved Bees Algorithm.....	63
8.1.3	Porovnanie BA a IBA	64
8.2	ABC ALGORITHM	67
8.3	PSO ALGORITHM	69
8.4	FIREFLY ALGORITHM	71
8.5	POROVNANIE VŠETKÝCH ALGORITMOV	73
	ZÁVER	76
	ZOZNAM POUŽITEJ LITERATÚRY	77
	ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK.....	81
	ZOZNAM OBRÁZKOV	82
	ZOZNAM PRÍLOH.....	84

ÚVOD

Numerická optimalizácia nachádza uplatnenie v mnohých oblastiach. Spočiatku boli optimalizačné problémy riešené klasickými matematickými metódami. S vývojom informačných technológií a ich uplatnením v takmer všetkých sférach ľudského pôsobenia, sa stretávame s čoraz zložitejšími problémami, na ktoré už nestačí deterministický výpočtový prístup. Približne v polovici minulého storočia vznikli nové optimalizačné techniky, ktoré napodobňujú princípy podľa Darwinovej teórie evolúcie a Mendelovho procesu génovej dedičnosti. Rojové optimalizačné algoritmy predstavujú špeciálnu skupinu evolučných algoritmov, do ktorej patria algoritmy inšpirované správaním sa včiel, mravcov, rýb, vtákov, svetlušiek a mnohých iných.

Cieľom tejto práce bolo vysvetliť a popísať základné princípy rojových optimalizačných algoritmov, naprogramovať vybraný včelí algoritmus a na benchmark IEEE CEC 2014 funkciách urobiť porovnanie s inými rojovými algoritmi.

V teoretickej časti práci sú predstavené základné pojmy a princípy evolučných algoritmov a optimalizačných metód. Ďalej je predstavená rojová inteligencia a s ňou spojené niektoré základné optimalizačné algoritmy. Samostatná kapitola je venovaná optimalizácii včelím rojom, kde je popísané správanie sa biologických včiel, ako aj niektoré vybrané optimalizačné algoritmy.

V praktickej časti sú predstavené samotné naprogramované algoritmy – Bees Algorithm, Improved Bees Algorithm a Artificial Bee Colony. Po nich nasledujú výsledky testov spracované v grafoch a tabuľkách spolu s výsledkami porovnávaných algoritmov Particle Swarm Optimization a Firefly Algorithm.

I. TEORETICKÁ ČASŤ

1 EVOLUČNÉ ALGORITMY

Evolučné algoritmy predstavujú tzv. prehľadavacie populačné algoritmy, ktoré využívajú princípy a techniky vynájdené samotnou prírodou. Evolúcia je proces, pri ktorom dochádza k vývinu jedincov, založený na ich schopnosti prežiť v konkurenčnom boji o jedlo alebo partnera vhodného na párenie. Úspešní jedinci odovzdávajú svoje genetické informácie potomkom a tak vzniká nová generácia, ktorá musí bojovať o prežitie.

Evolučné výpočtové techniky sú numerické algoritmy inšpirované týmto procesom. Pri svojich výpočtoch pracujú s číselnými parametrami a pre dosiahnutie cieľa využívajú spôsoby kríženia, mutácie či selekcie, ktoré vychádzajú z Darwinovej teórie evolúcie a Mendelovho procesu génovej dedičnosti.

Evolučné výpočtové techniky okrem evolučných algoritmov zahŕňajú aj ďalšie postupy, napr. genetické programovanie, evolučný hardware a i. [1]

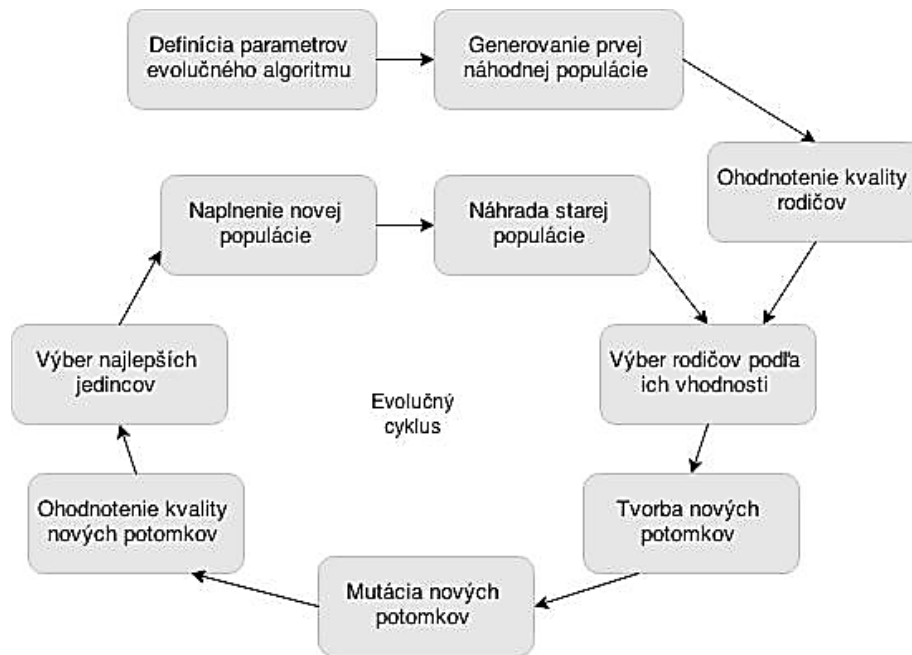
1.1 Všeobecný cyklus evolučných algoritmov

Všeobecný cyklus evolučných algoritmov sa dá popísať podľa obrázku (Obr. 1). Na začiatku sa definujú parametre, s ktorými bude evolučný algoritmus pracovať a ukončovacie podmienky, po ktorých algoritmus ukončí svoj výpočet. Tieto počiatočné parametre a podmienky sa s rôznym typom algoritmu menia.

V ďalších krokoch je vygenerovaná množina bodov, ktorá predstavuje prvú populáciu jedincov a je určená ich vhodnosť pre ďalšie pôsobenie v evolučnom cykle.

Nasleduje cyklus, ktorý sa opakuje až do splnenia určitých podmienok – napr. vypršania časovej lehoty, alebo zadaného maximálneho počtu generácií:

- výber rodičov podľa ich kvality, prípadne iných podmienok
- tvorba nových potomkov
- mutácia potomkov
- určená vhodnosť potomkov
- výber najlepších jedincov z populácie rodičov a potomkov
- zostavenie novej populácie
- nahradenie starej populácie novou



Obr. 1. Všeobecný evolučný cyklus

1.2 Základné pojmy

1.2.1 Jedinec

V evolučných algoritmoch je jedinec reprezentovaný ako číselný vektor s veľkosťou rovnajúcou sa počtu optimalizovaných parametrov účelovej funkcie, teda veľkosti dimenzie. Hodnoty jednotlivých parametrov sú vymedzené vopred určenými hranicami. V tabuľke je príklad jedného jedinca v 5-dimenzionálnom priestore.

Tab. 1. Zobrazenie jedného jedinca v populácii

	Dimenzia 1	Dimenzia 2	Dimenzia 3	Dimenzia 4	Dimenzia 5
Jedinec	-5.2564628	2.2589412	0.1514987	-1.8974166	4.48484131

1.2.2 Populácia

Populácia predstavuje skupinu jedincov, reprezentovanú ako maticu $N \times M$, kde M je počet jedincov a N je veľkosť dimenzie danej účelovej funkcie. Pre evolučné algoritmy je typické, že je počiatočná populácia zvolená náhodne, hľadanie ďalších jedincov je už zámerne ovplyvňované.

Tab. 2. Populácia s viacerými jedincami

	Dimenzia 1	Dimenzia 2	Dimenzia 3	Dimenzia 4	Dimenzia 5
Jedinec 1	5.4981562	-2.1814556	0.2569151	1.5295265	-3.1841562
Jedinec 2	0.2598624	4.1826261	-3.8528982	0.8248223	-2.8384924
Jedinec 3	-1.9825629	2.2812612	1.5742184	4.5484428	-0.6254951

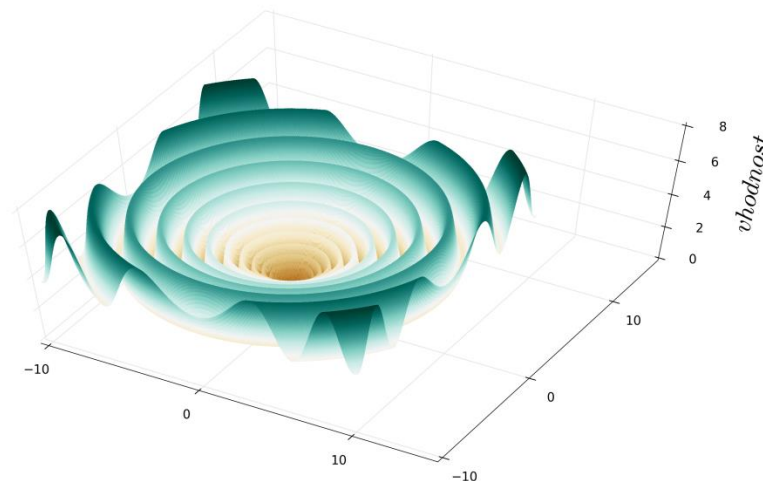
1.2.3 Účelová funkcia a vhodnosť

Pod pojmom *účelová funkcia* rozumieme funkciu, ktorú potrebujeme optimalizovať – teda nájsť jej minimum alebo maximum a budeme ju označovať ako $f(x)$. Ak existuje okolie bodu x_0 také, že platí vzťah:

$$f(x_0) < f(x) \quad (1)$$

pre všetky x z tohto okolia, funkcia má v bode x_0 ostré lokálne minimum. Ak otočíme vo vzťahu (1) nerovnosť, získame maximum funkcie. [1]

Každý jedinec evolučného algoritmu reprezentuje jedno samostatné riešenie danej optimalizovanej funkcie. Kvalita tohto riešenia určuje vhodnosť jedinca podieľať sa na ďalšom vývoji populácie. *Vhodnosť* jedinca označuje, nakoľko kvalitné je konkrétne riešenie, teda kombinácia jeho parametrov. Plocha vhodnosti predstavuje $(N + 1)$ -rozmerný priestor možných riešení, kde N je počet dimenzií (argumentov) optimalizovanej funkcie a posledný rozmer určuje vhodnosť. Na tejto ploche hľadáme globálny extrém, ktorý predstavuje riešenie optimalizačného problému. V prípade multimodálnych funkcií existuje viac rovnakých globálnych extrémov.



Obr. 2. Plocha vhodnosti – Schafferova F7 funkcia

1.2.4 Selekcia

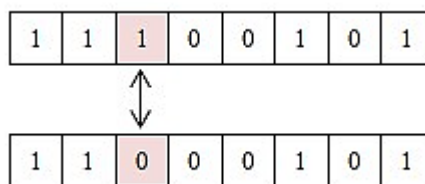
Úlohou selekcie je výber vhodných jedincov – rodičov, z ktorých budú úpravami vznikat' noví jedinci – potomkovia. Pri selekcii sú dôležité dve úlohy – vyberat' jedincov s čo najlepšou vhodnosťou a zároveň udržat' rôznorodosť populácie. Tieto dve úlohy sú navzájom v protiklade, preto hľadáme ich vhodnú kombináciu. Ak by sme vybrali do novej populácie len jedincov s najlepšou vhodnosťou, hrozí riziko rýchlej konverencie algoritmu do lokálneho extrému. Preto evolučné algoritmy pripúšťajú s určitou pravdepodobnosťou aj výber menej vhodného jedinca do ďalšej populácie.

Pre selekciu sa využívajú viaceré metódy, pričom jednotlivé algoritmy uprednostňujú rôzne spôsoby selekcie.

1.2.5 Genetické operátory

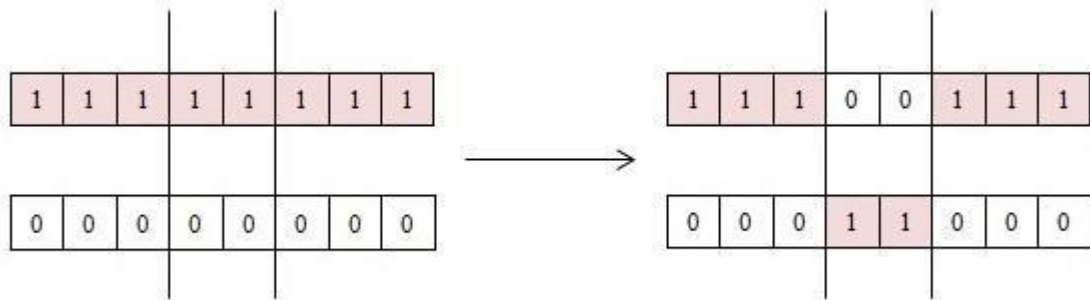
Genetické operátory predstavujú nástroj, pomocou ktorého sa z rodičov vytvárajú potomkovia. Medzi najčastejšie využívané genetické operátory v evolučných algoritmoch patrí *mutácia* a *kríženie*.

Mutácia predstavuje v biologickej analógii asexuálny genetický operátor, pretože potomok má iba jedného rodiča. [2] Výsledkom mutácie je zmena vnútornej štruktúry jedinca, napr. nahradenie jedného parametra jedinca novým parametrom. Zmutovaný potomok by mal obsahovať aspoň nejaký základ z rodiča, v opačnom prípade sa jedná o celkom nového jedinca. Pre určenie rozsahu mutácie sa využívajú pravdepodobnostné metódy.



Obr. 3. Jednobodová mutácia

Kríženie je v biologickej analógii sexuálny genetický operátor, nakoľko má potomok dvoch rodičov. Kríženie je riadené podľa algoritmu a jeho výsledkom je nový potomok, ktorý obsahuje gény oboch rodičov.



Obr. 4. Dvojbodové kríženie

Najčastejšie sú v evolučných algoritmoch využívané oba genetické operátory. Každý algoritmus preferuje svoj spôsob ich využitia – miera mutácie, pravdepodobnosť kríženia, spôsob kríženia a i.

1.3 Charakteristika evolučných algoritmov

Medzi základné rysy evolučných algoritmov patria:

- **Jednoduchosť** – obvykle sa dajú jednoducho naprogramovať
- **Hybridnosť** – pracujú s rôznymi množinami čísiel
- **Používanie dekadických čísiel** – na rozdiel od genetických algoritmov využívajú evolučné algoritmy dekadickú reprezentáciu, ktorá odstraňuje chyby spôsobené binárnym skreslením čísla
- **Rýchlosť** – sú schopné vďaka svojej jednoduchosti podávať rýchle výsledky
- **Vynikajúca schopnosť nájsť extrém** – aj pri patologických funkciách
- **Multimodalita** – schopnosť nájsť viac lokálnych extrémov [1]

1.4 Optimalizačné metódy

Existuje veľké množstvo rôznych algoritmov, ale neexistuje žiadny univerzálny, ktorý by podával jednoznačne najlepšie výsledky pre všetky optimalizačné problémy. Táto skutočnosť sa nazýva „No Free Lunch Teorém“. [1] Každý algoritmus je vhodný na inú triedu problémov. Ak jeden algoritmus rieši optimalizačný problém s vynikajúcimi výsledkami, je celkom možné, že s iným typom problému sa nebude vedieť vysporiadať. Výber toho správneho algoritmu je teda základom úspechu riešenia optimalizačných problémov.

Optimalizačné algoritmy sa využívajú k hľadaniu extrému danej účelovej funkcie pomocou vhodnej kombinácie jej argumentov. [1] Tieto algoritmy sa delia podľa rôznych kritérií, pričom názory na ich rozdelenie sa odlišujú, nakoľko jednotlivé algoritmy môžu svojou podstatou zasahovať do viacerých kategórií. Podľa ich vlastností môžeme algoritmy rozdeliť nasledovne:

- **Enumeratívne** – predstavujú výpočet všetkých možných kombinácií daného problému. Sú vhodné pre malý počet argumentov účelovej funkcie v diskkrétnej forme
- **Deterministické** – tieto algoritmy majú vopred stanovený postup a pri rovnakých počiatočných podmienkach dávajú rovnaký výsledok. Deterministické algoritmy majú rôzne obmedzenia, napr. vyžadujú malý a súvislý prehľadávaný priestor, či unimodálnu účelovú funkciu.
- **Stochastické** – sú založené na náhodnom prehľadávaní a výsledkom je vždy to najlepšie nájdené riešenie. To prináša mnohé nevýhody, napr. pomalosť, či malý počet argumentov účelovej funkcie. Ich úlohou nie je nájsť presný výsledok, sú teda skôr vhodné pre hrubý odhad.
- **Zmiešané** – tieto algoritmy využívajú prednosti deterministických a stochastických algoritmov. Sú vhodné pre riešenie multimodálnych funkcií, schopné prehľadávať veľký priestor možných riešení a nájsť kvalitné riešenia aj pri malom počte ohodnotení účelovej funkcie. Do tejto kategórie patria aj evolučné algoritmy.

2 ROJOVÁ INTELIGENCIA

Rojová inteligencia (Swarm intelligence) je disciplína zaoberajúca sa komplexnými prírodnými systémami, ich vlastnej organizácii a decentralizovaného riadenia. Roj pozostáva z mnohých jedincov, ktorí medzi sebou komunikujú a vzájomnými interakciami formujú inteligentné správanie celého spoločenstva. Príkladom môžu byť spoločenstvá včiel, mravcov, termitov, krdle vtákov a rýb, či bakteriálne kolónie, ale aj ľudské správanie vykazuje v niektorých oblastiach prvky rojovej inteligencie.

Medzi najdôležitejšie vlastnosti rojových inteligentných systémov patrí:

- Sú zložené z veľkého množstva jedincov
- Jedince sú relatívne homogénne – buď identické, alebo zložené z niekoľkých typov
- Jedince sú relatívne jednoduché
- Ich vzájomné interakcie sa zakladajú na jednoduchých pravidlách, využívajú lokálne informácie
- Celkové správanie roja má charakter samo-organizácie

Charakteristickou vlastnosťou rojovej inteligencie je schopnosť systému konať bez prítomnosti lidera, či nejakého vonkajšieho kontrolóra. Samo-organizácia roja zabezpečuje, že pri výpadku jedinca je systém schopný prispôbiť sa a fungovať ďalej. Roj predstavuje živý organizmus, ktorý vykazuje známky inteligencie. Vďaka tomu dokáže riešiť také úlohy, ktoré sú nad rámec síl a schopností jednotlivca.

V roku 1999 Bonabeau s kol. určili štyri základné charakteristické črty samo-organizácie v rojoch:

- **Pozitívna spätná väzba** – je jednoduché správanie, ktoré podporuje vytváranie vhodných štruktúr. Príkladom môže byť vytváranie feromónových ciest mravcami alebo včelie tance.
- **Negatívna spätná väzba** – vyvažuje pozitívnu väzbu a pomáha stabilizovať kolektívny vzor. Je potrebná k zabráneniu presýtenia, ktoré by mohlo nastať napr. pri vyčerpaní zdroja, preplneniu alebo súťaživosti pri zdrojoch potravy a i.

- **Výkyvy** – ako náhodné prehľadávanie, chyby, či vymieňanie úloh medzi jednotlivcami sú životne dôležité pre kreativitu a inováciu. Náhodnosť je často rozhodujúcim faktorom, pretože umožňuje nájsť nové riešenia.
- **Viacnásobné interakcie** pomáhajú jedincom v roji získať informácie od ostatných jedincov, takže sa šíria v celom spoločenstve. [8]

2.1 SOMA

Algoritmus SOMA (Self-Organizing Migration Algorithm) bol vyvinutý Ivanom Zelinkom v roku 1999. Je založený na vektorových operáciách a podobne ako pri evolučných algoritmoch, po jednom cykle je vytvorená nová populácia. Algoritmus SOMA môže byť teda zaradený medzi evolučné algoritmy, ale ako autor hovorí, presnejšie je jeho zaradenie medzi rojové algoritmy. Nová populácia totiž nevzniká v pravom evolučnom zmysle – ako následok genetických operácií a vymieraním starej populácie, ale pohybom jedincov po ploche vhodnosti. Jedince nasledujú svojho vodcu (leadera) a vzájomne sa ovplyvňujú, čím napodobňujú inteligentné správanie sociálnej skupiny - stádo hľadajúce potravu, spoločenstvo včiel, či mravcov. [1]

Algoritmus je citlivý na nastavenie počiatočných podmienok. K svojej činnosti využíva operácie mutácie a kríženia.

Mutácia je v SOMA pomenovaná ako **perturbácia**. Toto pomenovanie vychádza z faktu, že pohyb jedincov po ploche nie je mutovaný, ale náhodne prerušovaný – perturbovaný. Táto operácia sa riadi nastaviteľným parametrom PRT, z ktorého sa vypočíta PRTVector a pomocou náhodne vygenerovaného čísla sa prepočíta vektor pohybu daného jedinca smerom k leaderovi. Tento vektor je vypočítaný pre každý skok jedinca pomocou vzťahu

$$PRTVector_j = \begin{cases} 1 & \text{ak } rnd_j < PRT \\ 0 & \text{v ostatných prípadoch} \end{cases} \quad j = 1, \dots, N \quad (2)$$

Kríženie predstavuje v klasických evolučných algoritmoch tvorbu nového potomka z rodičov. V algoritme SOMA sa jedince pohybujú smerom k leaderovi v diskretných skokoch, v ktorých sú prepočítané ich vhodnosti. Na túto operáciu možno pozeráť ako na operáciu kríženia s tým rozdielom, že počas svojej cesty rodič „vyprodukuje“ väčšie množstvo potomkov, kým pri evolučných algoritmoch sa jedná o tvorbu jedného potomka. Pohyb jedinca sa riadi podľa vzťahu (3).

$$\vec{r} = \vec{r}_0 + \vec{m} t PRT\vec{V}ector \quad (3)$$

kde $t \in \langle 0, po\ Step\ až\ po, PathLength \rangle$, \vec{r} je kandidát na nové riešenie, \vec{r}_0 je pôvodný jedinec, m je rozdiel medzi leaderom a štartovacou pozíciou jedinca, $PRT\vec{V}ector$ je vektor perturbácie.

Základná verzia algoritmu má nasledujúci postup:

1. **Definícia parametrov** – Specimen má rovnaké nastavenie ako pri DE, PathLength – dĺžka cesty (1, 5], Step – krok (0,11; PathLength], PopSize – veľkosť populácie, PRT – perturbácia [0, 1] a Migrácia – ukončovací parameter, určuje počet opakovaní (ekvivalent ku počtu generácií), MinDiv – ukončovací parameter, určuje maximálny rozdiel medzi najlepším a najhorším jedincom v aktívnej populácii.
2. **Tvorba populácie** – vychádza z prototypového vektoru – Specimena.
3. **Migračné kolá** – každému jedincovi je vypočítaná vhodnosť a je určený leader skupiny pre nasledujúce kolo. Pomocou zadaných parametrov a vektorových výpočtov (3) skáču jedince k svojmu leaderovi. Pri každom skoku si jedinec počíta a ukladá najlepšiu vhodnosť. Do ďalšieho kola je potom z týchto hodnôt určený nový leader.
4. **Testovanie ukončovacích parametrov** – v tomto kroku sa testujú podmienky pre ukončenie behu algoritmu – MinDiv a Migrácia.
5. **Vyhodnotenie** - najlepší jedinec z migračných kôl predstavuje najlepšie riešenie daného optimalizovaného problému [1]

Algoritmus SOMA má 3 základné verzie:

- **all-to-one** – migrácia jedincov za jedným vodcom
- **all-to-all** – migrácia všetkých jedincov za vodcom, ktorým sa postupne stávajú všetky jedince v každom migračnom kole
- **all-to-all-adaptive** – podobne ako predchádzajúca verzia, ale po každom putovaní si vždy jedinec prepočíta a nastaví svoju najlepšiu polohu

2.2 Particle Swarm Optimization

Optimalizácia rojom častíc - Particle Swarm Optimization (PSO) je populačný algoritmus určený na riešenie optimalizačných problémov. V roku 1995 ho vyvinuli R. Eberhart a J. Kennedy, ktorí našli inšpiráciu vo vtáčích krdľoch.

Algoritmus pracuje s populáciou podobne ako genetické algoritmy, teda vyhľadáva najlepších jedincov ako možné riešenia, ale na rozdiel od nich nevyužíva evolučné operátory mutácie a kríženia. Jedince sa pohybujú po ploche vhodnosti a nasledujú najlepšie nájdené riešenie.

- 1. Definícia parametrov** – medzi nastaviteľné parametre patrí počet migrácií, veľkosť populácie, počet dimenzií, hranice rýchlosti jedincov a veľkosť učiacich faktorov. V novších verziách algoritmu je ešte nastaviteľný parameter – zotrvačnosť.
- 2. Tvorba populácie** – počiatočná populácia je náhodne vygenerovaná. Každý jedinec má vektor rýchlosti v určených hraniciach a vypočítanú vhodnosť. Jedinec s najlepšou vhodnosťou uloží svoju hodnotu tak, že je dostupná pre celú populáciu - označuje sa ako *gBest*. Zároveň si uchováva každý jedinec svoju osobnú najlepšiu hodnotu účelovej funkcie pod označením *pBest*. Rýchlosť a poloha jedincov je počítaná podľa rovníc (4) a (5).

$$v_d(t+1) = v_d(t) + c_1 \cdot rand \cdot (pBest_{i,d} - x_{i,d}(t)) + c_2 \cdot rand \cdot (gBest_d - x_{i,d}(t)) \quad (4)$$

$$x_{i,d}(t+1) = x_{i,d}(t) + v_d(t+1) \quad (5)$$

kde $v_d(t+1)$ – je rýchlosť jedinca v nasledujúcom kroku, v_d – je rýchlosť jedinca v tomto kroku, $x_{i,d}(t+1)$ – je pozícia jedinca v nasledujúcom kroku, $x_{i,d}$ – je pozícia jedinca v tomto kroku, $pBest_{i,d}$ – je najlepšia pozícia daného jedinca, $gBest_d$ – je najlepšia globálna pozícia, *rand* – náhodné číslo (0,1) a c_1, c_2 – sú učiace faktory.

