

Evoluční Bee Algoritmus (včelí algoritmus) v prostředí Mathematica

Evolutionary Bee Algorithm in the Mathematica Environment

Bc. Kristýna Němečková

Diplomová práce
2013



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Kristýna Němečková**
Osobní číslo: **A11408**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **prezenční**

Téma práce: **Evoluční Bee Algoritmus (včelí algoritmus)
v prostředí Mathematica**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Naprogramujte Bee Algoritmus v prostředí Mathematica.
3. Otestujte algoritmus na sadě vybraných testovacích funkcí.
4. Výsledky testování přehledně graficky a tabulkově zobrazte.
5. Vybrané dosažené výsledky porovnejte s libovolným jiným evolučním algoritmem.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: tištěná/elektronická

Seznam odborné literatury:

1. KENNEDY, James, Russell C EBERHART a Yuhui SHI. Swarm intelligence. San Francisco: Morgan Kaufmann, 2001, xxvii, 512 s. ISBN 15-586-0595-9.
2. BONABEAU, E. Swarm intelligence. New York: Univerzity Press, 1999, 307 s. ISBN 01-951-3159-2.
3. Parameter settings in evolutionary algorithms. Editor Fernando G Lobo, Cláudio F Lima, Zbigniew Michalewicz. Berlin: Springer, 2007, xii, 317 s. ISBN 978-3-540-69431-1.
4. DE JONG, Kenneth A. Evolutionary computation: a unified approach. Cambridge: MIT Press, 2006, ix, 256 s. ISBN 02-620-4194-4.
5. YANG, Xin-She. Introduction to computational mathematics. Hackensack, N.J.: World Scientific Pub., 2008, xi, 245 p. ISBN 98-128-1817-0.
6. MEHROTRA, Kishan. Elements of artificial neural networks. Cambridge, Mass.: MIT Press, 1997, xiv, 344 s. ISBN 978-0-262-13328-9.
7. ZELINKA, Ivan. Umělá inteligence v problémech globální optimalizace. BEN, 2002, 190 s. ISBN 80-7300-069-5.
8. ZELINKA I., OPLATKOVÁ Z., ŠEDA M., OŠMERA P., VČELAŘ F., Evoluční výpočetní techniky – principy a aplikace, BEN, Praha, 2008, ISBN 80-7300-218-3.

Vedoucí diplomové práce:

Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

22. února 2013

Termín odevzdání diplomové práce:

22. května 2013

Ve Zlíně dne 22. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan



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

ABSTRAKT

Tato diplomová práce se zabývá analýzou a testováním včelího algoritmu (Bee Algorithm). Tento evoluční algoritmus je v rámci diplomové práce porovnán s jiným evolučním algoritmem hejnového typu SOMA. Toto porovnání je provedeno na šesti různých standardních testovacích funkcích. V rámci teoretické části jsou vysvětleny základní pojmy z umělé inteligence a prvků evolučních výpočetních technik. Převážná část práce se věnuje interpretaci včelího algoritmu, respektive algoritmů vycházejících z principů chování biologických včel. Samotný algoritmus a veškeré přehledné grafy a tabulky jsou zpracovány v software Wolfram Mathematica.

Klíčová slova: Včelí algoritmus, testování, SOMA, Mathematica, Wolfram

ABSTRACT

This thesis is focused on the analysis and testing of the bee algorithm (Bee Algorithm). The evolutionary algorithm is within the thesis compared with other swarm intelligence type algorithm SOMA. This comparison is performed on six different standard test functions. Within the theoretical section, there are explained the basic concepts of artificial intelligence and evolutionary computation elements. Most of the work deals with the interpreting of the bee algorithm, respectively algorithms based on principles of biological behavior of bees. The algorithm itself and all illustrative graphs and tables are processed in the software Wolfram Mathematica.

Keywords: Bee Algorithm, testing, SOMA, Mathematica, Wolfram

Ráda bych poděkovala vedoucímu mé diplomové práce, panu Ing. Romanu Šenkeříkovi, Ph.D. za odbornou pomoc, cenné připomínky a ochotu pomoci k urychlení této práce. Dále bych dala svůj dík rodině, bez které by toto studium nebylo možné, svému příteli za pevné nervy a všem svým přátelům za psychickou podporu.