- 3. Migračné kolá** – pre každého jedinca je vypočítaná nová rýchlosť, ktorá závisí od parametrov *gBest*, *pBest* a predchádzajúcej rýchlosti. Potom sú vypočítané nové pozície jedincov a ich vhodnosť, ktorá je porovnávaná so súkromnou a globálnou najlepšou vhodnosťou.

4. **Testovanie ukončovacích parametrov** – v tomto kroku sa otestujú ukončovacie podmienky algoritmu – teda zadaný počet migrácií.

5. **Vyhodnotenie** – hodnota $gBest$ je najlepším riešením danej optimalizovanej funkcie.

Zotrvačnosť

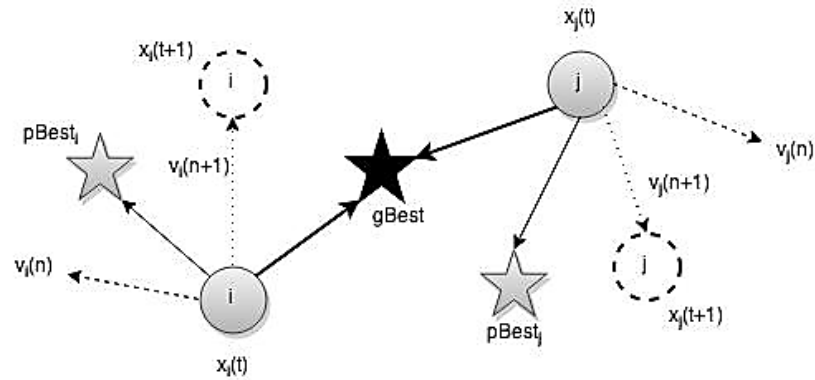
Vylepšená verzia algoritmu používa ďalší parameter – zotrvačnosť w . V základnej verzii bola často vygenerovaná príliš veľká rýchlosť jedinca, čo spôsobilo jeho vzdialenie od najlepšieho miesta. Vďaka parametru zotrvačnosti sa rýchlosť jedinca mení pomalšie. Pre výpočet sa používa vzťah:

$$v_d(t+1) = w \cdot v_d(t) + c_1 \cdot rand \cdot (pBest_{i,d} - x_{i,d}(t)) + c_2 \cdot rand \cdot (gBest_d - x_{i,d}(t)) \quad (6)$$

Základné verzie PSO algoritmu:

- **Susedstvo** (neighbourhood) – jedince sú rozdelené do častí. Algoritmus pomáha zabrániť predčasnej konvergencii do lokálneho extrému.
- **Speciation PSO** – určená pre viacúčelovú optimalizáciu. Časti populácie sa odlišujú – je vytvorená skupina najlepších jedincov, ktorá vedie ostatné skupiny.
- **Niching PSO** – určená pre viacúčelovú optimalizáciu
- **INPSO** – pri optimalizácii využíva nezávislé susedstvá
- **Hybrid PSO** – kombinácia INPSO a evolučného algoritmu GPEA
- **Dynamic Neighbourhood PSO** – dynamické susedstvo [1]

Algoritmus sa vďaka ľahkej implementácii teší veľkej obľúbenosti a je s úspechom používaný v mnohých oblastiach – optimalizácie funkcií, tréning neurónových sietí, fuzzy systémy a i.



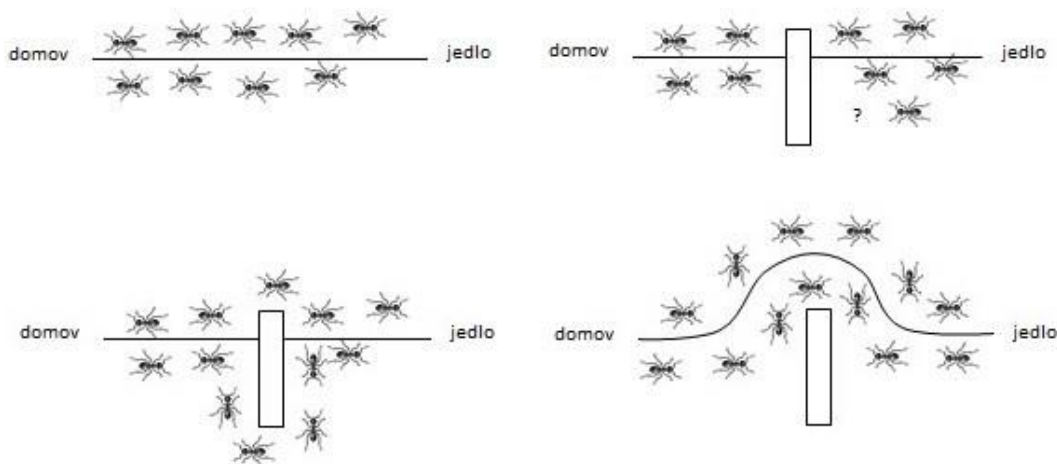
Obr. 5. Pohyb jedincov podľa algoritmu PSO

2.3 Ant Colony Optimization

Optimalizácia pomocou kolónie mravcov - Ant Colony Optimization (ACO) je populačný **metaheuristický** algoritmus určený na riešenie zložitých optimalizačných problémov. V roku 1992 Marco Dorigo vo svojej dizertačnej práci predstavil algoritmus ACO, ktorý je inšpirovaný správaním sa kolónie mravcov pri hľadaní potravy.

V štyridsiatych až päťdesiatych rokoch dvadsiateho storočia francúzsky entomológ Pierre-Paul Grassé prvý raz popísal správanie sa sociálneho hmyzu pomocou feromónov.

V roku 1989 vydal S. Gross s kolektívom výsledky experimentu s argentínskymi mravcami, známeho pod názvom „Experiment dvojitého mosta“. V tomto experimente prepojili hniezdo mravcov dvomi mostmi. Mravce začali preskúmať okolie, až nakoniec našli zdroj potravy. Spočiatku sa pohybovali rôzne, každý mravec si vybral iný most, po nejakom čase sa pohybovali všetky mravce rovnakou – tou najkratšou cestou. [6]



Obr. 6. Mravce pri hľadani potravy

Mravce sa v prírode pohybujú náhodne, rozptýlia sa okolo mraveniska a hľadajú zdroj potravy. Ak niektorý mravec takýto zdroj nájde, s kúskom potravy sa vráti späť do mraveniska. Počas svojej návratovej cesty necháva za sebou pachovú stopu – *feromón*. Podľa nej nájdu zdroj potravy aj ostatné mravce. Najskôr je cesta k zdroju úzka a nestabilná, pretože feromón je prchavá látka. Často býva takých ciest k jednému zdroju viac. Mravce uprednostňujú silnejšiu stopu a nakoľko po kratšej ceste stihnú prejsť rýchlejšie, koncentrácia feromónov je na takejto ceste čoraz výraznejšia. V dôsledku prchavosti feromónu, sa dlhšie cesty postupne vytrácajú, až sa mravce presúvajú po tej najkratšej možnej.

Výsledky Experimentu dvojitého mosta, ktorý vysvetľuje správanie sa mravcov pri zháňaní potravy, bol hlavnou inšpiráciou pre vznik ACO algoritmu.

Hlavnou myšlienkou algoritmu je, že pri každej iterácii sú aktualizované hodnoty feromónov pre všetkých m mravcov. Každý mravec počíta svoje vlastné riešenie. [6]

Algoritmus:

- 1. Definícia parametrov** – na začiatku je potrebné nastaviť feromónové stopy τ_{ij} a atraktivnosť η_{ij} pre všetky hrany (ij) . Medzi užívateľsky nastaviteľné parametre patrí α a β , ktoré majú vplyv na množstvo feromónov a atraktivnosť cesty a ρ je pomer vyparovania feromónov.
- 2. Konštrukcia kolónie mravcov** – pre každého mravca k na aktuálnej pozícii i sa vyberie zo všetkých možných hrán pozícia, do ktorej sa presunie podľa vypočítanej

pravdepodobnosti. Hrana, ktorú prešiel sa pridá do jeho tabu zoznamu, aby nemohol danú trasu opäť použiť.

3. **Aktualizácia feromónov** – po pohybe všetkých mravcov sa aktualizujú feromónové stopy na použitých hranách (ij).
4. **Testovanie ukončovacích parametrov** – ak nie sú splnené ukončovacie podmienky, opakuje sa algoritmus od bodu 2. Ukončovacou podmienkou je vytvorenie kompletnej trasy mravcami na grafe. [6]

Pravdepodobnosť, že sa mravec rozhodne pre cestu je daná vzťahom:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{c_{il} \in N(s^p)} \tau_{ij}^\alpha \cdot \eta_{ij}^\beta} & \text{ak } c_{il} \in N(s^p) \\ 0 & \text{v ostatných prípadoch} \end{cases} \quad (7)$$

kde $N(s^p)$ sú všetky uskutočniteľné zložky, do ktorých patria hrany (ij) kde l je nenavštievané mesto k -tým mravcom, parametre α a β predstavujú relatívnu dôležitosť feromónu verzus atraktívnosť η_{ij} , ktorá je daná vzťahom:

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (8)$$

kde d_{ij} je vzdialenosť medzi mestami i a j .

Feromón na hranách sa počíta podľa vzťahu:

$$\tau_{ij} \leftarrow (1 - \rho) \cdot \tau_{ij} + \sum_{k=1}^m \Delta\tau_{ij}^k \quad (9)$$

kde ρ je rýchlosť vyparovania, m je počet mravcov a $\Delta\tau_{ij}^k$ je množstvo feromónu ležiaceho na hrane (i, j) od mravca k .

Algoritmus má od doby svojho vzniku mnoho verzií, ktoré sa úspešne dajú aplikovať na kombinatorické problémy typu – obchodný cestujúci, či návrh telekomunikačných sietí a routovania.

2.4 Firefly Algorithm

Optimalizácia rojom svetlušiek - Firefly Algorithm (FFA) je populačný metaheuristický algoritmus, ktorý v roku 2007 sformuloval Xin-She Yang na Cambridge univerzite. Tento algoritmus je inšpirovaný rojom svetlušiek.

V prírode existuje približne 2000 druhov svetlušiek a väčšina z nich krátko a rytmicky bliká. Toto svetlo vytvárajú pomocou procesu zvaného bioluminiscencia. Doteraz sú známe dve hlavné funkcie, na ktoré svetlušky využívajú svoje blikajúce svetlo – aby prilákali potenciálnu korisť a aby dali vedieť partnerovi, že sú pripravené na párenie. Okrem toho svetluškám slúži svetlo aj ako varovný signál pre potenciálnych nepriateľov.

Algoritmus sa riadi tromi základnými pravidlami:

1. Všetky svetlušky sú unisex, takže jednotlivé svetlušky sú priťahované k sebe bez ohľadu na pohlavie.
2. Atraktivita je priamo úmerná ich jas, takže je uprednostnená tá, ktorá jasnejšie svieti. Jas a teda aj atraktivita so vzdialenosťou klesá. Pri dvoch rovnako atraktívnych svetluškách je výber urobený náhodne.
3. Jas svetlušiek je priamo úmerný kvalite riešenia účelovej funkcie. [7]

Algoritmus:

1. **Definícia parametrov** – na začiatku sa inicializuje prvá populácia svetlušiek x_i ($i = 1, 2, \dots, n$). Pomocou účelovej funkcie sa určí intenzita jas I_i pre každú svetlušku x_i a definuje sa absorpčný koeficient γ .

$$I(r) = I_s / r^2 \quad (10)$$

kde I_s je intenzita zdroja, intenzita svetla I sa mení so vzdialenosťou r .

2. **Generačný cyklus** – každá svetluška sa v cykle porovnáva s ostatnými svetluškami. Ak je intenzita jas $I_i < I_j$, svetluška i sa posunie k svetluške j podľa vzťahu (11). Zmení sa atraktivita v závislosti od vzdialenosti r prostredníctvom vzťahu (12). Ohodnotia sa nové riešenia a aktualizuje sa pre každú svetlušku intenzita jas. Cyklus sa takto opakuje pre všetky svetlušky.

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_j - x_i) + \alpha \epsilon_i \quad (11)$$

kde x_i je pozícia i -tej svetlušky, α je parameter náhodnosti a ϵ_i je vektor náhodných čísiel z Gaussovho alebo rovnomerného rozdelenia.

$$\beta(r) = \beta_0 e^{-\gamma r^2} \quad (12)$$

kde $\beta(r)$ je atraktivita pri vzdialenosti r , γ je absorpčný koeficient.

3. Aktualizácia výsledkov – svetlušky sa zotriedia a vyberie sa najlepšie riešenie.

4. Testovanie ukončovacích parametrov – algoritmus sa opakuje od bodu 2, pokiaľ nie je vypočítaný potrebný počet generácií. [7]

Algoritmus má od svojho vzniku viacero modifikácií. Úspešne sa používa na riešenie rôznych optimalizačných problémov ako digitálna kompresia, detekcia chýb, plánovanie (Job Shop Scheduling problem), dynamické problémy a i.

2.5 Cuckoo Search

Algoritmus Cuckoo Search (CS) patrí medzi mladšie metaheuristické algoritmy inšpirované prírodou. Algoritmus navrhol v roku 2009 Xin-She Yang a Suash Deb. Bol inšpirovaný agresívnym reprodukčným správaním kukučiek.

Kukučky sú známe svojim parazitickým spôsobom rozmnožovania sa. Samičky znášajú oplodnené vajcia do hniezd iného druhu vtákov, aby ich potomstvo nevedomky vychovali adoptívny rodičia. Ak hostitelia odhalia tento podvod, zvyčajne vajíčka vyhodia z hniezda von, alebo častejšie opustia hniezdo a vystavajú si nové. Niektoré druhy kukučiek vedia farbou a vzorom napodobniť vajíčka hostiteľa, aby znížili pravdepodobnosť, že hostitelia ich vajíčka opustia.

Algoritmus má tieto základné pravidlá:

- Každá kukučka znesie naraz iba jedno vajce a umiestni ho do náhodne vybraného hniezda hostiteľa.
- Hniezda s najlepšou kvalitou vajec predstavujú najlepšie riešenia a sú prenesené do ďalšej generácie.
- Počet dostupných hostiteľských hniezd je fixný. Hostiteľ môže objaviť parazitné vajce s pravdepodobnosťou $p_a \in [0, 1]$, pričom také vajce môže buď z hniezda vyhodiť, alebo hniezdo opustiť.

Každé vajce v hniezde predstavuje riešenie danej optimalizovanej funkcie a vajce kukučky predstavuje nové riešenie. Algoritmus môže byť rozšírený pre komplikovanejšie prípady, keď sa v hniezde nachádza viac vajec.

Autori algoritmu zistili, že algoritmus CS dosahuje lepšie výsledky v kombinácii s Lévyho letom, než s náhodne vybranými krokmi. **Lévyho let** popisuje pohyb živočíchov v prírode. Pre algoritmus je charakteristické striedanie krátkych úsekov s dlhými skokmi. Pomocou Lévyho letu si CS algoritmus vyberá nové riešenia.

Algoritmus:

- 1. Definícia parametrov** – na začiatku je potrebné nastaviť počet host'ujúcich hniezd – n , pravdepodobnosť odhalenia votrelca – p_a , faktor zmeny veľkosti kroku – α , účelovú funkciu a veľkosť jej dimenzie – d .
- 2. Inicializácia** – náhodne je vygenerovaná počiatočná populácia n hniezd na pozíciách $X = [x_1^0, x_2^0, \dots, x_n^0]$, pre každú je vypočítaná hodnota účelovej funkcie $f(x_i^{(t)})$ – vhodnosť riešenia a je uložené najlepšie globálne riešenie g_t^0 .
- 3. Generačný cyklus** – vygeneruje sa nové riešenie (kukučka) $x_i^{(t+1)}$ náhodne pomocou Lévyho letu podľa vzťahu:

$$x_i^{(t+1)} = x_i^t + \alpha \otimes \text{Lévy}(\lambda) \quad (13)$$

Určí sa vhodnosť novej kukučky $f(x_i^{(t+1)})$. Potom sa náhodne vyberie hniezdo x_j spomedzi všetkých n hniezd a jeho vhodnosť sa porovná s vhodnosťou vygenerovanej kukučky. Ak $f(x_i^{(t+1)}) > f(x_j^{(t)})$, tak kukučka nahradí staré riešenie (nakladie vajce do vybraného hniezda).

Ak hostitelia odhalia podvod ($p_a < 0,25$) – hniezdo opustia a postavajú nové (náhodne vygenerované riešenie pomocou Lévyho letu).

- 4. Aktualizácia výsledkov** – určí sa vhodnosť nových hniezd a uloží sa najlepšie riešenie g_t^* .
- 5. Testovanie ukončovacích parametrov** – ak nie je splnená podmienka ukončenia algoritmu – maximálny počet generácií, pokračuje sa od bodu 3. [7]

V základnej verzii sú hodnoty α a p_a fixne určené. Testy však ukázali, že je vhodnejšie tieto hodnoty dynamicky meniť s narastajúcim počtom generácií. Algoritmus sa úspešne používa na riešenie inžinierskych optimalizačných a NP-tiažkých kombinatorických problémov ako – problém obchodného cestujúceho, problém batohu, učenie neurónových sietí a iné.

2.6 Bat Algorithm

Netopierí algoritmus - Bat Algorithm je metaheuristický algoritmus, ktorý zostrojil v roku 2010 Xin-She Yang. Inšpiráciu našiel v echolokačnom správaní netopierov. Algoritmus predstavuje účinný prostriedok na riešenie ťažkých optimalizačných problémov.

Netopiere sú fascinujúce tvory, ktoré dlho priťahujú pozornosť ľudí. Sú to jediné lietajúce cicavce a majú vynikajúcu schopnosť orientovať v prostredí a hľadať svoju korisť pomocou echolokácie. Vydávajú vysokofrekvenčné a veľmi hlasné zvukové signály a sledujú ich vracajúcu sa ozvenu, pričom šírka pásma, frekvencia a hlasitosť je rozdielna pre rôzne druhy.

Algoritmus sa riadi týmito základnými pravidlami:

- Všetky netopiere používajú echolokáciu na zaistenie odstupú a rozoznajú rozdiel medzi korisťou a bariérami.
- Netopiere lietajú náhodne s rýchlosťou v_i na pozícii x_i s konštantnou frekvenciou f_{min} s vlnovou dĺžkou λ a hlasitosťou A_0 na hľadanie potravy. Môžu automaticky upraviť vlnovú dĺžku svojich impulzov a nastaviť mieru impulznej emisie $r \in [0,1]$ v závislosti od blízkosti cieľa.
- Hlasitosť sa mení od veľkej hodnoty A_0 pri hľadaní koristi, po minimálnu hodnotu A_{min} v jej blízkosti.

Netopiere sa pohybujú náhodne s nastavenou frekvenciou f_i , rýchlosťou v_i^t a pozíciou x_i^t v n -dimenzionálnom prehľadávacom priestore, pričom ich hodnoty sa upravujú pomocou vzťahov

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \quad (14)$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i, \quad (15)$$

$$x_i^t = x_i^{t-1} + v_i^t, \quad (16)$$

kde $\beta \in [0,1]$ predstavuje náhodný vektor rovnomerného rozloženia a x_* je aktuálne najlepšie globálne riešenie. [20]

Pre lokálne prehľadávanie sú použité náhodné kroky na generovanie nových riešení pre každého netopiera zo súčasnej populácie pomocou vzťahu

$$x_{new} = x_{old} + \epsilon A^t, \quad (17)$$

kde $\epsilon \in [-1,1]$ je náhodné číslo a A^t je priemerná hlasitosť všetkých netopierov v rovnakom čase. [20]

Všeobecne platí, že hlasitosť klesá a miera impulznej emisie stúpa, keď netopier nájde svoju korisť. Počítajú sa podľa vzťahov

$$A_i^{t+1} = \alpha A_i^t, \quad (18)$$

$$r_i^{t+1} = r_i^0 [1 - \exp(-\gamma t)], \quad (19)$$

kde α a γ sú konštanty. α je podobná ako ochladzovací faktor v simulovanom tavení. Pre každé $0 < \alpha < 1$ a $\gamma > 0$ platí

$$A_i^t \rightarrow 0, \quad r_i^t \rightarrow r_i^0, \quad \text{pre } t \rightarrow \infty \quad (20)$$

Algoritmus

1. **Inicializácia** – na začiatku sa inicializuje populácia netopierov – ich pozície x_i^t a rýchlosť r_i^t . Definuje sa frekvencia f_i pre x_i , pulzová frekvencia r_i a hlasitosť A_i a určí sa vhodnosť prvej populácie.
2. **Generačný cyklus** – pomocou náhodných letov a lokálneho prehľadávania sa vygenerujú nové netopiere a určí sa ich vhodnosť. Aktualizujú sa hodnoty echolokácie.
3. **Aktualizácia výsledkov** – netopiere sa zotriedia a uloží sa najlepšia hodnota x_* .
4. **Testovanie ukončovacích parametrov** – ak nie je splnená podmienka ukončenia algoritmu, generačný cyklus sa opakuje.

Použitie

Netopierí algoritmus a jeho upravené verzie bol úspešne použitý na rôzne optimalizačné problémy ako napr. data mining, plánovacie optimalizačné problémy či inžiniersky dizajn a iné.

3 OPTIMALIZÁCIA ROJOM VČIEL

Včelie algoritmy tvoria ďalšiu skupinu optimalizačných algoritmov, ktoré sú veľmi podobné mravčím algoritmom. Obe patria nepochybne medzi veľmi populárne rojové algoritmy. [7]

Včelie algoritmy sa delia do dvoch skupín – prvá je inšpirovaná párením včiel, druhá zháňaním potravy. Bolo vytvorených veľa verzií algoritmov, medzi ktoré patria napr. Včelí algoritmus - Bees Algorithm (BA), Umelá včelia kolónia - Artificial Bee Colony (ABC), Optimalizácia včelím párením - Honeybee Mating Optimization Algorithm (HBMO) a iné.

3.1 Včely v prírode

Včelí roj patrí medzi najzaujímavejšie a najskúmanejšie roje v prírode. Dokáže sa dynamicky prispôbovať zmenám v prostredí inteligentným kolektívnym spôsobom. Medonosné včely majú fotografickú pamäť, priestorové senzory a navigačný systém, vďaka čomu sa vedú výborne orientovať. Kolektívne rozhodujú o výbere nového hniezda, starajú sa o kráľovnú a potomstvo, zbierajú, uchovávajú a distribuujú med a peľ, zháňajú potravu a komunikujú. Tieto ich vlastnosti boli podnetom pre mnohých výskumníkov k vytváraniu modelov inteligentného správania včiel. [8]

Včela medonosná je sociálny hmyz, ktorý žije vo veľkých spoločenstvách – tzv. včelstvách. Včelstvo je výsledkom evolučného vývoja počas asi 100 miliónov rokov, pričom včela medonosná je na najvyššom stupni. Niektoré druhy včiel žijú samotárskym spôsobom, avšak včela medonosná nedokáže žiť dlhšiu dobu osamote. Včelstvo predstavuje dômyselný systém riadený svojimi zákonmi, v ktorom má každý člen presne vymedzenú úlohu.

Vo včelstve žijú tri druhy včiel – kráľovná matka, robotnice a trúdy. Vo svojom vrcholovom štádiu tvorí včelstvo jedna matka, niekoľko stoviek trúdov a 40.000 – 60.000 robotníc. Vývoj jednej včely prebieha pomocou metamorfózy – od vajíčka, cez larvu, kuklu, až po dospelú včelu.

Matka – včelia kráľovná

Matka je najcennejším členom kráľovskej včelej rodiny. Jej poslaním je zaistiť reprodukciu a teda zachovanie včelstva. Dožíva sa zvyčajne 2-6 rokov, čo je najdlhšie z celého včelstva. Matka sa vyvíja z rovnakého oplodneného vajíčka ako včely robotnice,

avšak podmienky na vývoj má odlišné. Vďaka špeciálnej strave (materská kašička) a bunke (matečník), v ktorej vyrastá, sa jej vyvinú pohlavné orgány. Má oproti robotniciam dlhší zadoček s tisíckami tzv. semien (samčích aj samičích), ktoré kladie podľa potreby - z neoplozených vajícok sa vyvinú trúdy, z oplodnených robotnice. [9]

V jednom včelstve sa nachádza zvyčajne len jedna matka, avšak niekedy môžu koexistovať aj dve matky – stará a mladá - jej dcéra. Toto spoluzitie však trvá len krátky čas, po ktorom ostáva vo včelstve len mladá matka. Tento proces sa nazýva *tichá výmena*.

Ďalší spôsob, pri ktorom včelstvo získa novú matku, je proces *rojenia sa včiel*. Ide o prirodzený spôsob rozmnožovania včelstva, pri ktorom stará matka nakladie do matečníkov vajička a potom s väčšinou včiel opustí svoj domov, aby si našli nový. V pôvodnom roji si robotnice vychovávajú novú matku. Ak sa vyliadne viac matiek, nastávajú kráľovské boje o vládu vo včelstve.

Robotnice

Robotnica je potenciálna matka, ktorá má však nedokonale vyvinuté pohlavné orgány a nemôže sa páriť s trúdmi. Vo včelstve tvoria najpočetnejšiu skupinu a majú rôzne funkcie – kŕmenie a starostlivosť o matku aj o potomstvo, udržiavanie čistoty, stavba plástov a i.

Robotnice sa delia na *mladušky* a *lietavky*. Každá robotnica si za svoj život vyskúša všetky úlohy. Ako mladuška vykonáva práce v úli, neskôr vylieta z hniezda a stáva sa z nej lietavka. Lietavky majú na starosť zber nektáru, peľu, nosenie vody a živicového tmelu – propolisu.

Robotnica má krátky život – prvých 40 dní pracuje v úli, ako lietavka sa dožíva ešte 1 – 3 týždne v závislosti od pridelenej úlohy. Výnimkou sú zimné mesiace, keď včely neopúšťajú úl a dožívajú sa 6-8 mesiacov. [9]

Robotnice majú vyvinuté špeciálne tzv. *Nossonove žľazy*, určené na výrobu špeciálneho feromónu, ktorý je jedinečný pre včelstvo a pomocou neho sa navzájom poznávajú. Ku svojim družkám z rovnakého včelstva sa správajú priateľsky, voči cudzím včelám, predovšetkým matkám, sú agresívne.

Trúdy

Trúd je včelí samček, ktorý sa vyliadne z neoplozeného vajička, ktoré matka kladie do špeciálnych buniek. Jeho hlavným poslaním je spáriť sa s mladou matkou. Včely trúdom využívajú aj na udržiavanie optimálnej teploty v úli.[9] Po dosiahnutí dospelosti sa trúdy

v úli nudia, lebo nepracujú ako robotnice. Za pekného počasia vylietajú von na tzv. *zhromaždenia trúdov*, kde sa schádzajú trúdy z okolia a čakajú, kým priletia mladé matky, aby sa s nimi mohli spáriť. Po spárení sa s matkou trúd zomiera. Ak nemal toľko šťastia, aby sa spáril, zomiera tiež. Buď kvôli prebytku semena jeho telo nevydrží nátlak a praskne, alebo zomrie ako bezdomovec, keď ho na jeseň vykáže včelstvo z úľa von.

3.1.1 Svadobný let

Len čo sa vyliadne vo včelstve nová matka, najneskôr do týždňa ju robotnice vyženú na svadobný let. Kráľovná letí počas párenia ďaleko od svojho včelstva na zhromaždenia trúdov, kde nájde predovšetkým trúdy z iných včelstiev. Počas letu trúdy nasledujú matku a pária sa s ňou vo vzduchu v závislosti od jej rýchlosti a od ich schopností. Spermie sú uložené v matkinom semennom vaku, ktorá po párení odlieta naspäť do úľa a zhromaždenie trúdov viac nevyhľadáva. [9]

Pri svadobnom lete je matka oplodnená viacerými trúdmi (približne 8 – 15). Výskumy ukázali, že táto promiskuita je veľmi prospešná. Vo geneticky rôznorodom včelstve, ktoré založila matka inseminovaná viacerými trúdmi, bolo zaznamenaných o 36% viac kývavých tancov, o 39% viac zásob medu a peľu a prehľadávané územie okolo úľa bolo takmer o 1km vzdialenejšie, čo znamená, že lietavky znášali potravu z väčšej plochy. [10]

3.1.2 Hľadanie potravy a včelí tanec

Včelie tance podrobne preskúmal a vysvetlil profesor K. Von Frisch, za čo dostal v roku 1973 Nobelovu cenu.

Zháňanie potravy patrí medzi najdôležitejšie úlohy v úli. Keď lietavka nájde bohatý zdroj potravy, uloží si z neho do medového vaku a vráti sa späť do úľa, aby oznámila družkám, kde sa tento zdroj nachádza. Ponúkne im z medového vaku a potom im zatancuje a zaspieva. Podľa vzdialenosti zdroja potravy, zvolí typ tanca.

Kruhový tanec

Keď je zdroj potravy blízko úľa (zvyčajne do 50 metrov), včela zatancuje kruhový tanec. Týmto tancom odovzdá družkám informáciu o vzdialenosti zdroja, ale nie o jeho polohe. Potom včely vyletia z úľa a v sústredených kružniciach vyhľadáujú zdroj potravy, ktorý okúsili od tancujúcej včely. [11]

Kosákový tanec

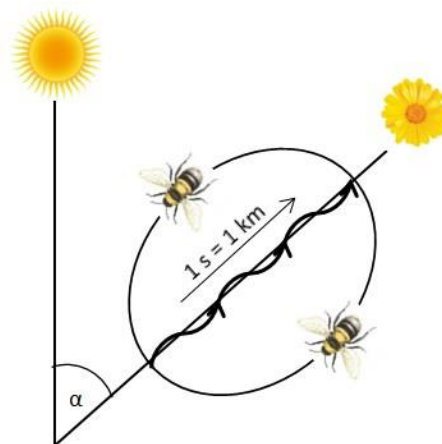
Ak sa nachádza zdroj potravy v strednej vzdialenosti (50 – 150 metrov), tancuje včela tanec, ktorý je v tvare polmesiaca. Je to prechodný tanec medzi kruhovým a natriasavým tancom. [11]

Natriasavý tanec

Tento typ tanca predvádzajú včely pri vzdialenosti väčšej ako 150 metrov. Určuje nielen vzdialenosť, ale aj smer a výdatnosť potravinového zdroja. Tanec má tvar čísla 8, pričom v strede osmičky včela energicky kýva celým telom. V rovnakom čase včela vydáva bzučavý zvuk pomocou trepotania krídiel. Tento zvuk zohráva kľúčovú úlohu, lebo pomocou neho dokážu včely určiť polohu a smer tanca lietavky. [12]

Dĺžka natriasavej časti tanca je spoľahlivým ukazovateľom vzdialenosti potravinového zdroja od úľa. So vzdialenosťou sa predlžuje doba natriasavého tanca, pričom tento vzťah je približne lineárny (Obr. 1). S pribúdajúcou vzdialenosťou klesá atraktivnosť zdroja. [11]

Smer natriasavého tanca udáva zas smer, ktorým sa nachádza potrava. Uhol medzi vertikálnou osou plástu a natriasavou časťou tanca je taký istý, ako uhol medzi slnkom a zdrojom potravy.



Obr. 7. Natriasavý tanec

3.1.3 Správanie sa včiel

V. Tereshko vytvoril model potravinového správania sa včelstva, ktorý rozdelil do troch základných častí:

- **Potravinové zdroje** – pri výbere potravy včela vyhodnocuje niekoľko vlastností - vzdialenosť od úľa, bohatstvo energie, chuť a náročnosť zberu.
- **Zamestnané robotnice** – včela je zamestnaná na konkrétnom zdroji, z ktorého práve ťaží. Čakajúcim včelám v úli predá informácie o tomto zdroji – smer, vzdialenosť a výdatnosť.
- **Nezamestnané robotnice** – sa delia na dve skupiny:
 - vyčkávané včely – čakajú v úli na informácie od zamestnaných včiel
 - zvedovia – náhodne prehládajú okolie

Tereshko definuje dve metódy správania sa včelstva:

- **Nábor na potravinový zdroj** - zamestnané robotnice sa delia o informácie s pravdepodobnosťou priamo úmernou výdatnosti zdroja. Z toho dôvodu je nábor včiel tiež úmerný výdatnosti zdroja. Pri objavení zdroja potravy sa z nezamestnanej robotnice stáva zamestnaná.
- **Opustenie zdroja** – po vyčerpaní zdroja sa zo zamestnanej robotnice stáva nezamestnaná. [13]

3.2 Prehľad algoritmov inšpirovaných včelami

Rok	Autori	Algoritmus	Využitie
1991	Theraulaz s kol.	Wasp Swarm Optimization	Inžinierske problémy
1997	Sato a Hagiwara	Bee System	Vylepšenie genetických algoritmov
2001	Abbass	Mating Bee Optimization (MBO)	Optimalizačné problémy, routovanie, Data mining
2001	Lučić, Teodorović	Bee Colony Optimization (BCO)	Problém obchodného cestujúceho
2003	Jung	Queen-Bee Evolution	Vylepšenie genetických algoritmov, fuzzy systémy
2004	Wedde s kol.	BeeHive	Problém sieťového routovania
2005	Karaboga	Artificial Bee Colony (ABC)	Numerická optimalizácia, neurónové siete
2005	Drias s kol.	Bee Swarm Optimization (BSO)	MAX-SAT problém
2005	Yang	Virtual Bees Algorithm	Inžinierske problémy
2006	Pham s kol.	Bees Algorithm (BA)	Numerická optimalizácia, neurónové siete, inžinierske problémy
2006	Navrat	Bee Hive Model	Webové prehľadávanie
2006	Wedde s kol.	BeeHiveAIS	Routovacie protokoly
2006	Baig, Rashid	Honey Bee Foraging (HBF) algorithm	Numerické problémy
2007	Quijano, Passino	Honey Bee Social Foraging Algorithm	Optimalizačné problémy
2007	Afshar, Marin s kol.	HBMO	Optimalizačné problémy
2009	McCaffrey, Dierking	Simulated Bee Colony	Optimalizačné problémy
2009	Comellas s kol.	Bumblebees Algorithm	Problém farbenia grafu
2012	Maia s kol.	OptBees	Multimodálna optimalizácia
2013	Bitam, Mellouk	Bees Life Algorithm	Inžinierska optimalizácia

3.3 Bees Algorithm

Včelí algoritmus - Bees Algorithm (BA) navrhol v roku 2006 D.T. Pham s kolektívom. BA je optimalizačný algoritmus inšpirovaný správaním sa včiel pri zháňaní potravy.

Počiatočné požiadavky:

n – veľkosť populácie

m – počet vybraných miest z n navštívených miest

e – počet elitných miest z m vybraných miest

nep – počet včiel vybraných k elitným miestam

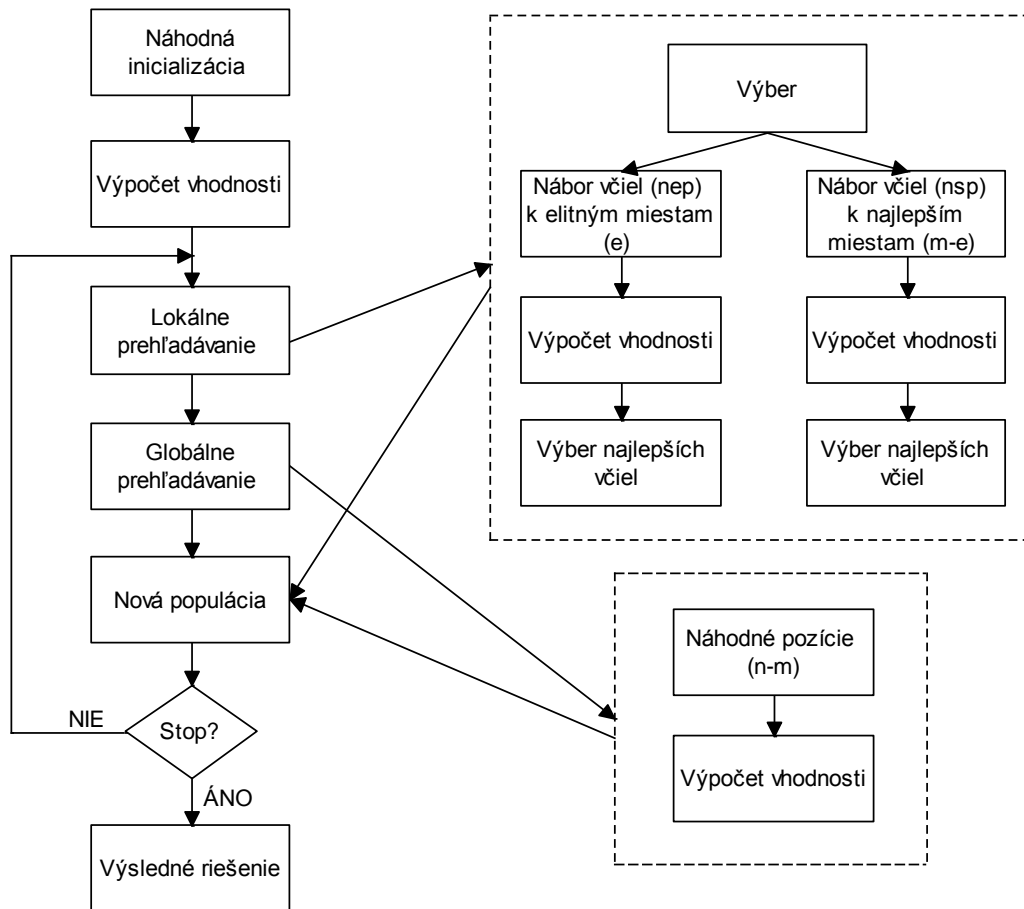
nsp – počet včiel vybraných k zvyšným ($m - e$) miestam

ngh – veľkosť prehl'adávaného okolia

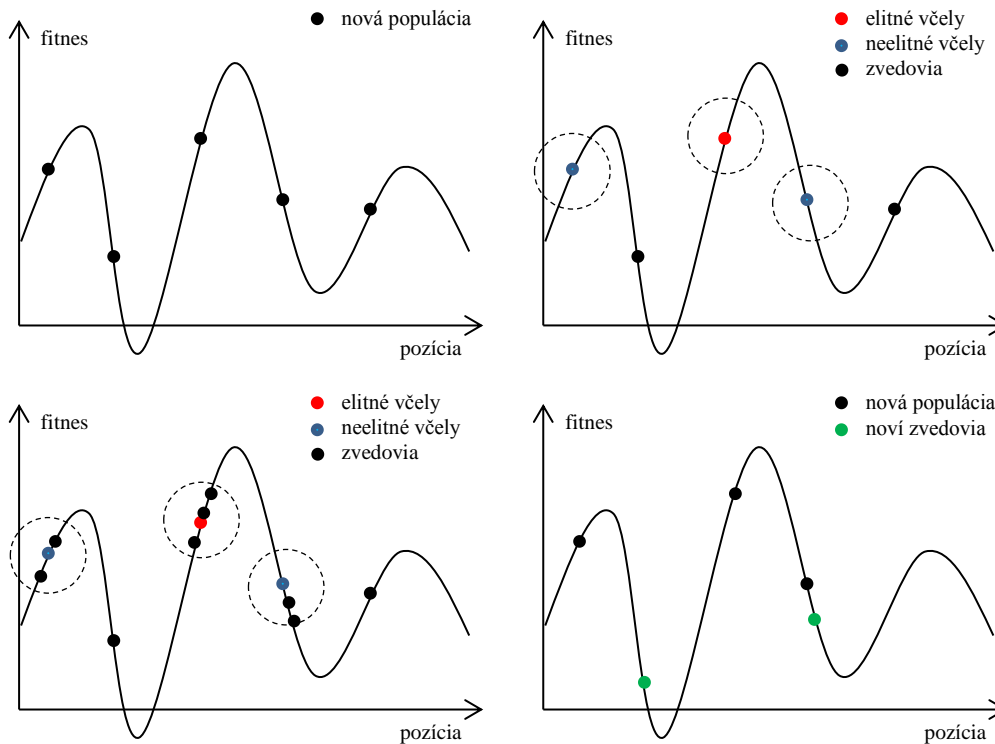
iterations – ukončovacie kritérium, počet iterácií

Pseudokód

1. Inicializuj populáciu n zvedov náhodne na ploche vhodnosti
2. Vypočítaj vhodnosť populácie
3. **Pokiaľ** (nie je splnené ukončovacie kritérium)
4. Vyber m miest na prehl'adávanie okolia (vybrané sú včely s najlepšou vhodnosťou)
5. Pošli včely k vybraným miestam (viac včiel k e elitným, menej k ostatným ($m - e$) vybraným miestam) a vypočítaj ich vhodnosť
6. Vyber najlepšie včely z každého okolia
7. Vygeneruj ostatným včelám ($n - m$) nové pozície a vypočítaj ich vhodnosť
8. **Koniec cyklu** [15]



Obr. 8. Diagram algoritmu BA



Obr. 9. Generačný cyklus BA algoritmu

Algoritmus má viacero upravených verzí:

- Binárny BA (Xu a kol. 2010)
- Distribuovaný BA (Jetvić a kol. 2012)
- Hybridný BA (Shafia a kol. 2011, Lien a Cheng 2012)
- Multi-objektívny BA (Pham a Ghanbarzadeh 2007)
- Vylepšené susedstvo BA (Ahmad 2012) [16]

Použitie

Algoritmus je schopný riešiť funkčné aj kombinatorické problémy a v praxi našiel mnoho využití – napr. optimalizácia výrobného systému, riadenie robotov, optimalizácia mechanických návrhov, dizajn filtrov, plánovanie úloh, učenie neurónových sietí a iné.

3.4 Improved Bees Algorithm

Vylepšený včelí algoritmus Improved Bees Algorithm (IBA) popísali D.T. Pham and M. Castellano svojej práci v roku 2009. [24] Vylepšenia algoritmu sa týkajú lokálneho prehľadávania a zahŕňajú dve stratégie:

- zmenšovanie okolia ngh
- opustenie prehľadávaných miest

3.4.1 Zmenšovanie okolia ngh

V prípade, že lokálne prehľadanie v okolí zdrojov m neprináša žiadne zlepšenie, dochádza k postupnému zmenšovaniu hodnoty ngh . Táto stratégia zabezpečí intenzívnejšie prehľadanie v okolí daného zdroja.

Na začiatku je veľkosť okolia ngh nastavená na vysokú hodnotu a v prípade rovnakých výsledkov sa zmenšuje podľa predpisu:

$$ngh(t + 1) = 0,8 * ngh(t) \quad (21)$$

kde $ngh(t + 1)$ je nasledujúca veľkosť okolia a $ngh(t)$ je aktuálna veľkosť okolia.

3.4.2 Opustenie prehľadávaného miesta

Táto stratégia sa použije v prípade, že zmenšovanie hodnoty ngh dlhodobo neprináša žiadne zlepšenie. V prípade prekročenia vopred určeného limitu stagnácie, predpokladá sa,

že bol dosiahnutý vrchol lokálneho prehľadávania a ďalší pokrok už nie je možný. Prehľadanie v okolí daného zdroja je ukončené a následne je vygenerované nové riešenie.

3.5 Artificial Bee Colony Algorithm

Umelá včelia kolónia - Artificial Bee Colony Algorithm (ABC) navrhol v roku 2005 D. Karaboga. Algoritmus simuluje správanie sa včiel podľa Tereshkovo modelu (3.1.3) a dokáže riešiť multimodálne optimalizačné problémy.

Základné vlastnosti algoritmu:

- Kolónia sa skladá z troch typov včiel: zamestnané včely, vyčkávajúce a zvedovia.
- Polovica kolónie pozostáva zo zamestnaných včiel, druhá polovica z vyčkávajúcich.
- Každému zdroju potravy je pridelená iba jedna včela, teda počet zamestnaných včiel sa rovná počtu zdrojov.
- Zamestnaná včela, ktorá vyčerpá zdroj potravy, sa stáva zvedom.

Poloha zdroja potravy predstavuje jedno riešenie optimalizovaného problému a množstvo nektáru predstavuje vhodnosť tohto riešenia.

V prvom kroku je náhodne vygenerovaná populácia. Počet včiel predstavuje zároveň počet zdrojov $x_{ij} (i = 1, 2, \dots, SN; j = 1, 2, \dots, D)$, kde SN je počet zamestnaných včiel a D je veľkosť dimenzie (počet argumentov optimalizovanej funkcie). Pre výpočet nového zdroja sa používa vzťah:

$$x_{ij} = \min_j + \text{rand}(0,1) * (\max_j - \min_j) \quad (22)$$

kde \min_j a \max_j sú hranice plochy vhodnosti pre každú dimenziu.

Potom sa v cykle opakujú fázy prehľadávania okolia zdrojov najskôr zamestnanými včelami podľa vzťahu:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (23)$$

kde $j \in \{1, 2, \dots, D\}$, $k \in \{1, 2, \dots, SN\}$ pričom $k \neq i$ a ϕ_{ij} je náhodné číslo $[-1, 1]$. Ak je kvalita nového riešenia v_{ij} lepšia než x_{ij} , tieto sa zamenia.

V ďalšej fáze sú k zdrojom jedla vyslané vyčkávajúce včely, ktoré sú pridelené podľa vypočítaných pravdepodobností pre každý zdroj vzťahom

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad (24)$$

kde fit_i je vhodnosť zdroja i vypočítaná úmerne k zdroju nektáru na pozícii i . K výpočtu fitness hodnoty môže byť použitý vzťah:

$$fit_i = \begin{cases} \frac{1}{1 + f_i} & \text{if } f_i \geq 0 \\ 1 + abs(f_i) & \text{if } f_i < 0 \end{cases} \quad (25)$$

kde f_i je hodnota účelovej funkcie na pozícii i . Ak je kvalita nového riešenia v_{ij} lepšia než x_{ij} , tieto sa zamenia.

V poslednej fáze sa kontroluje, či nebol niektorý zdroj vyčerpaný. Ak áno, zo zamestnanej včely sa stane zved, ktorému je náhodne vygenerovaná pozícia podľa vzťahu (22).

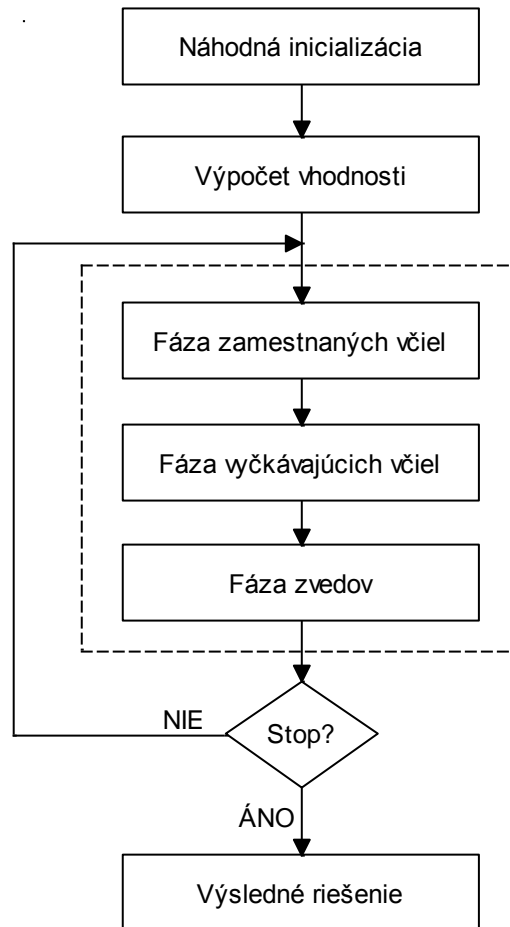
Cyklus sa opakuje, pokiaľ nie je splnená ukončovacia podmienka maximálneho počtu cyklov (*MCN*).

Pseudokód

1. Inicializuj populáciu x_{ij} pomocou vzťahu (22)
2. Urči vhodnosť populácie
3. *cyklus* = 1
4. **Repeat**
5. Vypočítaj nové zdroje v_{ij} pre zamestnané včely pomocou vzťahu (23) a ohodnot' ich.
6. Vyber najlepšie riešenia z v_{ij} a x_{ij}
7. Vypočítaj pravdepodobnostné hodnoty P_{ij} pre zdroje x_{ij}
8. Vypočítaj nové zdroje v_{ij} pre vyčkávajúce včely zvolené v závislosti na pravdepodobnosti P_{ij} a ohodnot' ich.
9. Vyber najlepšie riešenia z v_{ij} a x_{ij}
10. Nájdi vyčerpané zdroje pre zvedov, ak existujú a zameň ich za nové náhodne vypočítané zdroje x_{ij} podľa vzťahu (22)
11. Ulož hodnotu najlepšieho dosiahnutého riešenia

12. $cyklus = cyklus + 1$

13. **Until** $cyklus = MCN$ [14]



Obr. 10. Diagram ABC algoritmu

Použitie

Algoritmus má dobrú schopnosť dostať sa z lokálneho extrému, čo z neho robí výborný nástroj na riešenie numerických optimalizačných problémov. Využíva sa napr. na trénovanie neurónových sietí, spracovanie obrazu, kombinatorické problémy, celočíselné programovanie, inžinierske projektovanie (IIR filtre) a mnoho iných optimalizačných problémov.

3.6 Honey Bee Mating Optimization

Optimalizácia párením včelej kráľovnej - Honey Bee Mating Optimization (HBMO) je metaheuristický algoritmus, ktorý navrhol Hussein A. Abbass v roku 2001. Algoritmus simuluje správanie sa včiel počas párenia. Keďže používa metódy kríženia a mutácie, patrí skôr medzi genetické algoritmy.

Kráľovná začína svadobný let s počiatočnou rýchlosťou a energiou a vracia sa do včelína, keď minie energiu, alebo keď naplní spermotéku. Svadobný let predstavuje pohyb kráľovnej stavovým priestorom, počas ktorého sa pári s trúdmi. Pravdepodobnosť, že sa trúd spári s kráľovnou je vyjadrená vzťahom:

$$Prob(Q, D) = e^{-\frac{\Delta(f)}{S(t)}} \quad (26)$$

kde $Prob(Q, D)$ je pravdepodobnosť, že trúd D pridá semeno do kráľovnej Q spermotéky, $\Delta(f)$ je absolútny rozdiel medzi vhodnosťou D a Q a $S(t)$ je rýchlosť kráľovnej v čase t . Tento vzťah je obdobou simulovaného tavenia. Pravdepodobnosť párenia je vysoká buď na začiatku letu, alebo ak má trúd rovnako dobrú vhodnosť ako kráľovná. [17]

Pri každom pohybe v priestore klesá kráľovnej rýchlosť $S(t)$ a energia $E(t)$ podľa vzťahov:

$$S(t + 1) = \alpha \cdot S(t) \quad (27)$$

$$E(t + 1) = E(t) - \gamma \quad (28)$$

kde koeficient $\alpha \in [0,1]$ a γ predstavuje koeficient znižovania energie pri každom pohybe. Potomkovia sú generovaní procesom spárenia kráľovnej s trúdom - krížením ich genotypu.

V HBMO algoritme majú robotnice za úlohu starať sa o potomstvo. Každá robotnica predstavuje určitú heuristiku, ktorá má zlepšovať genotyp potomstva pomocou lokálneho prehládavania. Pravdepodobnosť, že bude robotnica vybraná, sa určuje pomocou jej vhodnosti. Vhodnosť robotníc narastá po každom úspešnom vylepšení potomka.

V poslednej fáze je stará kráľovná nahradená novou, ktorá predstavuje najlepšie dosiahnuté riešenie.