„Alea iacta est“

Gaius Julius Caesar

Prohlašuji, že

- beru na vědomí, že odevzdáním 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 bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk 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 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 – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování 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 bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem 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 bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD	10
I TEORETICKÁ ČÁST	12
1 ZÁKLADY UMĚLÉ INTELIGENCE	13
1.1 HISTORIE	13
1.2 NEURONOVÁ SÍŤ	14
1.3 APLIKACE UMĚLÉ INTELIGENCE.....	15
1.4 OPTIMALIZACE A ÚČELOVÉ FUNKCE.....	15
1.4.1 Stochastické algoritmy	16
2 SOFT COMPUTING	17
2.1 FUZZY LOGIKA	17
2.2 ÚVOD DO EVOLUČNÍCH ALGORITMŮ	18
2.3 METAHEURISTIKA	19
3 VYBRANÉ EVOLUČNÍ ALGORITMY	21
3.1 GENETICKÉ ALGORITMY	21
3.2 DIFERENCIÁLNÍ EVOLUCE	21
3.3 SOMA.....	22
3.3.1 Potomci	22
3.3.2 Parametry	22
3.3.3 Mutace a křížení	23
3.3.4 Princip SOMA	24
4 INTELIGENCE HEJNA (SWARM INTELLIGENCE)	25
4.1 OPTIMALIZACE HEJNEM ČÁSTIC (PARTICLE SWARM OPTIMIZATION)	25
4.1.1 Princip PSO	26
4.2 OPTIMALIZACE MRAVENČÍ KOLONIÍ (ANT COLONY OPTIMIZATION)	27
4.2.1 Princip ACO	27
5 OPTIMALIZACE VČELÍM ROJEM.....	29
5.1 BIOLOGICKÉ VČELY	29
5.1.1 Včelí tanec	30
5.1.2 Chování včel	30
5.2 UMĚLÉ VČELY	32
5.2.1 Studie o umělých včelích systémech	32
5.3 UMĚLÁ VČELÍ KOLONIE (ARTIFICIAL BEE COLONY).....	34
5.3.1 Životní cyklus algoritmu.....	35
5.3.2 Inicializační fáze.....	36
5.3.3 Fáze dělnic.....	36
5.3.4 Fáze vyčkávacích včel.....	36
5.3.5 Fáze skautů.....	37

5.3.6	Příklad ABC	38
5.4	VČELÍ ALGORITHMUS (BEE ALGORITHM).....	41
5.4.1	Životní cyklus algoritmu.....	41
5.4.2	Příklad BA.....	42
5.5	OPTIMALIZACE VČELÍCH KOLONIÍ (BEE COLONY OPTIMIZATION).....	45
5.5.1	Životní cyklus algoritmu.....	46
5.6	VČELÍ SYSTÉM (BEE SYSTEM).....	46
II	PRAKTICKÁ ČÁST.....	48
6	MATHEMATICA	49
7	TESTOVACÍ FUNKCE.....	50
7.1	CHARAKTERISTIKA TESTU.....	50
7.2	VYBRANÉ TESTOVACÍ FUNKCE.....	50
7.2.1	1st De Jong.....	51
7.2.2	2nd De Jong.....	51
7.2.3	Schwefel.....	52
7.2.4	Rastrigin	53
7.2.5	Shifted 1st De Jong.....	54
7.2.6	Shifted Rastrigin	55
8	IMPLEMENTACE VČELÍHO ALGORITMU.....	57
8.1	ROZBOR FUNKCÍ.....	57
8.2	ADAPTIVE BEE ALGORITHM.....	57
9	VÝSLEDKY TESTOVÁNÍ.....	59
9.1	1ST DE JONG	61
9.1.1	5. dimenze	62
9.1.2	30. dimenze.....	63
9.2	2ND DE JONG	64
9.2.1	5. dimenze	65
9.2.2	30. dimenze.....	66
9.3	SCHWEFEL	67
9.3.1	5. dimenze	68
9.3.2	30. dimenze.....	69
9.4	RASTRIGIN	70
9.4.1	5. dimenze	71
9.4.2	30. dimenze.....	72
9.5	SHIFTED 1ST DE JONG.....	73
9.5.1	5. dimenze	74
9.5.2	30. dimenze.....	75
9.6	SHIFTED RASTRIGIN	76
9.6.1	5. dimenze	77
9.6.2	30. dimenze.....	78
	ZÁVĚR	79

ZÁVĚR V ANGLIČTINĚ.....	81
SEZNAM POUŽITÉ LITERATURY.....	83
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	87
SEZNAM OBRÁZKŮ.....	88
SEZNAM TABULEK	90
SEZNAM PŘÍLOH	91

ÚVOD

Tato diplomová práce převážně pojednává o algoritmech, které jsou inspirovány reálnými jevy v přírodě. Jsou to právě algoritmy, které vycházejí ze skutečných vlastností biologických včel. Převážná část práce se pak věnuje evolučnímu včelímu algoritmu a jeho testování. Jelikož včelí algoritmus patří do umělé inteligence, jsou v teoretické části zmíněny základy umělé inteligence a s ní spojená historie, neuronová síť, aplikace atd. Dále je zmíněn i soft computing, jenž se skládá, mimo jiné, z evolučních výpočtů, a s ním spojené evoluční algoritmy.