Pseudokód

1. inicializuj robotnice
2. náhodne vygeneruj kráľovné
3. aplikuj lokálne prehládavanie pomocou robotníc na vylepšenie kráľovien
4. **for** maximálny počet svadobných letov
5. **for each** pre každú kráľovnú
6. inicializuj jej energiu, rýchlosť a pozíciu

7. **while** pokiaľ nie je plná spermotéka a kráľovná má energiu
8. vyber trúda použitím pravdepodobnostného vzťahu (26)
9. **If** ak je trúd vybratý
10. pridaj jeho genotyp do kráľovnej spermotéky
11. **end if**
12. aktualizuj rýchlosť a energiu kráľovnej
13. **end while**
14. **end for each**
15. generuj potomkov pomocou kríženia a mutácie
16. použi robotnice na vylepšenie potomkov
17. aktualizuj vhodnosť robotniciam
18. **while** pokiaľ je najlepší potomok lepší než najhoršia kráľovná
19. nahraď najmenej vhodnú kráľovnú najlepším potomkom
20. odstráň potomka zo zoznamu potomkov
21. **end while**
22. **end for** [18]

Použitie

Algoritmus v svojej prvej podobe použil jeho autor na riešenie tzv. SAT problémov, ktoré patria do množiny NP-úplných problémov. Od svojho vzniku má algoritmus viaceré verzie, napr. Fast Marriage in Honey Bees Optimization (FMHBO), The Honey-Bees Optimization (HBO), Improved Marriage in Honey Bees Optimization (IMBO), ktoré boli použité napr. na riešenie problému rozvrhu, vodnú distribúciu, inžinierske modelovane a mnoho iných optimalizačných problémov.

3.7 Artificial Beehive Algorithm

Optimalizácia pomocou umelého včelstva - Artificial Beehive Algorithm (ABHA) je optimalizačný algoritmus, ktorý zostrojil M. A. Muñoz a kol. v roku 2009.

Algoritmus definuje súbor základných pravidiel, ktoré musia byť dodržané:

- Aktuálnu pozíciu jedinca $\theta(t)$, ktorá reprezentuje jedno riešenie.
- Aktuálnu hodnotu zdroja $J(\theta(t))$, a predchádzajúcu hodnotu $J(\theta(t-1))$
- Tendencia opustenia zdroja p_{ab} v rozmedzí 0 – 1 predstavuje túžbu jedinca opustiť zdroj a je inkrementovaná fixnou hodnotou ρ .
- Motivácia samo navádzania p_h v rozmedzí 0 – 1 predstavuje túžbu jedinca pokračovať vo vyhľadávaní zdroja potravy a je inkrementovaná fixnou hodnotou C .
- Algoritmus definuje 4 stavy, v ktorých sa môže včela nachádzať:

1. stav – *nováčik*, včela je v úli a nemá žiadne informácie o zdrojoch. Môže začať náhodné prehľadávanie, alebo sledovať tanec. Hodnota $\theta(t) = NaN$ (not a number), teda bez čísla.
2. stav – *experimentálny*, včela je v úli, ale má informácie o zdroji. Ak je zdroj kvalitný, môže predať informácie ostatným včelám pomocou tanca označeného ako pravdepodobnostný výber (p_{si}) podľa vzťahu

$$p_{si} = - \left(\frac{1}{\max(J_j) - \min(J_j)} \right) (J_i - \max(J_i)) \quad (29)$$

kde i označuje jedného z j jedincov s dostupným tancom. Ak včela nemá informácie o kvalitnom zdroji, môže začať náhodné prehľadávanie, alebo sledovať tanec.

3. stav – *prehľadávanie*, včela vyletí z úľa a hľadá lepší zdroj jedla než aktuálny. Jej pozícia je upravovaná podľa vzťahu

$$\theta_i(t+1) = \theta_i(t) + SS(i)\psi(t) \quad (30)$$

kde $SS(i)$ je veľkosť kroku v smere $\psi(t)$.

4. stav – *zdroj jedla*, po vyčerpaní hodnoty prehľadávacieho počítadla včela zhodnotí platnosť svojho zdroja jedla. Ak našla lepší zdroj, uloží jeho hodnotu.
- Algoritmus využíva sadu pravdepodobností – súbor náhodných hodnôt pre výpočet pravdepodobnostných výberov populácie. [19]

Použitie

Výsledky testov ukázali, že ABHA algoritmus podáva výkon porovnateľný s podobnými optimalizačnými technikami, v niektorých prípadoch bol jeho výkon lepší. Algoritmus je

vhodný napr. pri identifikácii parametrov v riadení, učení neuronových sietí a fuzzy logiky a pre mnoho iných optimalizačných problémov. [19]

3.8 Bee Colony Optimization

Optimalizácia pomocou kolónie včiel - Bee Colony Optimization (BCO) je metaheuristický prírodou inšpirovaný algoritmus určený na riešenie ťažkých kombinatorických optimalizačných problémov. BCO bol navrhli v roku 2005 D. Teodorović a M. Dell'Orco.

Pseudokód

1. B – počet včiel v úli
2. NC – počet konštruktívnych pohybov počas jedného dopredného prechodu
3. inicializuj všetky včely s prázdny m riešením
4. **for** pre každú včelu vykonaj v nasledujúcom doprednom prechode:
 - a. nastav $k = 1$ // počítadlo konštruktívnych pohybov
 - b. ohodnot' všetky konštruktívne pohyby
 - c. podľa hodnotenia vyber jeden pohyb použitím ruletového výberu
 - d. $k = k + 1$; Ak $k \leq NC$ chod' na krok b.
5. všetky včely sú späť v úli //začína spätný prechod
6. zorad' včely podľa hodnoty ich účelovej funkcie
7. každá včela sa náhodne rozhoduje, či bude pokračovať vo vlastnom prehl'adávaní a stane sa verbovačom, alebo sa stane nasledovníkom, pričom včely s vyššou hodnotou účelovej funkcie majú väčšiu šancu na vlastné prehl'adávanie
8. každému nasledovníkovi vyber nové riešenie spomedzi verbovačov ruletovým výberom
9. ak nie je splnené kritérium ukončenia, chod' na krok 4
10. výsledkom je najlepšie riešenie [21]

Použitie

BCO algoritmus bol upravený do viacerých verzií a úspešne sa využíva na riešenie rôznych optimalizačných problémov, napr. problém plánovania, routovanie, doprava, problém obchodného cestujúceho a iné.

II. PRAKTICKÁ ČASŤ

4 CIELE PRAKTICKEJ ČASTI PRÁCE

Úlohou praktickej časti tejto práce bolo implementovať vybraný swarm algoritmus inšpirovaný správaním sa biologických včiel a porovnať ho s inými algoritmami. K tomuto účelu som si vybrala algoritmy – Bees Algorithm (BA), Improved Bees Algorithm a Artificial Bee Colony Algorithm (ABC). Ich popis sa nachádza v teoretickej časti tejto práce (3.3), (3.4) a (3.5). Ďalším testovanými algoritmami sú Particle Swarm Optimization (PSO), ktorý je popísaný v časti (2.2) a Firefly Algorithm (FFA) popísaný v časti (2.4).

Najskôr nasleduje popis jednotlivých vytvorených programov a ich ovládania. V ďalšej časti sú predstavené testovacie funkcie a na záver sú uvedené výsledky experimentálneho testovania.

Pre implementáciu algoritmov som si vybrala prostredie C#. Programovací jazyk C# je moderný, objektovo orientovaný jazyk od firmy Microsoft, ktorý patrí medzi jazyky typu C. Využíva silné stránky jazykov C++, Java a Visual Basic. Medzi jeho hlavné prednosti patrí typová bezpečnosť, jednoduchosť, robustnosť a univerzálna využiteľnosť, vďaka čomu sa zaradil medzi najpopulárnejšie prostriedky pre vývoj objektovo orientovaných programov.

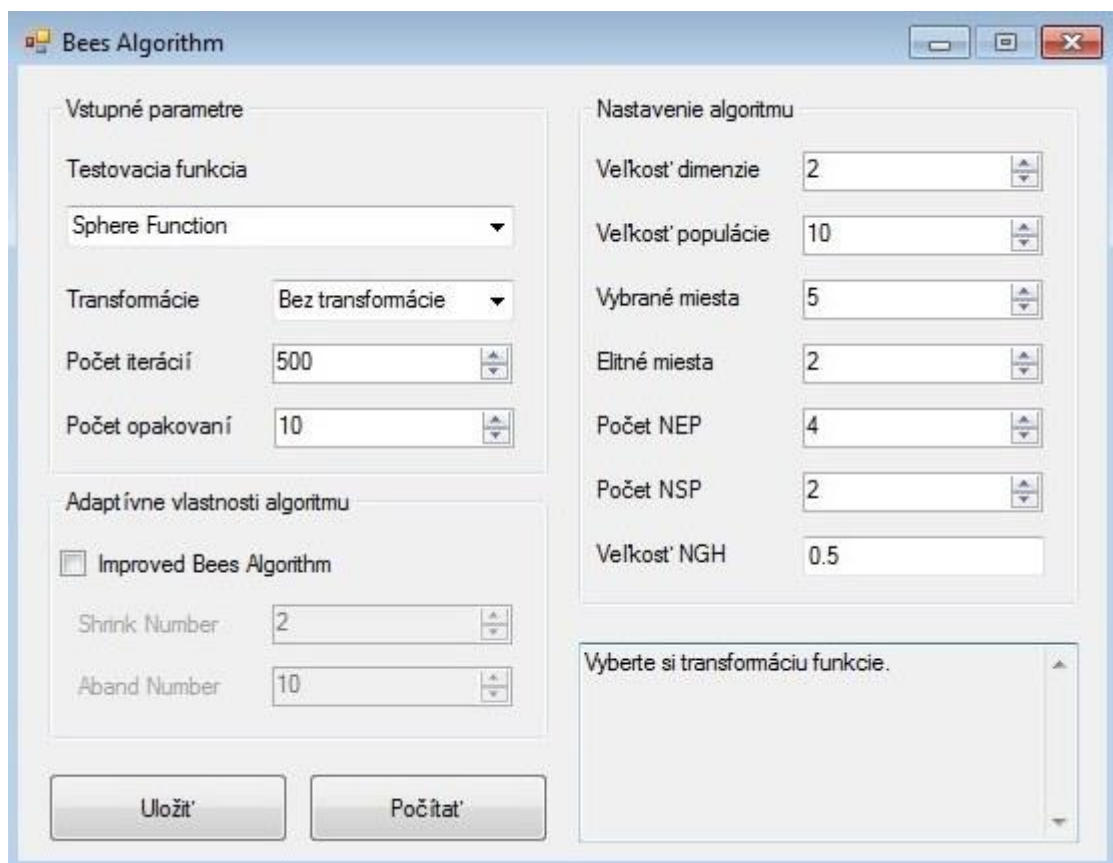
5 POPIS A OVLÁDANIE PROGRAMU PRE BEES ALGORITHM

Implementovaný algoritmus sa skladá zo základného algoritmu Bees Algorithm, ktorý je popísaný v časti 3.3 a je rozšírený o možnosť vylepšenia na Improved Bees Algorithm popísaného v časti 3.4.

Program je rozdelený do dvoch častí – jednu časť predstavuje užívateľské rozhranie pre zadávanie počiatočných parametrov s názvom *BeeApp*, druhú časť tvorí samotný algoritmus pod názvom *BeesAlgorithm*. Obe časti sú implementované v spoločnom prostredí, pričom algoritmická časť je použiteľná aj samostatne.

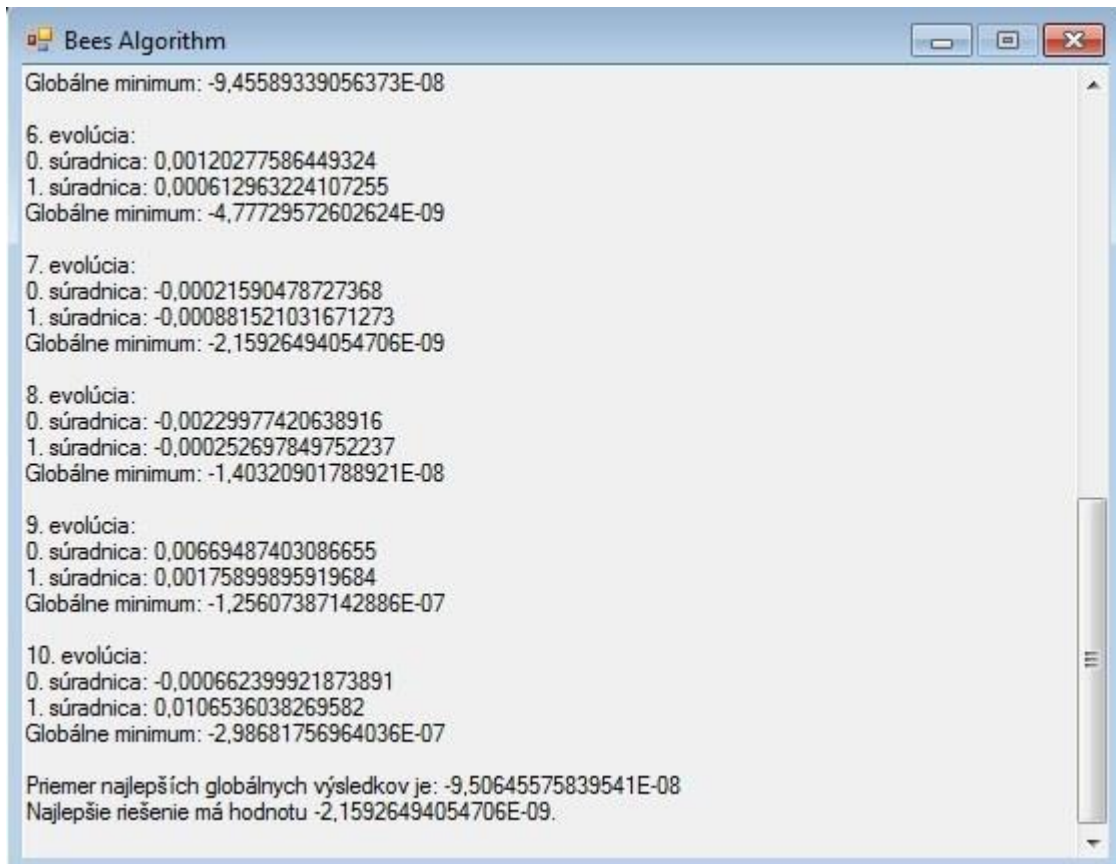
5.1 Program *BeeApp*

BeeApp slúži ako rozhranie, v ktorom si užívateľ môže vybrať počiatočné nastavenia. V prípade voľby vylepšeného algoritmu, budú užívateľovi sprístupnené ďalšie položky nastaviteľných parametrov potrebné pre beh programu.



Obr. 11. Užívateľské rozhranie programu k algoritmu BA

Aplikácia je použiteľná aj na samostatné ukladanie výsledkov, ktoré budú uložené do súboru `config.xml`. V prípade, že si užívateľ zvolí výpočet programu, výsledky budú zobrazené v pomocnom textboxe.



Obr. 12. Ukážka výsledných výpočtov programu

5.1.1 Trieda FormBees

Trieda obsahuje ovládacie prvky formuláru potrebné na nastavenie vstupných parametrov algoritmu. Trieda zároveň slúži na ovládanie programu *BeesAlgorithm*.

Premenné

- `private Parameters parameters` - vstupné parametre
- `private string path` - pomocná premenná na ukladanie cesty súboru

Hlavné metódy

- `private void FormBeesLoad()` - načítanie formulára
- `private void OnSaveClick()` - uloženie parametrov do `config.xml`
- `private void OnStartComputeClick()` - štart výpočtu

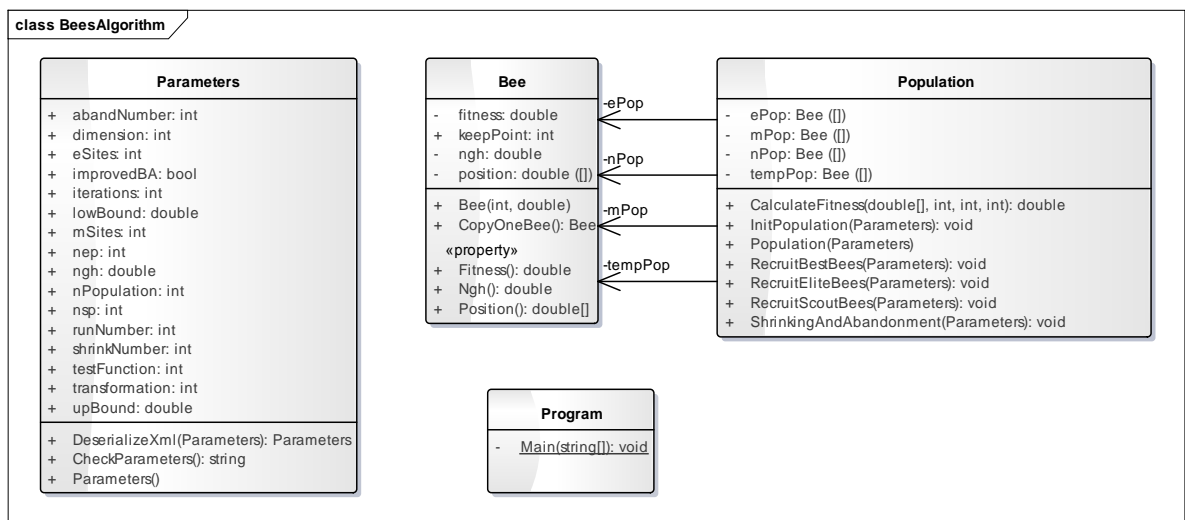
`public void LoadParameters()` - načítanie parametrov z formulára
`private bool CheckParameters()` - overenie správnosti načítaných parametrov
`private void SerializeToXml()` - serializácia vstupov do `config.xml`
`private void StartBeesAlgorithm()` - štart BA algoritmu
`private void PrintMessage()` - výpis výsledkov do `tbInfoBox`
`private void tbInfoBoxDoubleClick()` - zväčšovanie/zmenšovanie `tbInfoBox`
`private void cbImprovedBAClick()` - viditeľnosť položiek pre IBA algoritmus

Trieda obsahuje ďalšie metódy na obsluhu udalostí formulára, pomocou ktorých je zobrazovaná rada v textboxe `tbInfoBox`.

5.2 Program BeesAlgorithm

Program sa dá používať aj samostatne bez užívateľskej aplikácie. Na načítanie vstupných parametrov používa súbor `config.xml`, alebo východiskové hodnoty z triedy `Parameters`.

Základné triedy s premennými a metódami sú zobrazené v diagrame tried (Obr. 13).



Obr. 13. Diagram tried BA algoritmu

5.2.1 Trieda Parameters

V tejto triede sú deklarované a definované všetky vstupné premenné potrebné pre výpočty algoritmu.

Premenné

<code>public int testFunction;</code>	- testovacia funkcia
<code>public int dimension;</code>	- počet dimenzií
<code>public int nPopulation;</code>	- veľkosť populácie
<code>public int mSites;</code>	- počet vybraných najlepších miest
<code>public int eSites</code>	- počet elitných miest
<code>public int nep;</code>	- počet včiel k elitným miestam e
<code>public int nsp;</code>	- počet včiel k $(m - e)$ miestam
<code>public double ngh;</code>	- veľkosť prehľadávaného okolia
<code>public double lowBound;</code>	- dolná hranica prehľadávaného priestoru
<code>public double upBound;</code>	- horná hranica prehľadávaného priestoru
<code>public int runNumber;</code>	- počet evolúcií
<code>public int iterations;</code>	- počet iterácií
<code>public int transformation;</code>	- vybraná transformácia
<code>public bool improvedBA;</code>	- vylepšený algoritmus IBA
<code>public int shrinkNumber;</code>	- hranica pre znižovanie hodnoty ng
<code>public int abandNumber;</code>	- hranica pre opustenie vybraného miesta

Konštruktor

<code>public Parameters()</code>	- konštruktor obsahuje východiskové nastavenia parametrov
----------------------------------	---

Metódy

<code>public Parameters DeserializeXml()</code>	- načítanie vstupov z <code>config.xml</code> súboru
<code>public string CheckParameters()</code>	- kontrola správnosti nastavených parametrov

5.2.2 Trieda Bee

Trieda Bee obsahuje základné premenné a metódy jedného jedinca populácie.

Premenné

<code>private double[] position;</code>	- pozície včely
<code>private double fitness;</code>	- vhodnosť včely
<code>public int keepPoint;</code>	- počítadlo stagnácie
<code>private double ngh;</code>	- veľkosť okolia <i>ngh</i> – využíva sa pri IBA

Metódy

<code>CopyOneBee()</code>	- hĺbková kópia jednej včely
---------------------------	------------------------------

Trieda `Bee` ešte obsahuje vlastnosti `Position`, `Fitness` a `Ngh` na nastavenie týchto hodnôt.

5.2.3 Trieda Population

Táto trieda predstavuje populáciu algoritmu, ktorá pracuje s objektmi triedy `Bee`. Je rozdelená na tri časti – v prvej sú samotné operácie s populáciou, druhá časť obsahuje testovacie funkcie, tretia časť obsahuje metódy na výpis výsledkov a ich zápis do súborov.

Základné premenné

<code>private Bee[] nPop;</code>	- hlavná populácia včiel
<code>private Bee[] ePop;</code>	- elitné včely <i>nep</i>
<code>private Bee[] mPop;</code>	- ostatné včely <i>nsp</i>
<code>private Bee[] tempPop;</code>	- pomocná populácia včiel

Základné metódy

<code>InitPopulation()</code>	- inicializácia prvej populácie
<code>CalculateFitness()</code>	- výpočet vhodnosti
<code>RecruitEliteBees()</code>	- prehľadávanie okolia elitných miest
<code>RecruitBestBees()</code>	- prehľadávanie okolia ostatných miest
<code>ShrinkingAndAbandonment()</code>	- adaptívny algoritmus
<code>RecruitScoutBees()</code>	- globálne prehľadávanie

5.2.4 Třída Program

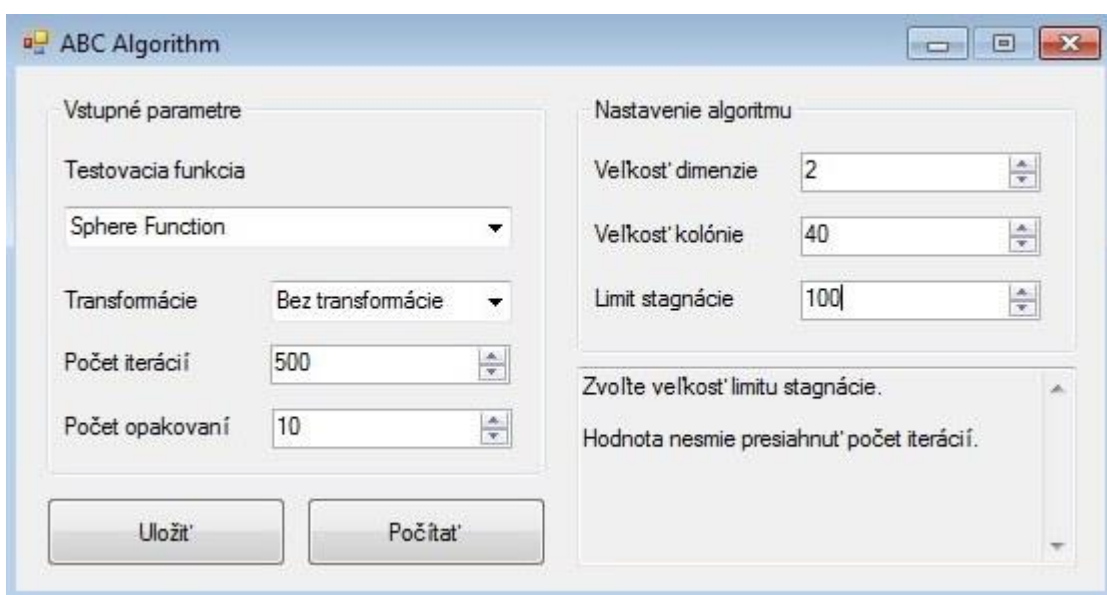
Táto trieda obsahuje hlavný program algoritmu, kde sú vytvárané objekty ostatných tried a operácie s nimi.

6 POPIS A OVLÁDANIE PROGRAMU PRE ABC ALGORITMUS

Tento program je rovnako ako predchádzajúci rozdelený do dvoch častí – prvú časť tvorí užívateľské rozhranie *ABCapp*, druhá časť predstavuje samotný algoritmus s názvom *ABCAlgorithm*.

6.1 Program *ABCapp*

Aplikácia slúži ako užívateľské rozhranie pre nastavenie počiatkových parametrov potrebných pre výpočet algoritmu.



Obr. 14. Užívateľské rozhranie k algoritmu ABC

Rovnako ako predchádzajúca aplikácia umožňuje buď samotné uloženie nastavovaných parametrov do súboru `config.xml`, alebo môže byť použitá na výpočet zadanej funkcie.

6.1.1 Trieda *Form1*

Trieda obsahuje ovládacie prvky formuláru potrebné na nastavenie vstupných parametrov algoritmu. Trieda zároveň slúži na ovládanie programu *ABCAlgorithm*.

Premenné

`private Parameters parameters` - vstupné parametre
`private string path` - pomocná premenná na ukladanie cesty súboru

Hlavné metódy

`private void OnSaveClick()` - uloženie parametrov do `config.xml`

private void OnStartComputeClick() - štart výpočtu

public void LoadParameters() - načítanie parametrov z formulára

private bool CheckParameters() - overenie správnosti načítaných parametrov

private void SerializeToXml() - serializácia vstupov do config.xml

private void StartABCAlgorithm() - štart ABC algoritmu

private void PrintMessage() - výpis výsledkov do tbInfoBox

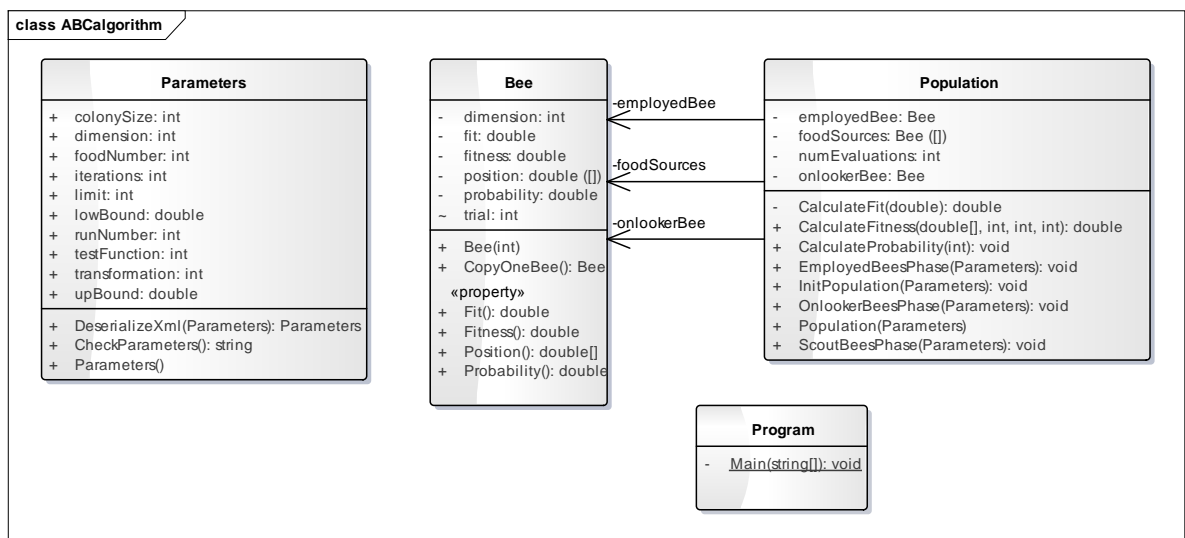
private void tbInfoBoxDoubleClick() - zväčšovanie/zmenšovanie tbInfoBox

Trieda tiež obsahuje ďalšie metódy na obsluhu udalostí formulára, pomocou ktorých je zobrazovaná rada v textboxe tbInfoBox.

6.2 Program *ABCAlgorithm*

Program môže byť použitý s aplikáciou *ABCapp* alebo samostatne. Vstupné parametre sa načítajú buď zo súboru *config.xml*, alebo budú použité východiskové hodnoty z triedy *Parameters*.

Na obrázku (Obr. 15) je znázornený diagram tried ABC algoritmu. V diagrame sú zaznamenané základné triedy s premennými a základnými metódami.



Obr. 15. Diagram tried algoritmu ABC

6.2.1 Trieda Parameters

Táto trieda je obsahujúca vstupné premenné potrebné pre chod algoritmu. Uvedené sú len odlišné premenné oproti BA algoritmu.

Premenné

```
public int colonySize;           - veľkosť populácie
public int foodNumber;          - počet zdrojov jedla/zamestnaných včiel
public int limit;               - hodnota stagnácie
public int cycleNumber;         - počet generácií - konečné kritérium
```

Metódy

```
public Parameters DeserializeXml() - načítanie vstupov z config.xml súboru
public string CheckParameters()   - kontrola správnosti nastavených parametrov
```

6.2.2 Trieda Bee

Trieda Bee predstavuje jedného jedinca populácie a obsahuje premenné, metódy a vlastnosti podobné ako trieda Bee v predchádzajúcom algoritme.

Základné premenné

```
private double[] position;       - pozícia včely
private double functionValue;    - vhodnosť včely
private double fitness;          - hodnota potrebná pre výpočet pravdepodobnosti
internal int trial;              - počítadlo stagnácie
private double probability;      - pravdepodobnostná hodnota
```

6.2.3 Trieda Population

Tu budú predstavené iba odlišné metódy oproti predchádzajúcemu algoritmu.

Premenné

```
private Bee[] foodSources;       - zdroje jedla/populácia včiel
private Bee employedBee;        - pomocná premenná – zamestnaná včela
private Bee onlookerBee;        - pomocná premenná - zved
```

Metódy

public double CalculateFitness() - výpočet vhodnosti jedinca

private double CalculateFit() - výpočet normalizovaných fitness hodnôt

public void EmployedBeesPhase() - fáza zamestnaných včiel

public void CalculateProbability() - výpočet pravdepodobnosti

public void OnlookerBeesPhase() - fáza vyčkávajúcich včiel

public void ScoutBeesPhase() - fáza zvedov

6.2.4 Trieda Program

Trieda rovnako ako pri predchádzajúcom programe obsahuje hlavný program algoritmu potrebný pre vytváranie objektov ostatných tried a operácie s nimi.

7 TESTOVACIE FUNKCIE

Algoritmy boli otestované na sade vybraných testovacích funkcií definovaných v IEEE CEC 2014. Ich prehľad je uvedený v Tab. 3. [23] Podrobnosti o testovacích funkciách sú v prílohe P I.

Tab. 3. Prehľad IEEE CEC'14 testovacích funkcií

	Číslo	Funkcia	$F_i^* = F_i(\mathbf{x}^*)$
Unimodálne funkcie	1	Eliptická funkcia s rotáciou	100
	2	Bent Cigar funkcia s rotáciou	200
	3	Discus funkcia s rotáciou	300
Jednoduché multimodálne funkcie	4	Rosenbrockova funkcia s posunom a rotáciou	400
	5	Ackleyho funkcia s posunom a rotáciou	500
	6	Weierstrassova funkcia s posunom a rotáciou	600
	7	Griewankova funkcia s posunom a rotáciou	700
	8	Rastriginova funkcia s posunom	800
	9	Rastriginova funkcia s posunom a rotáciou	900
	10	Schwefelova funkcia s posunom	1000
	11	Schwefelova funkcia s posunom a rotáciou	1100
	12	Katsuura funkcia s posunom a rotáciou	1200
	13	HappyCat funkcia s posunom a rotáciou	1300
	14	HGBat funkcia s posunom a rotáciou	1400
	15	Rozšírená Griewankova plus Rosenbrockova funkcia s posunom a rotáciou	1500
	16	Rozšírená Scafferova F6 funkcia s posunom a rotáciou	1600
Hybridné funkcie	17	Hybridná funkcia 1 ($N = 3$)	1700
	18	Hybridná funkcia 2 ($N = 3$)	1800
	19	Hybridná funkcia 3 ($N = 4$)	1900
	20	Hybridná funkcia 4 ($N = 4$)	2000
	21	Hybridná funkcia 5 ($N = 5$)	2100
	22	Hybridná funkcia 6 ($N = 5$)	2200
Kompozitné funkcie	23	Kompozitná funkcia 1 ($N = 5$)	2300
	24	Kompozitná funkcia 2 ($N = 3$)	2400
	25	Kompozitná funkcia 3 ($N = 3$)	2500
	26	Kompozitná funkcia 4 ($N = 5$)	2600
	27	Kompozitná funkcia 5 ($N = 5$)	2700
	28	Kompozitná funkcia 6 ($N = 5$)	2800
	29	Kompozitná funkcia 7 ($N = 3$)	2900
	30	Kompozitná funkcia 8 ($N = 3$)	3000

8 VÝSLEDKY TESTOVANIA

Táto časť práce sa venuje samotným výsledkom testovania. Najskôr budú predstavené výsledky pre jednotlivé algoritmy zvlášť, potom budú výsledky algoritmov vzájomne porovnané.

Každý algoritmus bol pre jednotlivé funkcie spustený 30-krát, aby bol zaistený čo najlepší výsledok výpočtu. Kvalita jednotlivých algoritmov je porovnaná podľa počtu ohodnotení účelovej funkcie. Vzhľadom na to, že počas jednej iterácie dochádza pri jednotlivých algoritmoch k rôznemu počtu ohodnotení účelovej funkcie, podáva tento prístup najobjektívnejší obraz o kvalite algoritmov. Počet ohodnotení účelovej funkcie je približne 50000 pre každý algoritmus. Funkcie boli testované pre 2 a 10 dimenzií.

8.1 Bees Algorithm a Improved Bees Algorithm

Pri oboch algoritmoch BA aj IBA boli nastavené vstupné parametre na rovnaké hodnoty:

Veľkosť populácie:	30
Počet vybraných m miest:	10
Počet vybraných e miest:	3
Počet včiel k e miestam:	10
Počet včiel k $(m - e)$ miestam:	3
Hranice prehľadávaného priestoru:	[-100; 100]
Počet generácií:	800
Ďalšie nastavenia pre algoritmus IBA:	
Hranica pre zmenšovanie ngh :	150
Hranica pre opustenie miesta:	300

Parametre boli zvolené podľa osobných testov a zdrojov literatúry, hodnoty pre vylepšený algoritmus IBA boli určené z vlastného niekoľkonásobného testovania.

8.1.1 Bees Algorithm

V tabuľke (Tab. 4) sú zobrazené výsledky testovania algoritmu Bees Algorithmus. Hybridné funkcie nie sú definované pre 2 dimenzie. Z výsledkov je zrejmé, že pri malom počte dimenzií dokáže algoritmus nájsť extrém, zatiaľ čo pri vyššom počte dimenzií už nie sú výsledky natoľko spoľahlivé. Najväčšie problémy má algoritmus pri prvých troch funkciách – Eliptická, Bent Cigar a Discus funkcia s rotáciou.

Tab. 4. Výsledky BA algoritmu

Funkcia	Najlepšie riešenie	Priemer najlepšíh	Priemer mediánov	Najlepšie riešenie	Priemer najlepšíh	Priemer mediánov
	2D			10D		
1	-0,01972976	-5,97857557	-777,908866	-44873,3356	-176180,166	-1037955,76
2	-0,00704164	-20,2046285	-694,734669	-143073,755	-336792,555	-621058,626
3	-0,02840892	-8,14040637	-790,06638	-3000,91986	-7972,72327	-21992,6228
4	-1,48E-08	-1,80E-07	-2,05E-06	-0,09916793	-2,1828991	-6,99617224
5	-0,00199006	-0,01278132	-0,06372456	-18,9909865	-20,3325269	-20,5884965
6	-0,00369714	-0,01091402	-0,05961196	-5,03582288	-6,65004266	-8,99037357
7	-5,64E-05	-0,00059075	-0,00778892	-0,43938596	-0,58832162	-0,83829383
8	-3,57E-07	-3,99E-05	-0,69687126	-5,25604919	-25,8759405	-50,3446231
9	-4,85E-07	-4,52E-05	-0,73123644	-7,21970048	-24,3629627	-54,5163155
10	-1,75E-05	-0,07453955	-0,40087352	-321,018327	-700,002173	-1223,79863
11	-1,62E-05	-0,00085401	-0,35995387	-256,265197	-795,297329	-1329,18481
12	-0,01123777	-0,05610501	-0,19727464	-0,21802974	-0,30433904	-0,67571352
13	-0,02537196	-0,05219954	-0,13775368	-0,13680064	-0,21985387	-0,48321368
14	-0,00243247	-0,00754099	-0,0559216	-0,062185	-0,16505993	-0,3367323
15	-2,15E-13	-0,00231839	-0,04934378	-4,16185574	-5,9506917	-10,8821001
16	-5,57E-05	-0,01816621	-0,06161788	-2,37217713	-3,16011197	-3,74042608
17				-898,467662	-2066,33545	-45425,969
18				-126,526798	-417,636719	-3445,54618
19				-2,02899497	-3,53800657	-6,26202925
20				-64,878546	-296,033488	-4949,7175
21				-420,082711	-990,046118	-5089,72745
22				-25,0843222	-28,2733268	-55,6050395
23	-0,00987675	-0,03633636	-0,1641451	-1,15545601	-239,685802	-256,74259
24	-0,00010136	-0,00183898	-3,50028502	-234,96929	-257,670231	-303,808995
25	-0,00125728	-0,00688148	-0,04383563	-183,349547	-195,706223	-201,955616
26	-2,76E-06	-0,00833826	-0,10276312	-106,048535	-106,17796	-106,373153
27	-0,03401634	-0,11378477	-0,32739396	-1,3434693	-2,03108769	-4,01682165
28	-0,00335969	-0,04316291	-0,15750295	-240,828634	-398,683673	-486,854928
29				-205,099283	-307,50655	-1287,51209
30				-285,036049	-499,302799	-1097,78832

8.1.2 Improved Bees Algorithm

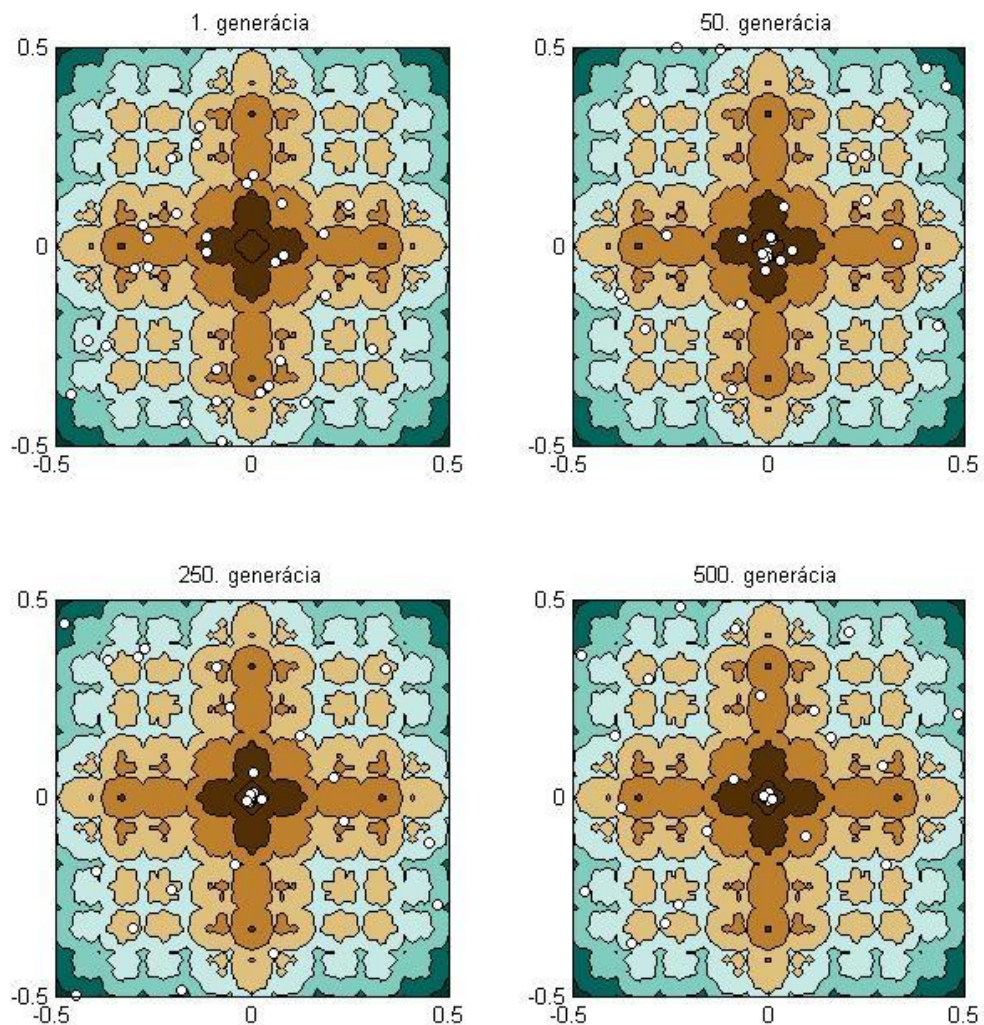
Výsledky testovania sú zobrazené v tabuľke (Tab. 5). Algoritmus bol schopný spoľahlivo nájsť extrémny funkcií pri 2 dimenziách. Pri 10 dimenziách už výsledky nie sú natoľko presné a algoritmus má problém s extrémami pri Eliptickej a Discus funkcii s rotáciou.

Tab. 5. Výsledky IBA algoritmu

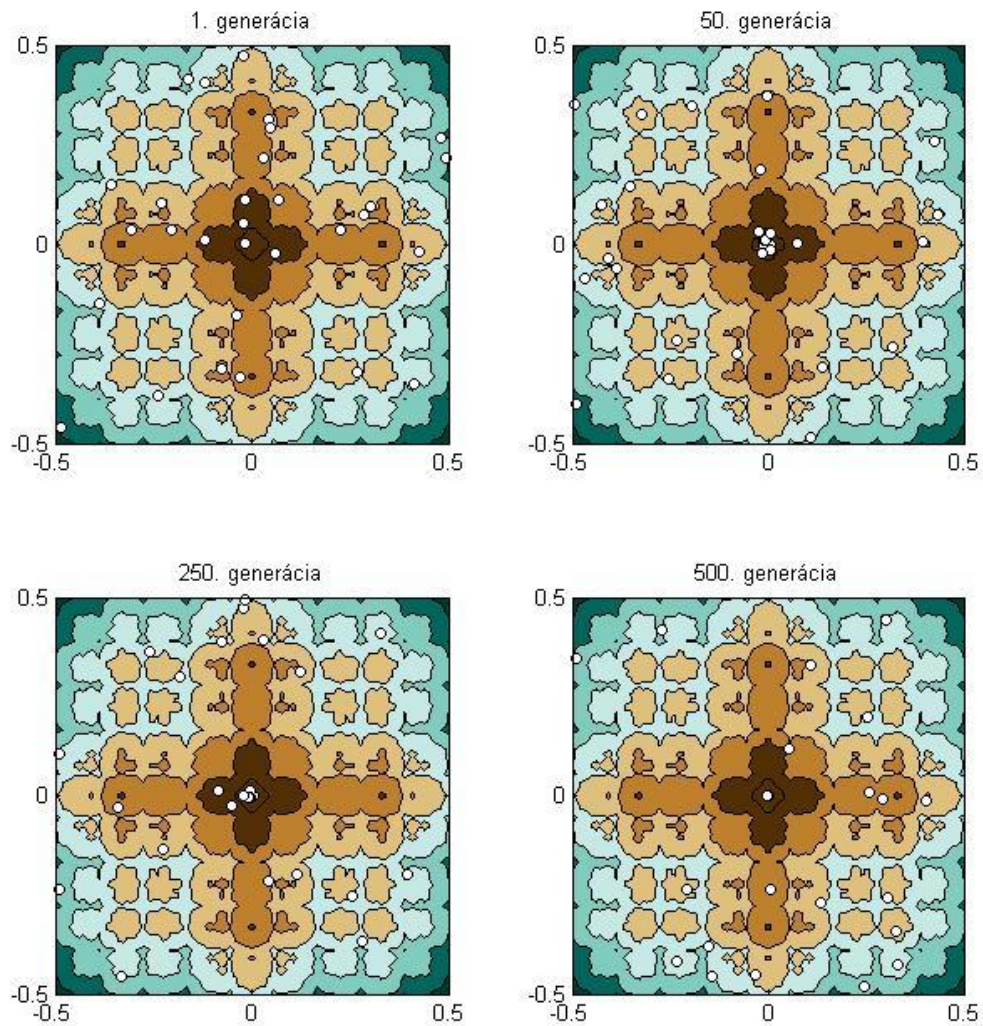
Funkcia	Najlepšie riešenie	Priemer najlepších	Priemer mediánov	Najlepšie riešenie	Priemer najlepších	Priemer mediánov
	2D			10D		
1	-0,00175074	-10,6021641	-654,06481	-15074,7523	-192939,526	-940306,668
2	-0,0003592	-12,5689084	-642,651848	-2,25890142	-350,588733	-4784,66799
3	-0,002201	-11,9749901	-759,338689	-4311,85937	-10575,4386	-23363,3013
4	-4,45E-16	-3,73E-12	-4,11E-09	-0,03133353	-1,7256982	-6,85268297
5	-2,95E-07	-2,56E-05	-0,00057393	-19,8055561	-19,9921516	-20,0000001
6	-4,45E-05	-0,00041769	-0,01535354	-5,08589521	-6,49443776	-8,85387989
7	-2,15E-11	-5,61E-07	-0,00898651	-0,01750279	-0,06159618	-0,18051136
8	-9,41E-14	-1,94E-07	-0,69814698	-6,97966885	-24,3450178	-51,7376249
9	-1,28E-12	-5,02E-08	-0,69716829	-13,9294263	-25,6229791	-52,9526605
10	-2,86E-11	-0,01186569	-0,34372743	-122,587429	-652,838514	-1216,30665
11	-5,12E-12	-2,00E-06	-0,32267133	-471,168096	-757,497392	-1385,22209
12	-1,89E-06	-0,00012669	-0,00243557	-0,00686367	-0,06069672	-0,38177283
13	-0,00120664	-0,00799681	-0,07302881	-0,07973271	-0,15358491	-0,40744448
14	-2,57E-05	-0,00223372	-0,04692734	-0,10932831	-0,16683687	-0,31592497
15	0	-0,00131722	-0,04518319	-2,82024455	-5,29546103	-11,9070649
16	-2,70E-09	-0,01426298	-0,05244348	-1,98107714	-3,10638546	-3,74267745
17				-897,985251	-2334,74436	-48604,86
18				-66,9979743	-223,324884	-3453,8796
19				-2,65180415	-3,8509898	-7,1518098
20				-33,0410395	-535,051616	-5931,90524
21				-359,867646	-1064,93417	-4961,22734
22				-21,5889862	-25,9383893	-54,6971449
23	-2,07E-06	-5,63E-05	-0,00174228	-0,00584745	-213,938649	-256,729332
24	-1,29E-07	-1,69E-05	-3,26855029	-235,467394	-255,259351	-298,393298
25	-9,73E-07	-2,68E-05	-0,00458906	-190,529845	-197,230396	-201,741075
26	-1,42E-09	-0,00793668	-0,10028137	-106,101692	-106,189841	-106,382579
27	-0,00123559	-0,00765783	-0,06734176	-0,91431616	-1,98094203	-3,87452154
28	-2,94E-06	-5,70E-05	-0,00181834	-381,675844	-415,464056	-485,463268
29				-208,026225	-320,305969	-1333,09824
30				-269,697849	-481,421503	-999,557865

8.1.3 Porovnanie BA a IBA

Na obrázkoch (Obr. 1) a (Obr. 17) je zobrazený evolučný vývoj populácie pre oba algoritmy pomocou kontúrovacieho grafu pre Weierstrassovu funkciu. Hľadaný extrém sa nachádza uprostred grafu. Na obrázkoch vidieť, že kým algoritmus IBA skonvergoval do extrému pri 500. generácii, algoritmus BA sa k nemu len približuje. Rôznorodosť zvyšnej populácie je spôsobená samotnou podstatou algoritmu – z celkovej populácie n jedincov sa pracuje s vybranými m jedincami a zvyšok tvoria zvedovia, čo sú náhodne vygenerované riešenia na ploche vhodnosti.



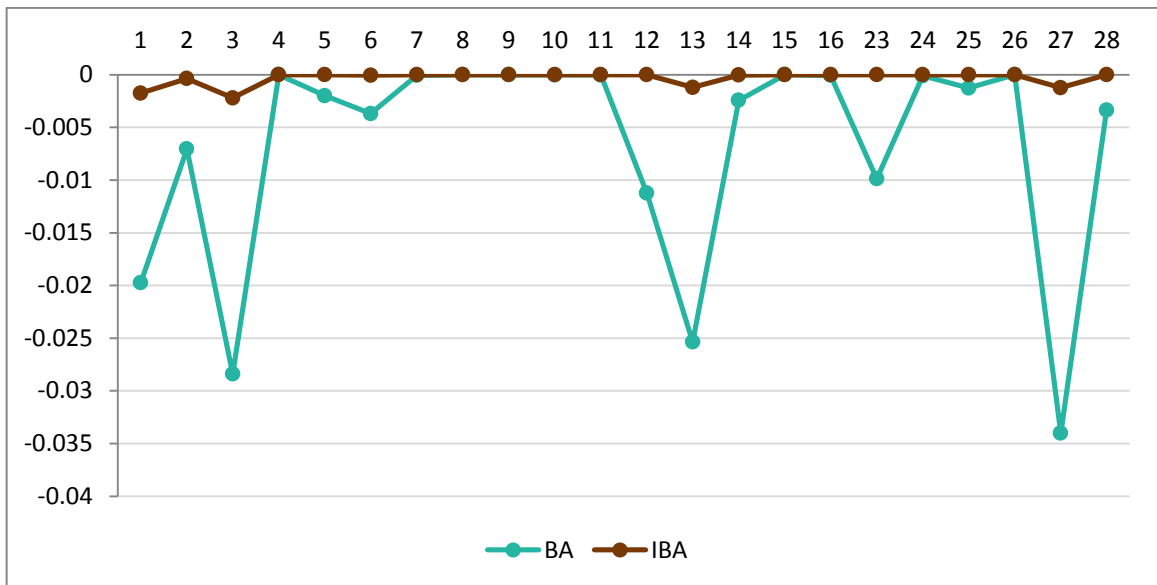
Obr. 16. Evolučný vývoj populácie BA pre Weierstrassovu funkciu



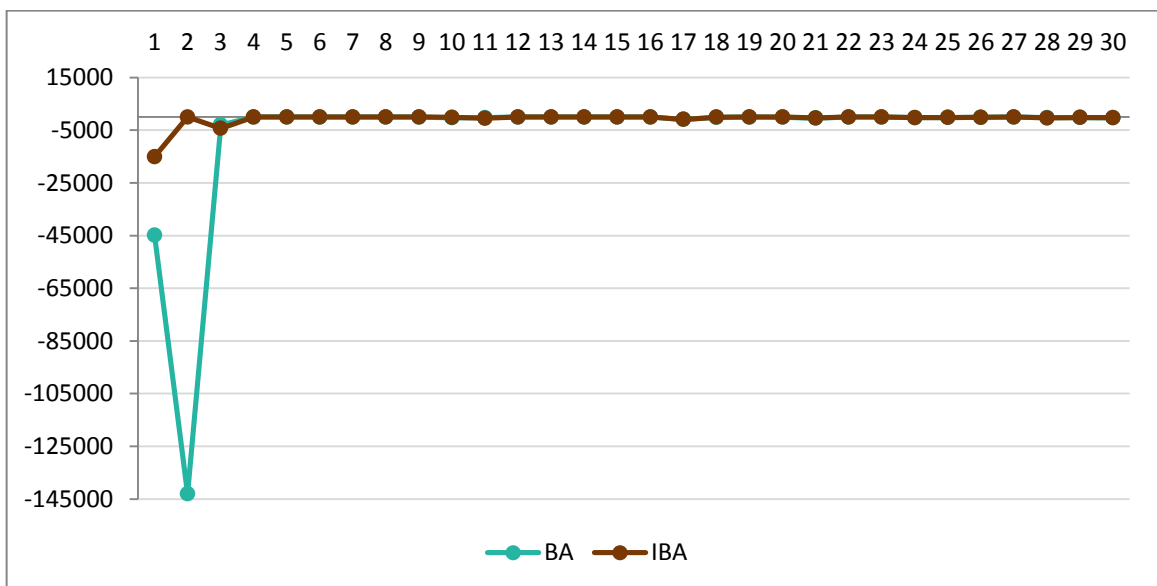
Obr. 17. Evolučný vývoj populácie IBA pre Weierstrassovu funkciu

Na obrázkoch (Obr. 18) a (Obr. 19) sú zobrazené grafy najlepších riešení zo všetkých evolúcií pre všetky testovacie funkcie. Prvý graf zobrazuje výsledky pre 2 dimenzie, preto sú opäť vynechané hybridné funkcie a kompozitné funkcie, ktoré s nimi pracujú. Z grafu je zrejmé, že oba algoritmy našli hľadané extrémny, ale vylepšený algoritmus IBA má presnejšie hodnoty.

Na druhom grafe sú zobrazené výsledky pre 10 dimenzií. Tento graf poskytuje skôr makro pohľad vďaka vysokým hodnotám druhej funkcie, čo spôsobilo, že sa hodnoty oboch algoritmov pri ostatných funkciách prekrývajú. Podľa grafu (Obr. 19) a výsledkov tabuliek (Tab. 4) a (Tab. 5) bol vylepšený algoritmus podstatne úspešnejší v prvých dvoch funkciách. Pri ostatných funkciách dosiahli oba algoritmy podobné výsledky.

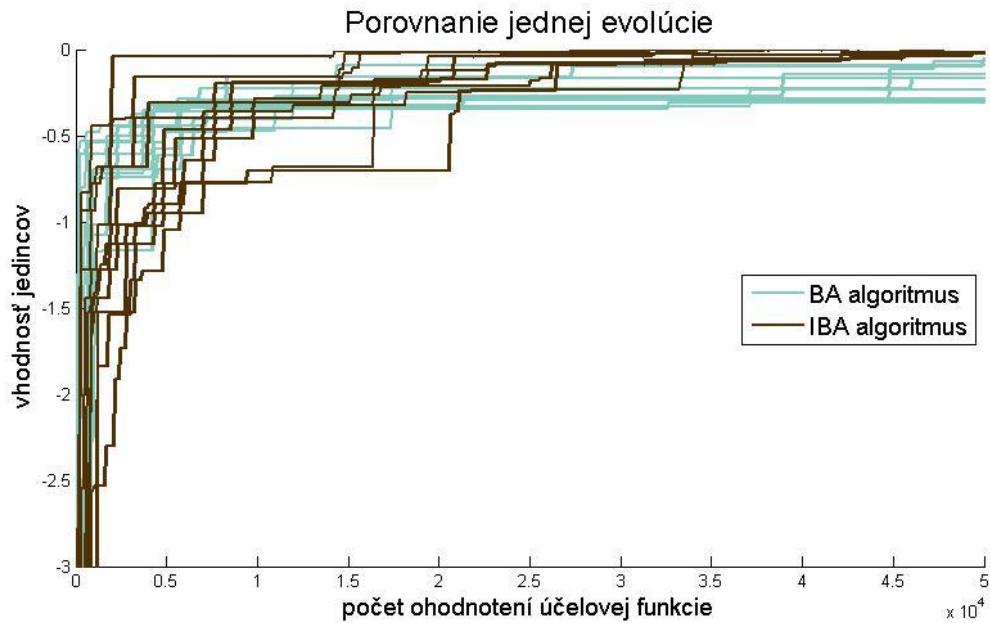


Obr. 18. Kvalita nejlepších jedinců algoritmov BA a IBA při 2D



Obr. 19. Kvalita nejlepších jedinců algoritmov BA a IBA při 10D

Na obrázku (Obr. 20) je zobrazený graf jedné evoluce pro oba algoritmy při funkci Katsuura. Na grafě vidieť, že Improved Bees Algorithm konverguje rýchlejšie než klasický Bees Algorithm. Je to spôsobené práve zmenšováním okolia ngh pri lokálnom prehľadávání.



Obr. 20. Porovnanie jednej evolúcie BA a IBA algoritmov

8.2 ABC Algorithm

Testy algoritmu boli urobené s nasledujúcimi vstupnými parametrami:

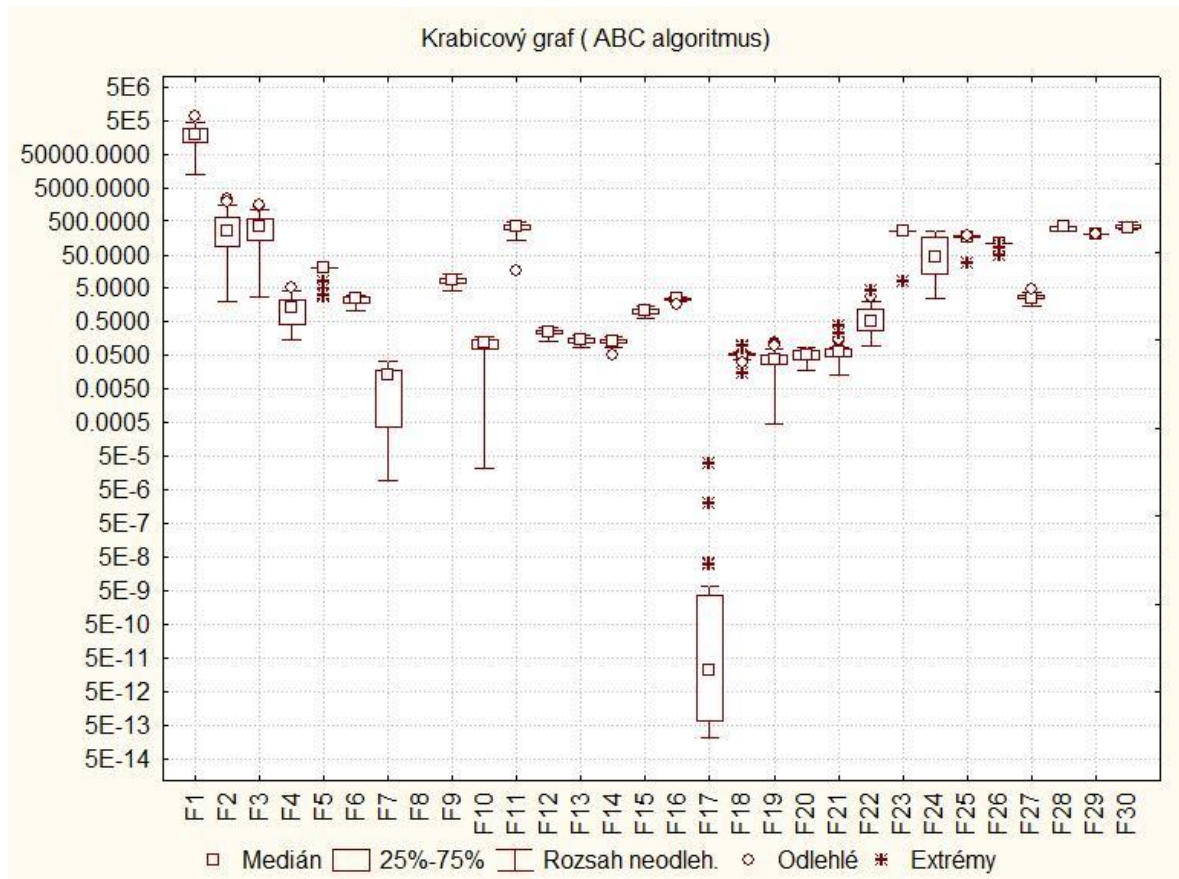
Veľkosť kolónie:	50
Počet zdrojov jedla:	colonySize / 2
Počet zvedov v jednej generácii:	1
Stagnačný limit:	100
Hranice prehľadávaného priestoru:	[-100; 100]
Počet generácií:	1000

Veľkosť kolónie a počet generácií boli nastavené tak, aby bolo možné porovnať výsledky s ostatnými algoritmi. Pri týchto hodnotách má algoritmus približne rovnaký počet ohodnotení účelovej funkcie ako ostatné algoritmy. Zvyšné parametre boli určené podľa zdrojov použitej literatúry. [25]

Tab. 6. Výsledky ABC algoritmu

Funkcia	Najlepšie riešenie	Priemer najlepších	Priemer mediánov	Najlepšie riešenie	Priemer najlepších	Priemer mediánov
	2D			10D		
1	0,899796265	34,91696509	31203,66338	15296,37356	141370,4071	13102334,55
2	0,943978675	26,58669473	31262,98746	2,991636438	338,3155987	71302864
3	0,500561159	19,84911378	31756,16152	49,82894547	425,1278997	12566,70014
4	1,12E-05	0,002951108	0,088112454	0,148207753	1,468930987	11,79406022
5	0	0	2,915330805	3,12414863	18,0036681	20,48682525
6	0	8,59E-07	0,527166099	0,82334993	2,080054101	8,127194931
7	3,13E-13	0,000311968	0,046923949	2,01E-05	0,016375667	0,349031491
8	0	0	0,237780838	0	4,14E-16	2,289280992
9	0	5,13E-06	1,854402555	4,277915952	8,342927527	43,19052499
10	0	0	2,981088703	1,48E-05	0,122406607	88,29996653
11	1,14E-13	0,047248884	197,8941853	25,4323702	285,0352061	1357,476633
12	0,013380731	0,098968034	3,859235711	0,137113584	0,259914043	1,530935055
13	0,016553173	0,036561924	0,242931243	0,094153119	0,151103298	1,122296042
14	0,001038961	0,005344269	0,18358397	0,089745054	0,126727262	3,129612145
15	0	4,76E-06	0,123174618	0,435199788	1,09282277	8,94202444
16	0	1,06E-05	0,075532196	2,043555319	2,460438361	4,035181409
17				7,20E-17	2,57E-06	34,1964017
18				0,023240663	0,056428721	0,877009111
19				0,000243786	0,044606633	2,170702204
20				0,018318006	0,058875303	1,08995328
21				0,013181805	0,093564852	12323,63908
22				0,154855022	0,825629261	44,94399824
23	7,86E-99	7,86E-99	6,87607706	256,7179222	256,7179222	256,7179222
24	5,84E-15	0,334205444	128,7025013	1,284858236	94,63759692	290,8494857
25	1,55E-99	0,050401283	143,5241452	171,944326	176,2912644	199,3480724
26	4,01E-14	0,027620353	76,95677008	37,02203194	93,47011154	114,0213455
27	0,235521403	0,887585745	357,0415607	1,764615955	2,760001375	444,13578
28	7,86E-99	0,000611979	255,5072858	242,1067122	318,4982537	433,1918799
29				203,887321	205,9082257	736,5466817
30				258,0824472	340,77619	1202,61419

V tabuľke (Tab. 6) sa nachádzajú výsledky testovania ABC algoritmu. Z výsledkov je zrejmé, že algoritmus podal pri 2 dimenziách veľmi dobrý výkon. Pri 10 dimenziách mal problém najmä s Eliptickou funkciou, čo bolo spôsobené predovšetkým rotáciou funkcie. Pri ostatných funkciách sa dokázal vo väčšine prípadov uspokojivo priblížiť ku globálnemu extrému.



Obr. 21. Krabicový graf algoritmu ABC pre 10 dimenzií

Na obrázku (Obr. 21) je zobrazený krabicový graf výsledkov najlepších riešení pre 10 dimenzií.

8.3 PSO Algorithm

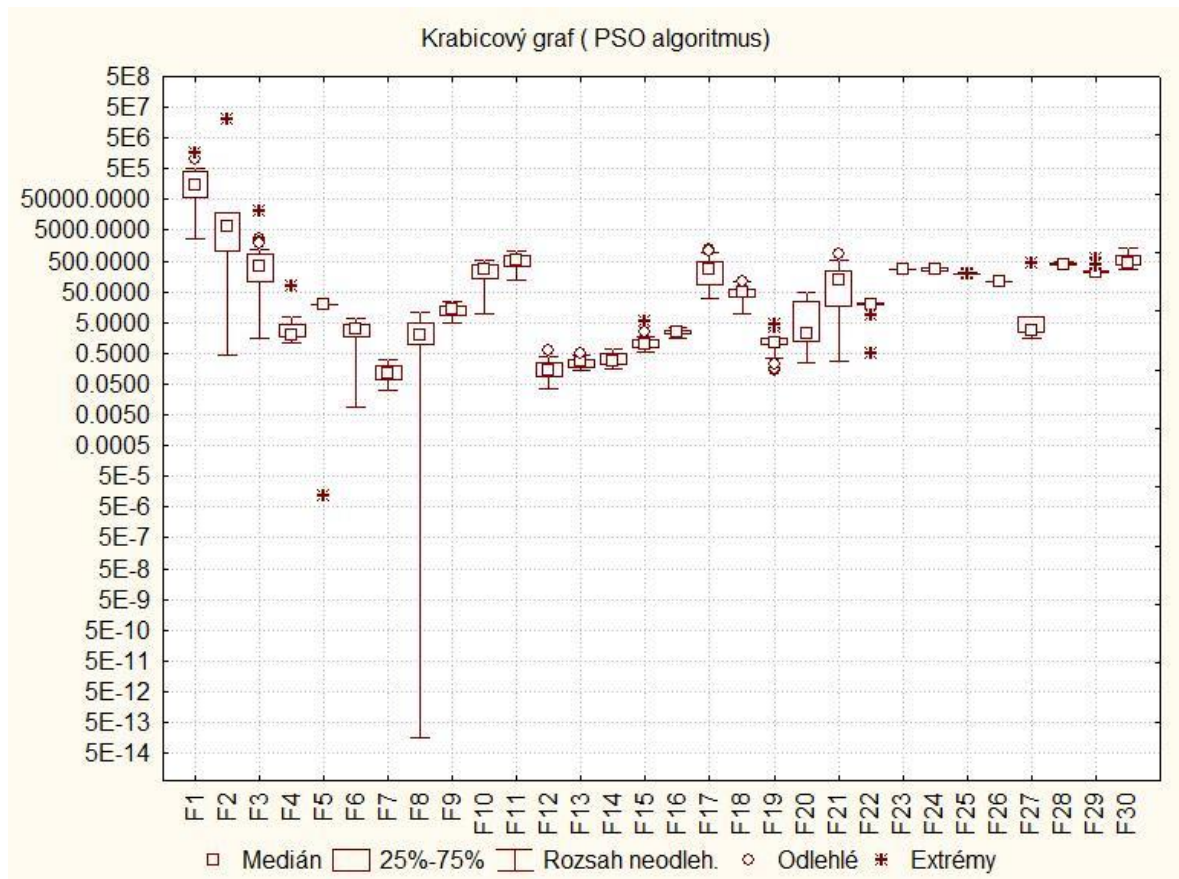
Počet častíc:	50
Váha zotrvačnosti:	0,729
Kognitívna váha:	1,49445
Sociálna váha:	1,49445
Pravdepodobnosť úmrtia:	0,01
Hranice prehľadávaného priestoru:	[-100; 100]
Počet generácií:	1000

Hodnoty algoritmu boli upravené podľa odporučených nastavení zo zdrojov literatúry. [26] Počet jedincov a generácií bol zvolený tak, aby mal algoritmus približne rovnaký počet ohodnotení účelovej funkcie ako ostatné testované algoritmy.

Tab. 7. Výsledky PSO algoritmu

Funkcia	Najlepšie riešenie	Priemer najlepších	Priemer mediánov	Najlepšie riešenie	Priemer najlepších	Priemer mediánov
	2D			10D		
1	0	4,92E-09	69103989,61	172,3127436	54504,70885	16504956,71
2	0	6,16E-09	14484416,32	0,496217494	9380,287819	434722974,5
3	0	1,28E-08	6956286,674	0,01324838	4,229575782	2210345,249
4	0	1,47E-20	1,343682251	0,007068994	0,147780324	109,4461322
5	0	2,00E-14	5,030930664	19,99976993	20,02243208	21,03698419
6	0	0	0,256872185	0,000568841	2,259504475	3,660437493
7	0	0	0,265227734	0,024573294	0,103531181	2,816300058
8	0	0	1,831143904	0	1,160784727	15,84334051
9	0	0	53,28227248	3,53986957	127,8303706	548,2860629
10	0	0	1,272764579	2,101152549	7,334945709	34,35378356
11	0	0,010405775	15,13709796	7,017284699	325,8956974	863,5086951
12	0	7,12E-12	5,898675403	0,014128477	0,081271218	4,618400909
13	8,55E-06	0,005553526	0,400083746	0,073816948	0,18795724	0,85455458
14	4,40E-06	0,001117282	2,903089495	0,041500759	0,116911361	2,774067746
15	0	0	3676,555371	0,441993686	0,922758716	61,19005733
16	0	0	0,259264481	1,140734834	2,16015674	3,300339951
17				1,73E-11	163,7607601	102612983,7
18				0,025645595	39,04366703	385594034,3
19				2,79E-06	1,073450551	9,167006551
20				0,001396416	0,200206746	74833167,53
21				0,001236853	8,327895269	2032317,226
22				0,237199454	17,6983478	82,87006202
23	7,86E-99	7,66E-14	18,47482018	256,7179222	256,7207337	264,8285274
24	7,63E-101	3,28E-15	32,97685441	135,2131646	230,6904646	423,0993415
25	1,55E-99	2,93E-15	12,00358711	55,27603789	187,0889574	191,4684603
26	5,40E-102	2,50E-15	19,40433276	106,0495306	106,0661229	106,3289678
27	1,32E-05	0,000689947	59,3794941	0,61737959	1,612956212	20,17723308
28	7,86E-99	7,86E-99	187,724181	356,0181835	374,4648029	430,6778889
29				202,9458404	203,7370001	7033,658911
30				198,4571047	337,6630851	4365966,583

Výsledky testovania preukázali schopnosť algoritmu spoľahlivo nájsť extrém pri nízkom počte argumentov účelovej funkcie. Pri 10 dimenziách sa výsledky najlepších riešení tiež približovali k hľadanému extrému, najväčšia odchýlka je pri prvej Eliptickej funkcii s rotáciou. Napriek dobrej schopnosti algoritmu nájsť extrém počas 30 evolúcií, v mnohých prípadoch je priemerná hodnota najlepších riešení veľmi vysoká, čo by mohlo svedčiť o jeho nestabilite.



Obr. 22. Krabicový graf algoritmu PSO pre 10 dimenzií

Na obrázku (Obr. 22) je zobrazený krabicový graf najlepších riešení všetkých funkcií pre 10 dimenzií.

8.4 Firefly Algorithm

Algoritmus bol testovaný pri nasledujúcich vstupných nastaveniach:

Veľkosť populácie:	50
Výpočet atraktivity:	odhadom
Distribúcia náhodných čísel:	Lévyho lety
Hranice prehľadávaného priestoru:	[-100; 100]
Počet generácií:	962

Výsledky testov potrebné k štatistickému porovnaniu algoritmu pre účely tejto práce poskytla Bc. Anežka Kazíková, ktorá spracovala algoritmus FFA v rámci svojej diplomovej práce pod názvom *Moderní hejnové algoritmy*. [27]

Vstupné parametre boli nastavené na dané hodnoty, aby algoritmus dosahoval približne rovnaký počet ohodnotení účelovej funkcie ako ostatné testované algoritmy.

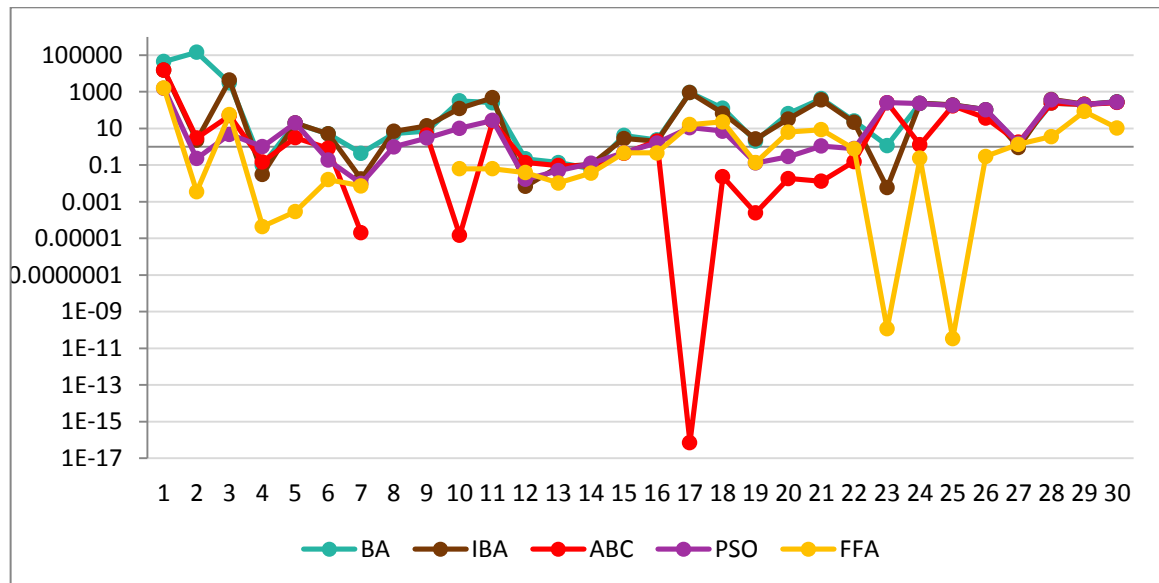
Tab. 8. Výsledky FFA algoritmu

Funkcia	Najlepšie riešenie	Priemer najlepších	Priemer mediánov	Najlepšie riešenie	Priemer najlepších	Priemer mediánov
	2D			10D		
1	2,88E-05	7,13818	7,13818	1601,19	53641,1	53641,1
2	1,09E-06	28,3147	28,3147	0,00358664	132,636	132,636
3	2,04E-08	40,3832	40,3832	58,0731	217,044	217,044
4	1,54E-30	0,000277175	0,000277175	4,42E-05	1,23577	1,23577
5	3,37E-14	1,17E-13	2,21E-11	0,000290044	7,48331	7,48331
6	-1,41E-05	-1,41E-05	-1,41E-05	0,0159672	0,165723	0,165723
7	0	0,0023011	0,0023011	0,00739604	0,0750819	0,0750819
8	0	0	0	0	2,474	6,83187
9	0	0	0	0	3,94621	9,30462
10	-9,99E-16	0,093652	52,1667	0,0624544	27,3849	52,5243
11	-7,77E-16	0,135275	16,3753	0,0624544	21,2352	147,76
12	3,09E-13	1,43E-12	6,38154	0,0389095	0,331308	2,07699
13	6,84E-05	0,0028265	0,00353888	0,010486	0,0271499	0,0282099
14	0,000297031	0,0226211	0,357571	0,0368677	0,227541	0,937046
15	0	0,0072739	0,873816	0,46128	1,18734	3,9433
16	0	0,0155455	0,31872	0,460891	1,18741	3,50687
17				16,3782	276,24	313,013
18				22,9093	77,2864	77,2864
19				0,136644	0,522193	1,13463
20				6,3502	19,3883	19,6595
21				8,53228	102,157	134,553
22				0,757878	14,5964	48,6941
23	3,43E-12	1,24E-11	2,25E-09	1,16E-10	0,0373594	0,0373594
24	1,68E-13	2,36E-12	4,64E-10	0,2357	71,1373	118,377
25	6,73E-13	2,21E-12	5,06E-10	3,43E-11	49,9741	87,726
26	1,26E-12	1,17E-11	2,12E-09	0,294107	116,021	119,635
27	1,01E-06	7,21E-05	8,25E-05	1,37988	352,817	436,183
28	3,68E-12	1,21E-11	2,09E-09	3,57634	1277,65	1302,88
29				84,0035	300,333	354,37
30				10,4174	219,317	220,635

V tabuľke (Tab. 8) sú uvedené výsledky testovania algoritmu FFA pri 2 a 10 dimenziách. Algoritmus preukázal pri 2-dimenziálnom probléme výbornú schopnosť nájsť extrém funkcie. Pri 10 dimenziách sa vo väčšine prípadov dokázal uspokojivo priblížiť k extrému, najväčšiu vzdialenosť zaznamenal pri Eliptickej funkcii s rotáciou.

8.5 Porovnanie všetkých algoritmov

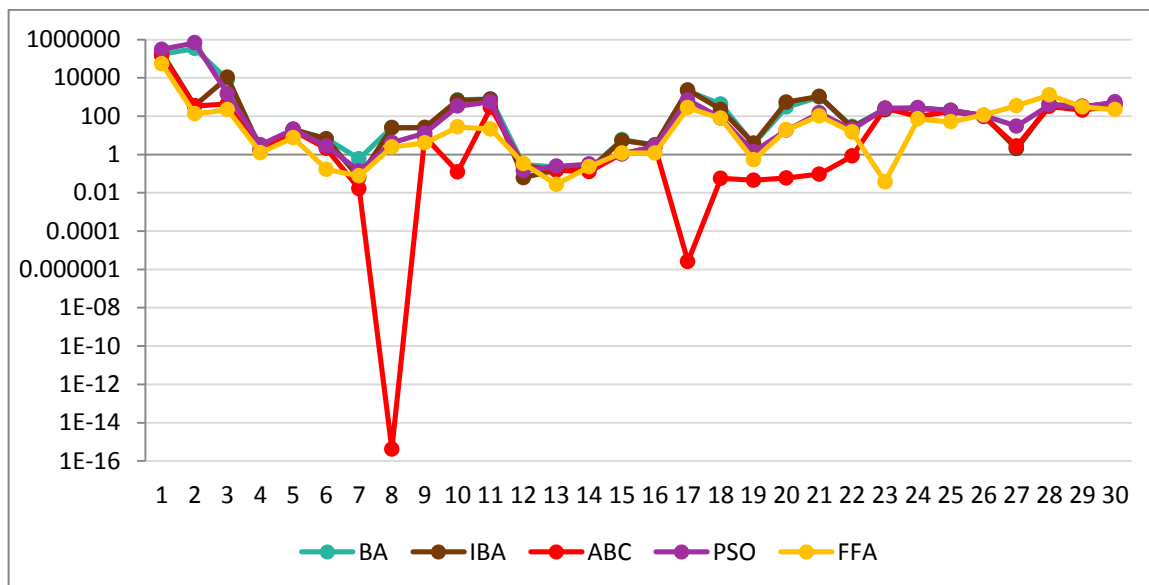
Algoritmy uvedené v predchádzajúcich kapitolách budú porovnané pri 10 dimenziách. Vzhľadom na to, že algoritmy BA a IBA hľadajú maximum účelovej funkcie, výsledné hodnoty sú všetky záporné. Pre účely vzájomného porovnania bude teda použitá ich absolútna hodnota vzdialenosti od extrému. Hodnoty extrémov pre všetky testované funkcie sú rovnaké, teda 0.



Obr. 23. Najlepšie riešenia zo všetkých evolúcií pre všetky algoritmy

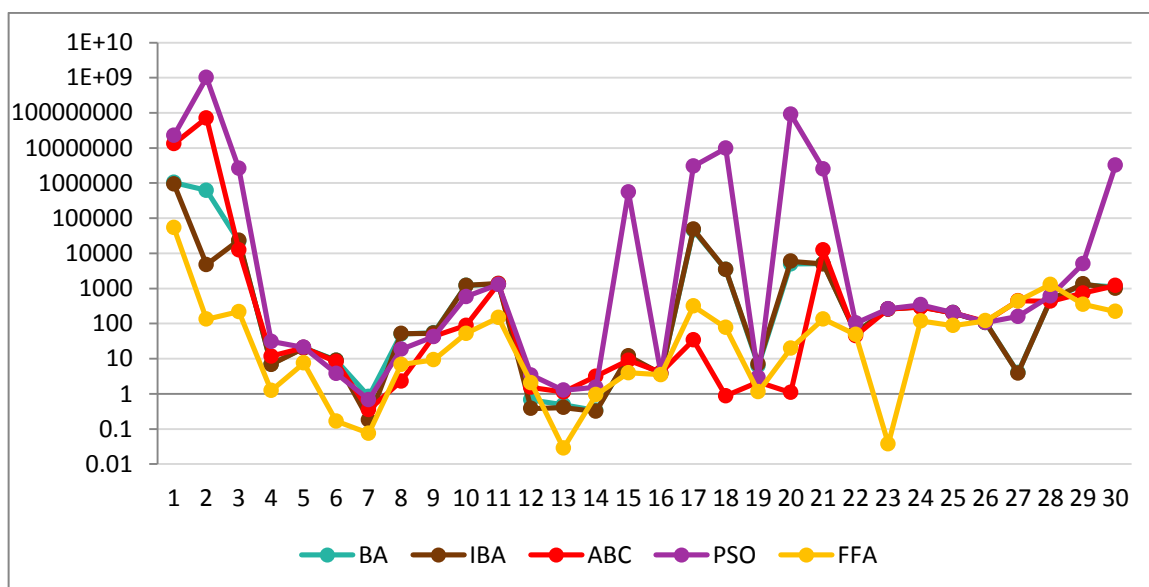
Na obrázku (Obr. 23) je zobrazený graf pre porovnanie najlepších riešení jednotlivých algoritmov zo všetkých evolúcií. Prerušené čiary zobrazuje graf v prípade, že daný algoritmus dáva výsledok 0, čo predstavuje najlepšie možné dosiahnuté riešenie. V ostatných prípadoch platí, že čím je nižšie položená krivka, tým lepšie riešenie algoritmus našiel.

Z grafu je vidieť rôznorodosť výsledkov pre jednotlivé algoritmy, čo podporuje „No Free Lunch Teorém“. Najlepšie výsledky dosahujú ABC a FFA algoritmy, pričom vo väčšine prípadov sa jedná o rôzne funkcie.



Obr. 24. Geometrický priemer najlepších riešení pre všetky algoritmy

Na obrázku (Obr. 24) je zobrazený graf výsledkov geometrického priemeru najlepších riešení zo všetkých evolúcií pre všetky algoritmy. V tomto prípade sa algoritmus ABC stále odlišuje výrazne lepšimi hodnotami riešení viacerých funkcií oproti ostatným algoritmom, len v prípade Kompozitnej funkcie 1 podal jednoznačne najlepší výsledok algoritmus FFA. Všetky algoritmy majú problém s hľadáním extrémů pri prvej – Eliptickej funkcii.



Obr. 25. Geometrický priemer mediánov posledných generácií pre všetky algoritmy

Na obrázku (Obr. 25) je zobrazený graf pre porovnanie výsledkov geometrického priemeru mediánov posledných generácií všetkých evolúcií pre všetky algoritmy. Vo väčšine prípa-

dov dosiahol algoritmus FFA najlepšie výsledky a naopak najhoršie výsledky vykazuje algoritmus PSO.

Výhodou algoritmu BA a jeho vylepšenej verzie IBA je schopnosť súčasného lokálneho aj globálneho prehľadávania. Algoritmus IBA obsahuje znižovanie prehľadávanej lokálnej oblasti, ako aj jej opustenie pri dlhodobom neúspechu, čo podporuje rýchlejšiu konvergenciu a môže byť prevenciou voči uviaznutiu algoritmu v lokálnom extréme. Nevýhodou oboch algoritmov je množstvo vstupných parametrov, ktoré je potrebné ladiť.

ABC algoritmus má silné lokálne aj globálne prehľadávanie a podal pri testovaní veľmi dobré výsledky. Jeho nevýhodou môže byť pravdepodobnostný výber jedincov na lokálne prehľadávanie, čo môže spôsobiť, že budú opustené aj lepšie riešenia.

Algoritmus PSO podporuje globálne prehľadávanie, je rýchly a pri viacerých evolúciách dokázal vyhľadať dobré riešenia. Veľký rozdiel medzi najlepším riešením, priemerom najlepších výsledkov a priemerom mediánov však naznačujú nejednoznačnosť algoritmu a jeho pomalú konvergenciu.

Nakoľko boli výsledky algoritmu FFA dodané z externého zdroja, nemožno posúdiť vplyv nastavenia počiatočných parametrov. Samotný algoritmus v porovnaní s ostatnými preukázal výbornú schopnosť nájsť globálny extrém.

ZÁVER

Úlohou tejto diplomovej práce bolo vysvetliť problematiku moderných rojových algoritmov, naprogramovať vybraný optimalizačný algoritmus, otestovať a porovnať ho s ďalšími rojovými algoritmi. K tomuto účelu boli vybrané dva algoritmy – včelí algoritmus (BA) aj s jeho modifikovanou verziou (IBA) a umelá včelia kolónia (ABC). Tieto algoritmy boli naprogramované a otestované v prostredí C#. Výkon algoritmov bol porovnaný s ďalšími dvomi algoritmi – optimalizácia rojom častíc (PSO) a optimalizácia svetluškami (FFA).

Teoretická časť práce vysvetľuje základné princípy evolučných algoritmov a optimalizačných techník. Súčasťou teoretickej časti je aj popis niektorých vybraných rojových algoritmov, optimalizácii včelím rojom je venovaná celá kapitola.

V praktickej časti sú najskôr predstavené samotné naprogramované algoritmy aj s popisom aplikácií na ich ovládanie. V ďalšej časti sú predstavené výsledky jednotlivých algoritmov, po ktorých nasleduje vzájomné porovnanie výkonov všetkých spomenutých algoritmov.

Všetky algoritmy boli testované na zložitých optimalizačných funkciách podľa IEEE CEC 2014. Výsledky testovania potvrdili pravdivosť „No Free Lunch Teorému“, teda že pre rôzne funkcie sú vhodné rôzne optimalizačné algoritmy. Žiadny z testovaných algoritmov sa nepreukázal ako univerzálny nástroj na riešenie všetkých funkcií.

ZOZNAM POUŽITEJ LITERATURY

- [1] ZELINKA, Ivan, Zuzana OPLATKOVÁ, Miloš ŠEDA, Pavel OŠMERA a František VČELAŘ. *Evoluční výpočetní techniky: principy a aplikace*. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3
- [2] MACH, Marián. *Evolučné algoritmy: prvky a principy*. Vyd. 1. Košice: Elfa, 2009, viii, 237 s. ISBN 978-80-8086-123-0
- [3] OBITKO, Marek. Introduction to Genetic Algorithms. *Genetic Algorithms* [online]. 2008 [cit. 2015-03-10]. Dostupné z: <http://www.obitko.com/tutorials/genetic-algorithms/index.php>
- [4] XIAOHUI HU, PH.D. PSO Tutorial. In: *Particle Swarm Optimization* [online]. 2006 [cit. 2015-03-12]. Dostupné z: <http://www.swarmintelligence.org/tutorials.php>
- [5] Ant Colony Optimization. DORIGO, Marco. *Scholarpedia* [online]. 2007 [cit. 2015-03-12]. Dostupné z: http://www.scholarpedia.org/article/Ant_colony_optimization
- [6] DORIGO, Marco, Mauro BIRATTARI a Thomas STUTZLE. *Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique* [online]. 2006 [cit. 2015-03-12]. Dostupné z: <http://iridia.ulb.ac.be/~mbiro/paperi/IridiaTr2006-023r001.pdf>
- [7] YANG, Xin-She. *Nature-inspired metaheuristic algorithms*. 2nd ed. Frome, Luniver Press, 2010, vi, 148 s. ISBN 978-1-905986-28-6
- [8] KARABOGA, Dervis a Bahriye AKAY. A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review*. 2009, vol. 31, 1-4, s. 61-85. DOI: 10.1007/s10462-009-9127-4. Dostupné z: <http://link.springer.com/10.1007/s10462-009-9127-4>
- [9] JAKUŠ, Miroslav. O včelách. *Včely* [online]. 2005 [cit. 2015-03-18]. Dostupné z: http://vcely.euweb.cz/Central_O_vcelach.htm
- [10] MATTILA, H. R., K. M BURKE a T. D SEELEY. Genetic diversity within honeybee colonies increases signal production by waggle-dancing foragers. *Proceedings of the Royal Society B: Biological Sciences*. 2008-04-07, vol. 275, issue 1636, s. 809-816. DOI: 10.1098/rspb.2007.1620. Dostupné z: <http://rspb.royalsocietypublishing.org/cgi/doi/10.1098/rspb.2007.1620>

- [11] TARPY, David R. The Honey Bee Dance Language. *NC State University* [online]. 2004 [cit. 2015-03-19]. Dostupné z: <http://www.cals.ncsu.edu/entomology/apiculture/pdfs/1.11%20copy.pdf>
- [12] HOŠEK, Pavel. O včelích tanečcích, orientaci a robotech. *Přírodovědecký časopis Vesmír* [online]. Praha: Vesmír, 1995, roč. 74, č. 1, s. 25 [cit. 2015-03-18]. Dostupné z: <http://casopis.vesmir.cz/clanek/o-vcelich-taneccich-orientaci-a-robotech>
- [13] KARABOGA, Dervis. An idea based on honey bee swarm for numerical optimization. In: *Technical Report TR06* [online]. 2005 [cit. 2015-03-19]. Dostupné z: http://mf.erciyes.edu.tr/abc/pub/tr06_2005.pdf
- [14] KARABOGA, Dervis a Bahriye BASTURK. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems. *Foundations of Fuzzy Logic and Soft Computing* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, s. 789 [cit. 2015-03-19]. ISBN 978-3-540-72917-4.
- [15] PHAM, D., A. GHANBARZADEH, E. KOÇ, S. OTRI, S. RAHIM a M. ZAIDI. The Bees Algorithm – A Novel Tool for Complex Optimisation Problems. *Intelligent production machines and systems: 2nd I*PROMS Virtual Conference, 3-14 July 2006* [online]. Oxford: Elsevier, 2006, 454–459 [cit. 2015-03-20]. ISBN 0080451578.
- [16] BO XING, Wen-Jing Gao. *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms* [online]. Aufl. 2014. Cham: Springer International Publishing, 2014 [cit. 2015-03-18]. ISBN 978-3-31-9034-041. Dostupné z: <http://link.springer.com/book/10.1007/978-3-319-03404-1>
- [17] HADDAD, Omid Bozorg, Abbas AFSHAR a Miguel A. MARÍÑO. Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization. *Water Resources Management*. 2006, vol. 20, issue 5, s. 661-680. DOI: 10.1007/s11269-005-9001-3. Dostupné z: <http://link.springer.com/10.1007/s11269-005-9001-3>
- [18] ABBASS, H.A. MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach. *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. IEEE, 2001, vol.1, s. 207-

214. DOI: 10.1109/CEC.2001.934391. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=934391>
- [19] MUÑOZ, Mario A., Jesús A. LÓPEZ a Eduardo CAICEDO. An artificial beehive algorithm for continuous optimization. *International Journal of Intelligent Systems*. 2009, vol. 24, issue 11, s. 1080-1093. DOI: 10.1002/int.20376. Dostupné z: <http://doi.wiley.com/10.1002/int.20376>
- [20] YANG, Xin-She. A New Metaheuristic Bat-Inspired Algorithm. In: (EDS), Juan González ... [et al.]. *Nature inspired cooperative strategies for optimization (NCSO 2010)* [online]. New York: Springer, 2010, s. 65 [cit. 2015-03-21]. ISBN 9783642125386ISSN 1860-949X. DOI: 10.1007/978-3-642-12538-6_6. Dostupné z: http://link.springer.com/10.1007/978-3-642-12538-6_6
- [21] TEODOROVIĆ, Dušan. Bee Colony Optimization (BCO). CHEE, Peng Lim, L JAIN a Satchidananda DEHURI. *Innovations in swarm intelligence*. New York: Springer, c2009, s. 39. ISBN 9783642042249.
- [22] KARABOGA, Dervis a Bahriye AKAY. A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation* [online]. 2009, vol. 214, issue 1, s. 108-132 [cit. 2015-03-19]. DOI: 10.1016/j.amc.2009.03.090. Dostupné z: <http://linkinghub.elsevier.com/retrieve/pii/S0096300309002860>
- [23] LIANG, J.J., B.Y. QU a P.N. SUGANTHAN. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. In: *Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Technical Report, Nanyang Technological University, Singapore* [online]. 2013 [cit. 2015-04-27]. Dostupné z: <http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC-2014/Definitions%20of%20CEC2014%20benchmark%20suite%20Part%20A.pdf>
- [24] PHAM, D T a M CASTELLANI. The Bees Algorithm: modelling foraging behaviour to solve continuous optimization problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* [online]. 2009-12-1, vol. 223, issue 12, s. 2919-2938 [cit. 2015-05-01]. DOI: 10.1243/09544062JMES1494. Dostupné z: <http://pic.sagepub.com/lookup/doi/10.1243/09544062JMES1494>

- [25] YUCE, Baris, Michael PACKIANATHER, Ernesto MASTROCINQUE, Duc PHAM a Alfredo LAMBIASE. 2013. Honey Bees Inspired Optimization Method: The Bees Algorithm. *Insects* [online]. 4(4): 646-662 [cit. 2015-05-08]. DOI: 10.3390/insects4040646. ISSN 2075-4450. Dostupné z: <http://www.mdpi.com/2075-4450/4/4/646/>
- [26] EBERHART, R.C. a Y. SHI. 2000. Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)* [online]. IEEE, (Vol. 1): 84-88 [cit. 2015-05-08]. DOI: 10.1109/CEC.2000.870279. ISBN 0-7803-6375-2. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=870279>
- [27] KAZÍKOVÁ, Anežka. *Moderní hejnové algoritmy*. 2015. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce Roman Šenkeřík.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

ABC	Artificial Bee Colony Algorithm.
ABHA	Artificial Beehive Algorithm.
ACO	Ant Colony Optimization.
BA	Bees Algorithm.
BCO	Bee Colony Optimization.
BSO	Bee Swarm Optimization.
CEC	Congress on Evolutionary Computation.
CS	Cuckoo Search.
FFA	Firefly Algorithm.
HBMO	Honey Bee Mating Optimization.
IBA	Improved Bees Algorithm.
IEEE	Institute of Electrical and Electronics Engineers.
MBO	Marriage in Honey Bees Optimization.
PSO	Particle Swarm Optimization.
SOMA	Self-Organizing Migration Algorithm.

ZOZNAM OBRÁZKOV

Obr. 1. Všeobecný evolučný cyklus	12
Obr. 2. Plocha vhodnosti – Schafferova F7 funkcia	13
Obr. 3. Jednobodová mutácia	14
Obr. 4. Dvojbodové kríženie.....	15
Obr. 5. Pohyb jedincov podľa algoritmu PSO	22
Obr. 6. Mravce pri hľadaní potravy	23
Obr. 7. Natriasavý tanec	34
Obr. 8. Diagram algoritmu BA	38
Obr. 9. Generačný cyklus BA algoritmu	38
Obr. 10. Diagram ABC algoritmu	42
Obr. 11. Užívateľské rozhranie programu k algoritmu BA	50
Obr. 12. Ukážka výsledných výpočtov programu	51
Obr. 13. Diagram tried BA algoritmu.....	52
Obr. 14. Užívateľské rozhranie k algoritmu ABC.....	56
Obr. 15. Diagram tried algoritmu ABC	57
Obr. 16. Evolučný vývoj populácie BA pre Weierstrassovu funkciu.....	64
Obr. 17. Evolučný vývoj populácie IBA pre Weierstrassovu funkciu	65
Obr. 18. Kvalita najlepších jedincov algoritmov BA a IBA pri 2D	66
Obr. 19. Kvalita najlepších jedincov algoritmov BA a IBA pri 10D	66
Obr. 20. Porovnanie jednej evolúcie BA a IBA algoritmov	67
Obr. 21. Krabicový graf algoritmu ABC pre 10 dimenzií	69
Obr. 22. Krabicový graf algoritmu PSO pre 10 dimenzií.....	71
Obr. 23. Najlepšie riešenia zo všetkých evolúcií pre všetky algoritmy.....	73
Obr. 24. Geometrický priemer najlepších riešení pre všetky algoritmy.....	74
Obr. 25. Geometrický priemer mediánov posledných generácií pre všetky algoritmy	74

ZOZNAM TABULIEK

Tab. 1. Zobrazenie jedného jedinca v populácii	12
Tab. 2. Populácia s viacerými jedincami	13
Tab. 3. Prehľad IEEE CEC'14 testovacích funkcií	60
Tab. 4. Výsledky BA algoritmu.....	62
Tab. 5. Výsledky IBA algoritmu	63
Tab. 6. Výsledky ABC algoritmu	68
Tab. 7. Výsledky PSO algoritmu	70
Tab. 8. Výsledky FFA algoritmu	72

ZOZNAM PRÍLOH

P I: Testovacie funkcie

P II: CD s výsledkami testovania a so zdrojovými kódmi

PRÍLOHA P I: TESTOVACIE FUNKCIE

Algoritmy boli otestované na sade vybraných testovacích funkcií definovaných v IEEE CEC 2014. [23]

Pre všetky funkcie je definovaný minimalizačný problém nasledovne:

$$\text{Min } f(x), x = [x_1, x_2, \dots, x_D]^T \quad (31)$$

kde D je veľkosť dimenzie, teda počet argumentov funkcie.

$\mathbf{o}_{i1} = [o_{i1}, o_{i2}, \dots, o_{iD}]^T$ predstavuje vektor posunu, ktorý je načítaný zo súboru.

Posun funkcie je určený vzťahom:

$$F_i(x) = f_i(x - \mathbf{o}_i) + F_i^* \quad (32)$$

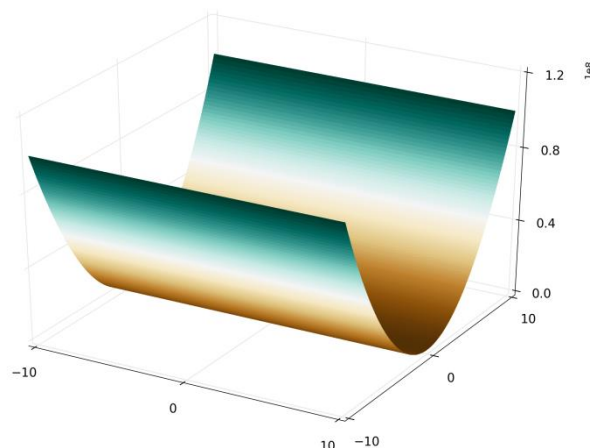
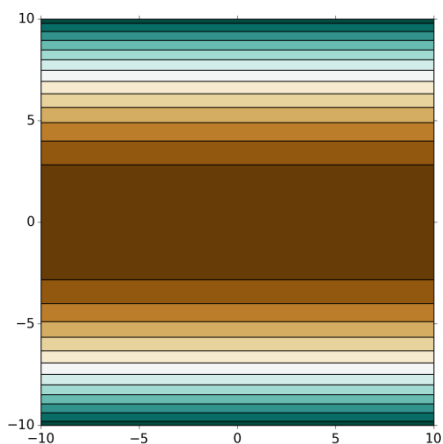
\mathbf{M}_i je rotačná matica načítaná zo súboru. Rotácia funkcie je určená vzťahom:

$$F_i(x) = f_i(\mathbf{M}(x - \mathbf{o}_i)) + F_i^* \quad (33)$$

Definície základných funkcií

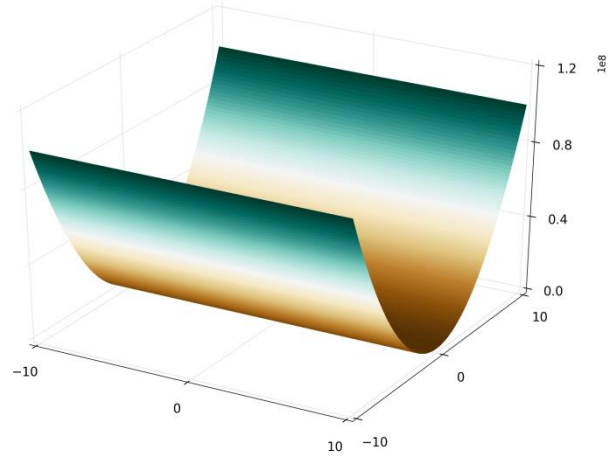
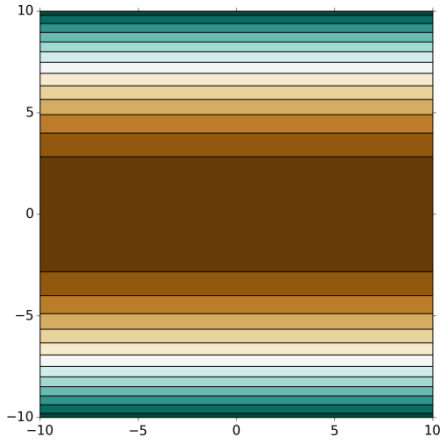
Eliptická funkcia

$$f_1(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} x_i^2 \quad (34)$$



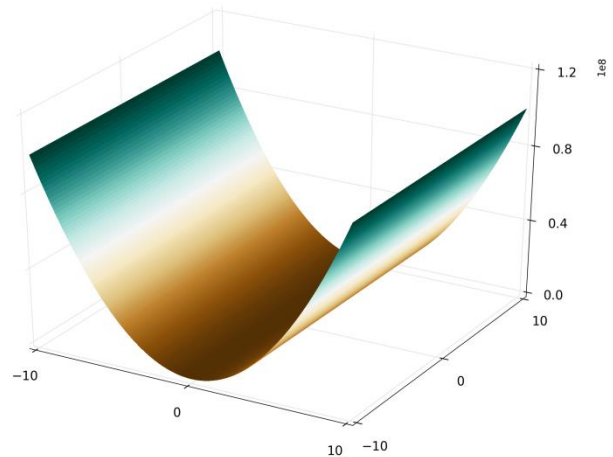
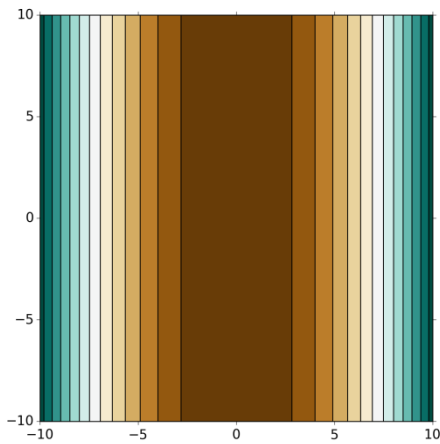
Bent Cigar funkcia

$$f_2(x) = x_1^2 + 10^6 \sum_{i=2}^D x_i^2 \quad (35)$$



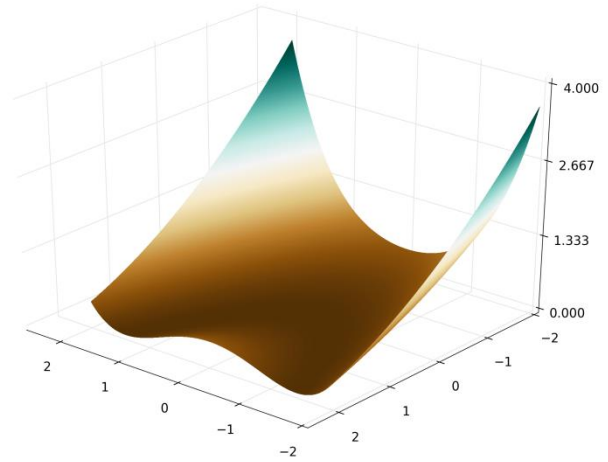
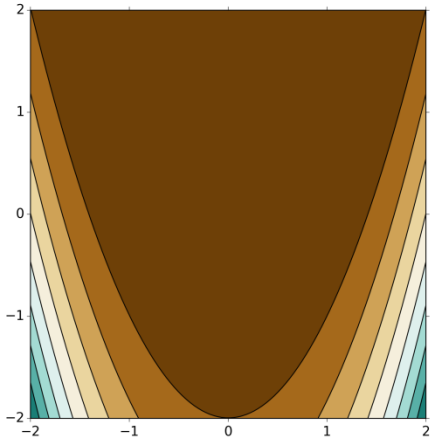
Discus funkcia

$$f_3(x) = 10^6 x_1^2 + \sum_{i=2}^D x_i^2 \quad (36)$$



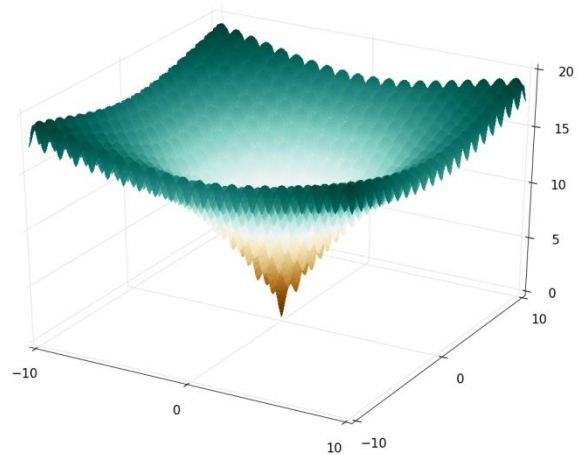
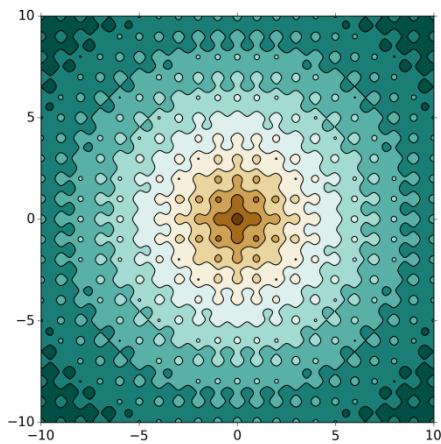
Rosenbrockova funkcia

$$f_4(x) = \sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \quad (37)$$



Ackleyho funkcia

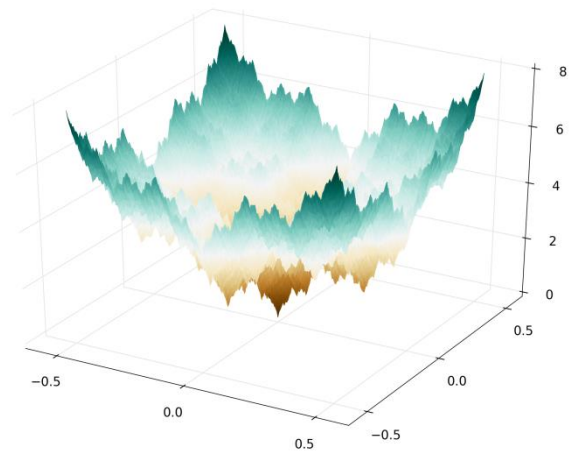
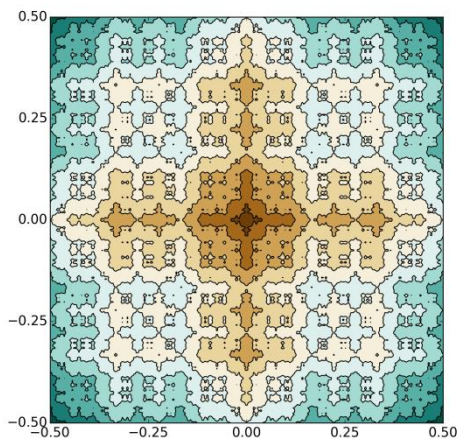
$$f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i)\right) + 20 + e \quad (38)$$



Weierstrass funkcia

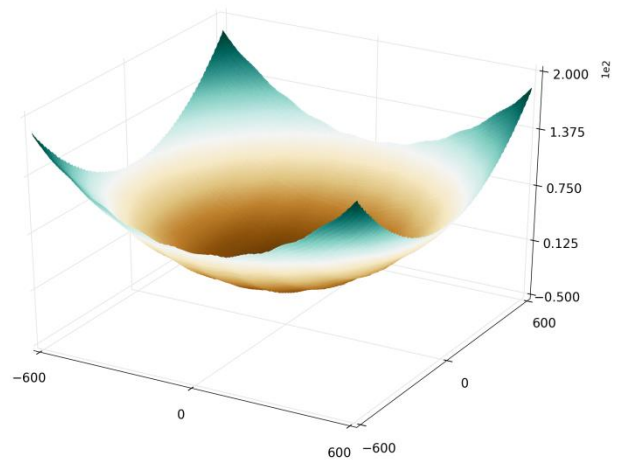
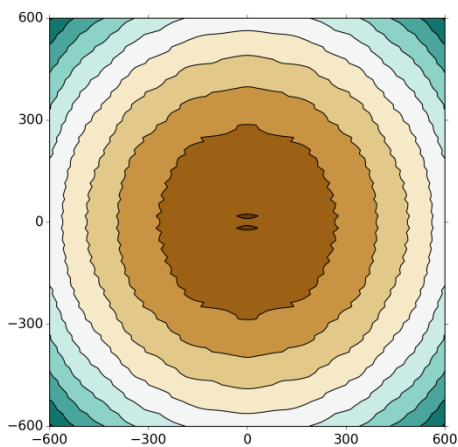
$$f_6(x) = \sum_{i=1}^D \left(\sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k(x_i + 0,5))] \right) - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k \cdot 0,5)] \quad (39)$$

$$a = 0,5, b = 3, k_{max} = 20$$



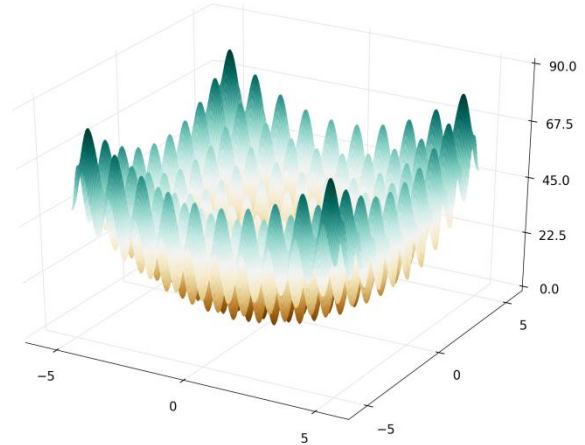
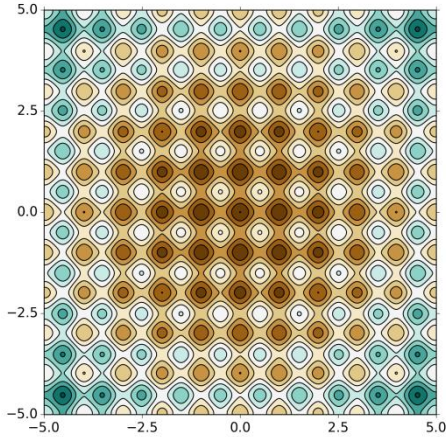
Griewankova funkcia

$$f_7(x) = \sum_{i=1}^D \frac{x_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (40)$$



Rastriginova funkcia

$$f_8(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (41)$$

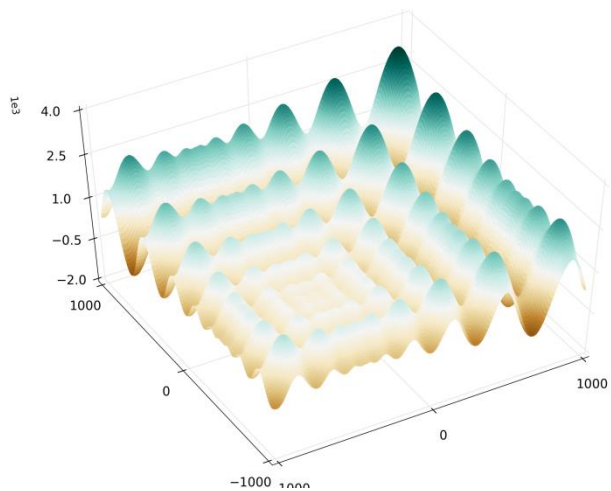
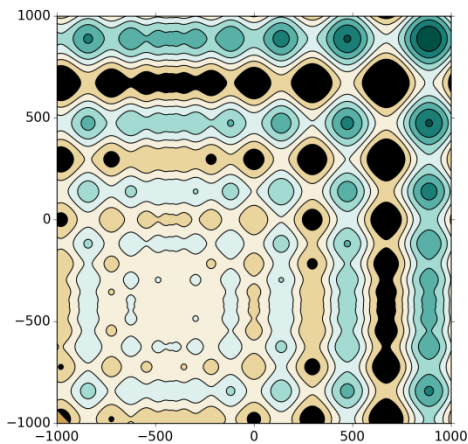


Modifikovaná Schwefelova funkcia

$$f_9(x) = 418,9829 \times D - \sum_{i=1}^D g(z_i) \quad (42)$$

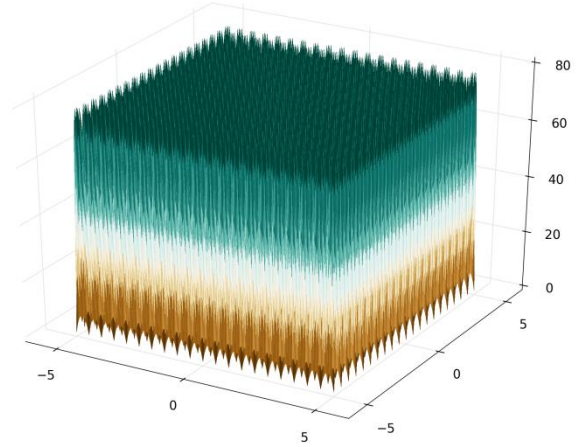
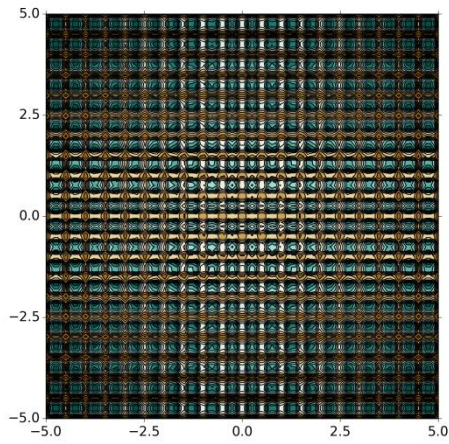
$$z_i = x_i + 4,209687462275036e + 002$$

$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}), & \text{if } |z_i| \leq 500 \\ (500 - \text{mod}(z_i, 500)) \sin(\sqrt{|500 - \text{mod}(z_i, 500)|}) - \frac{(z_i - 500)^2}{10000D}, & \text{if } z_i > 500 \\ (\text{mod}(|z_i|, 500) - 500) \sin(\sqrt{|\text{mod}(|z_i|, 500) - 500|}) - \frac{(z_i - 500)^2}{10000D}, & \text{if } z_i < -500 \end{cases}$$



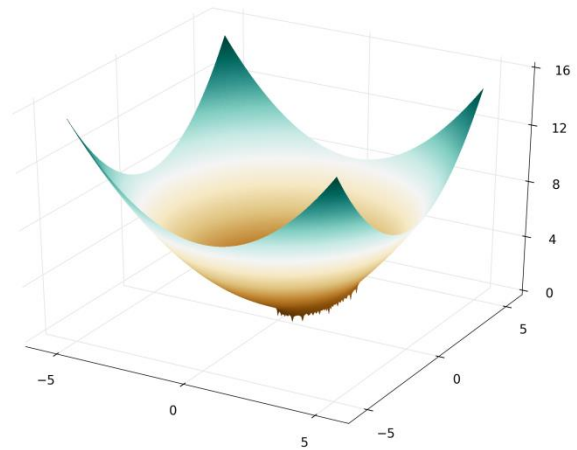
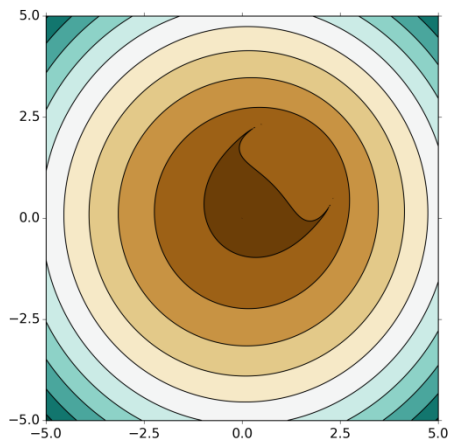
Katsuura funkcia

$$f_{10}(x) = \frac{10}{D^2} \prod_{i=1}^D \left(1 + i \sum_{j=1}^{32} \frac{|2^j x_i - \text{round}(2^j x_i)|^{\frac{10}{D^{1,2}}}}{2^j} \right) - \frac{10}{D^2} \quad (43)$$



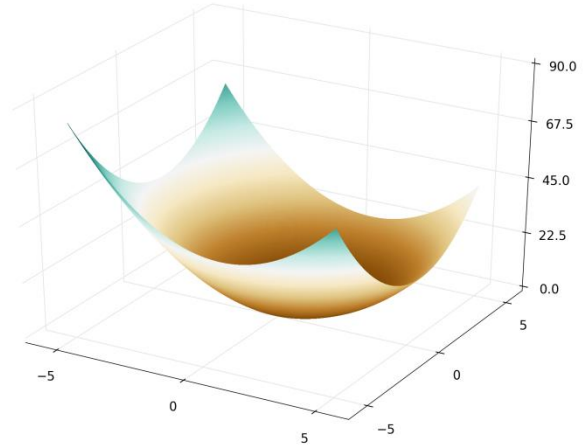
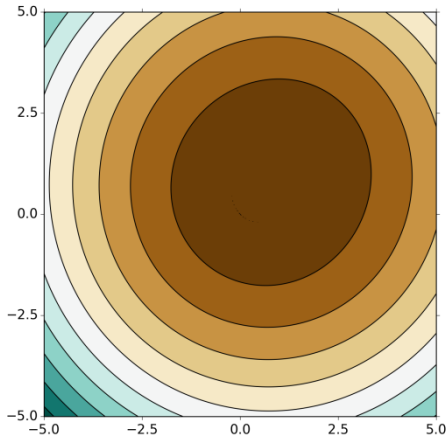
HappyCat funkcia

$$f_{11}(x) = \left| \sum_{i=1}^D x_i^2 - D \right|^{1/4} + (0,5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i) / D + 0,5 \quad (44)$$



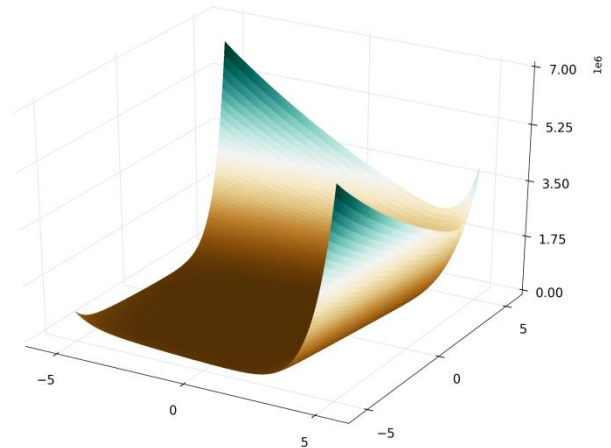
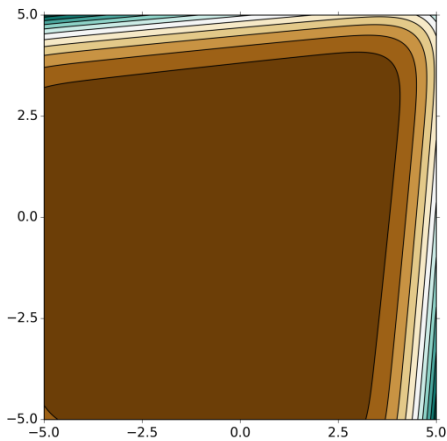
HGBat funkcia

$$f_{12}(x) = \left| \left(\sum_{i=1}^D x_i^2 \right)^2 - \left(\sum_{i=1}^D x_i \right)^2 \right|^{1/2} + (0,5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i) / D + 0,5 \quad (45)$$



Rozšířená Griewankova plus Rosenbrockova funkcia

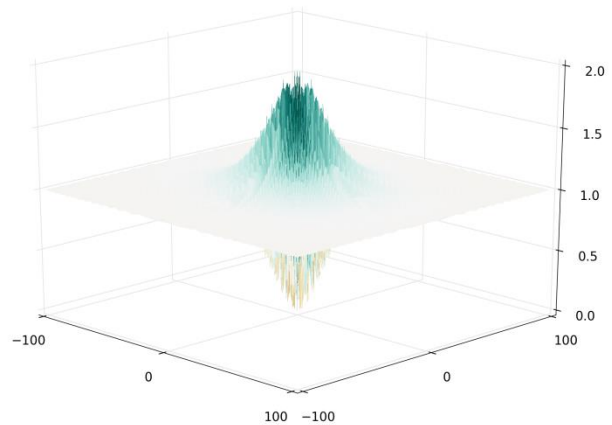
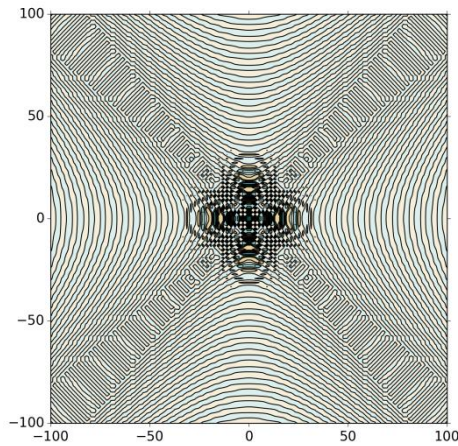
$$f_{13}(x) = f_7(f_4(x_1, x_2)) + f_7(f_4(x_2, x_3)) + \dots + f_7(f_4(x_{D-1}, x_D)) + f_7(f_4(x_D, x_1)) \quad (46)$$



Rozšírená Scafferova F6 funkcia

Scafferova F6 funkcia: $g(x, y) = 0.5 + \frac{(\sin^2(\sqrt{x^2+y^2})-0.5)}{(1+0.001(x^2+y^2))^2}$

$$f_{14}(x) = g(x_1, x_2) + g(x_2, x_3) + \dots + g(x_{D-1}, x_D) + g(x_D, x_1) \quad (47)$$



Hybridné funkcie

V sade testovacích hybridných funkcií sú premenné náhodne rozdelené do subkomponentov. Pre jednotlivé subkomponenty sú potom vypočítané rôzne základné testovacie funkcie podľa vzťahu:

$$F(x) = g_1(M_1 z_1) + g_2(M_2 z_2) + \dots + g_N(M_N z_N) + F^*(x) \quad (48)$$

kde $F(x)$ – je hybridná funkcia

$g_i(x)$ – je i -ta základná funkcia

N – je počet základných funkcií

p_i – je percentuálny podiel $g_i(x)$

Hybridná funkcia 1

$N = 3$

$p = [0,3; 0,3; 0,4]$

g_1 : Modifikovaná Schwefelova funkcia

g_2 : Rastriginova funkcia

g_3 : Eliptická funkcia

Hybridná funkcia 2

$N = 3$

$p = [0,3; 0,3; 0,4]$

g_1 : Bent Cigar funkcia

g_2 : HGBat funkcia

g_3 : Rastriginova funkcia

Hybridná funkcia 3

$N = 4$

$p = [0,2; 0,2; 0,3; 0,3]$

g_1 : Griewankova funkcia

g_2 : Weierstrassova funkcia

g_3 : Rosenbrockova funkcia

g_4 : Rozšírená Scafferova F6 funkcia

Hybridná funkcia 4

$N = 4$

$p = [0,2; 0,2; 0,3; 0,3]$

g_1 : HGBat funkcia

g_2 : Discus funkcia

g_3 : Rozšírená Griewankova plus Rosenbrockova funkcia

g_4 : Rastriginova funkcia

Hybridná funkcia 5

$N = 5$

$p = [0,1; 0,2; 0,2; 0,2; 0,3]$

g_1 : Rozšírená Scafferova F6 funkcia

g_2 : HGBat funkcia

g_3 : Rosenbrockova funkcia

g_4 : Modifikovaná Schwefelova funkcia

g_5 : Eliptická funkcia

Hybridná funkcia 6

$N = 5$

$p = [0,1; 0,2; 0,2; 0,2; 0,3]$

g_1 : Katsuura funkcia

g_2 : HappyCat funkcia

g_3 : Rozšírená Griewankova plus Rosenbrockova funkcia

g_4 : Modifikovaná Schwefelova funkcia

g_5 : Ackleyho funkcia

Kompozitné funkcie

Kompozitné funkcie sú počítané podľa vzťahu:

$$F(x) = \sum_{i=1}^N \{\omega_i * [\lambda_i g_i(x) + bias_i]\} + F^* \quad (49)$$

kde $F(x)$ – je kompozitná funkcia

$g_i(x)$ – je i -ta funkcia použitá na zostrojenie kompozitnej funkcie

N – je počet základných funkcií

$bias_i$ – definuje, ktoré optimum je globálne

o_i – je nová posunutá optimálna pozícia pre každú funkciu $g_i(x)$

σ_i – určuje rozsah každej funkcie $g_i(x)$

λ_i – určuje výšku každej funkcie $g_i(x)$

ω_i – je normalizovaná váha určená vzťahom $\omega_i = w_i / \sum_{i=1}^n w_i$

w_i – je hodnota váhy určená pre každú funkciu $g_i(x)$ nasledovne:

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^D (x_j - o_{ij})^2}} \exp\left(-\frac{\sum_{j=1}^D (x_j - o_{ij})^2}{2D\sigma_i^2}\right) \quad (50)$$

$$\text{Ak } x = o_i, \omega_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \text{ pre } j = 1, 2, \dots, N, f(x) = \text{bias}_i + f^*$$

Lokálne optimum, ktoré má menšiu bias hodnotu, je globálnym minimom. Globálne minimum všetkých kompozitných funkcií je rovné hodnote 0.

Kompozitná funkcia 1

$$N = 5, \quad \sigma = [10; 20; 30; 40; 50]$$

$$\lambda = [1; 1e - 6; 1e - 26; 1e - 6; 1e - 6]$$

$$\text{bias} = [0; 100; 200; 300; 400]$$

g_1 : Rosenbrockova funkcia s rotáciou

g_2 : Eliptická funkcia

g_3 : Bent Cigar funkcia s rotáciou

g_4 : Discus funkcia s rotáciou

g_5 : Eliptická funkcia

Kompozitná funkcia 2

$$N = 3, \quad \sigma = [20; 20; 20]$$

$$\lambda = [1; 1; 1]$$

$$\text{bias} = [0; 100; 200]$$

g_1 : Modifikovaná Schwefelova funkcia

g_2 : Rastriginova funkcia s rotáciou

g_3 : HGBat funkcia s rotáciou

Kompozitná funkcia 3

$$N = 3, \quad \sigma = [10; 30; 50]$$

$$\lambda = [0,25; 1; 1e - 7]$$

$$\text{bias} = [0; 100; 200]$$

g_1 : Modifikovaná Schwefelova funkcia s rotáciou

g_2 : Rastriginova funkcia s rotáciou

g_3 : Eliptická funkcia s rotáciou

Kompozitná funkcia 4

$N = 5$, $\sigma = [10; 10; 10; 10; 10]$

$\lambda = [0,25; 1; 1e - 7; 2,5; 10]$

$bias = [0; 100; 200; 300; 400]$

g_1 : Modifikovaná Schwefelova funkcia s rotáciou

g_2 : HappyCat funkcia s rotáciou

g_3 : Eliptická funkcia s rotáciou

g_4 : Weierstrassova funkcia s rotáciou

g_5 : Griewankova funkcia s rotáciou

Kompozitná funkcia 5

$N = 5$, $\sigma = [10; 10; 10; 20; 20]$

$\lambda = [10; 10; 2,5; 25; 1e - 6]$

$bias = [0; 100; 200; 300; 400]$

g_1 : HGBat funkcia s rotáciou

g_2 : Rastriginova funkcia s rotáciou

g_3 : Modifikovaná Schwefelova funkcia s rotáciou

g_4 : Weierstrassova funkcia s rotáciou

g_5 : Eliptická funkcia s rotáciou

Kompozitná funkcia 6

$N = 5$, $\sigma = [10; 20; 30; 40; 50]$

$\lambda = [2,5; 10; 2,5; 5e - 4; 1e - 6]$

$bias = [0; 100; 200; 300; 400]$

g_1 : Rozšírená Griewankova plus Rosenbrockova funkcia s rotáciou

g_2 : HappyCat funkcia s rotáciou

g_3 : Modifikovaná Schwefelova funkcia s rotáciou

g_4 : Rozšírená Scafferova F6 funkcia s rotáciou

g_5 : Eliptická funkcia s rotáciou

Kompozitná funkcia 7

$N = 3, \sigma = [10; 30; 50]$

$\lambda = [1; 1; 1]$

$bias = [0; 100; 200]$

g_1 : Hybridná funkcia 1

g_2 : Hybridná funkcia 2

g_3 : Hybridná funkcia 3

Kompozitná funkcia 8

$N = 3, \sigma = [10; 30; 50]$

$\lambda = [1; 1; 1]$

$bias = [0; 100; 200]$

g_1 : Hybridná funkcia 4

g_2 : Hybridná funkcia 5

g_3 : Hybridná funkcia 6