Jsou popsány některé z evolučních algoritmů, v tomto případě genetické algoritmy, diferenciální evoluce a SOMA. Větší prostor má SOMA, a to díky (z důvodu) testování a porovnání s včelím algoritmem.

Další část se věnuje *Swarm Intelligence*, kam se, mimo jiné, řadí i včelí algoritmus. Jsou zmíněni tři základní představitelé *Swarm Intelligence*: Optimalizace hejnem částic, Optimalizace mravenčí kolonií a Optimalizace včelím rojem. Tato včelí optimalizace bude shrnuta do jedné kapitoly, kde budou popsány algoritmy vycházející právě z chování skutečných včel pátrajících po potravě.

V návaznosti na optimalizaci včelím rojem jsou popsány rozdíly mezi biologickými a umělými včelami. Dále jsou rozebrány algoritmy, které se zabývají touto tematikou, jako jsou Umělá včelí kolonie, Včelí algoritmus, Optimalizace včelí kolonií a Včelí systém. Nejvíce prostoru dostanou Umělá včelí kolonie a Včelí algoritmus, se kterým se bude pracovat po zbytek této práce.

Včelí algoritmus bude naprogramován v softwaru Wolfram Mathematica, a proto je vhodné v jedné kapitole tento software uvést a charakterizovat. K testování včelího algoritmu a algoritmu SOMA je použito šest standardních testovacích funkcí. Každá z funkcí je představena a vykreslena k názornějšímu pochopení funkce. Mezi těchto šest testovacích funkcí patří funkce 1st De Jong, 2nd De Jong, Rastrigin, Schwefel, shifted 1st De Jong a shifted Rastrigin. Pro snadnější porozumění testování bude v kapitole Charakteristika testu blíže vysvětlen jeho princip. Každá testovací funkce je testována v šesti různých dimenzích. Tyto dimenze jsou 2D, 5D, 10D, 20D, 30D a 50D.

Ke každé testovací funkci je zobrazena přehledná tabulka výsledků obou testovaných algoritmů a grafy znázorňující průběhy výpočtů pro 5D a 30D. Grafy a tabulka budou rovněž vytvořeny programem Wolfram Mathematica.

I. TEORETICKÁ ČÁST

1 ZÁKLADY UMĚLÉ INTELIGENCE

Člověk se již od pradávna snaží vytvořit umělou inteligentní bytost k vykonávání prací a usnadnění bytí člověka.

Co to je vůbec inteligence? Definicí tohoto pojmu je vícero, dle A. Turinga lze říci, že „Bude-li stroj reagovat na podněty lidského partnera takovým způsobem, že člověk není schopen rozeznat, zda jedná se strojem či s osobou prostřednictvím terminálu, lze jej považovat za inteligentní.“ [1].

Umělá inteligence je označení vědní disciplíny pro všechny takové postupy a algoritmy, které vedou k určitému napodobení projevů inteligentního chování člověka a patří mezi jednu z nejrychleji se vyvíjejících vědních a technických disciplín.

Většina metod umělé inteligence obecně nezaručuje přesnost nalezeného řešení. Použití umělé inteligence není proto vhodné v případech, kdy je nutno zajistit plnou spolehlivost, nebo je-li k dispozici existující deterministická metoda řešení úlohy [2].

1.1 Historie

Počátky prací na neuronových sítích vychází z neurobiologických studií, které se datují do doby něco před sto lety. Mnoho dekad se spekulovalo, jak přesně nervové systémy pracují.

Otázky, které řeší, jak mezi sebou neuronové buňky komunikují, jaký práh elektrického podnětu potřebují k aktivaci a jak se jednotlivé buňky chovají, byly zodpovězeny až v polovině dvacátého století rozvojem neurologie jako vědy.

První matematický model jednoduchého neuronu byl připočítán Američanu W. S. McCullochu a jeho studentovi W. Pittsu. Tento model se prakticky používá dodnes. Na jeho práci navázal v roce 1958 F. Rosenblatt, jenž vytvořil první perceptronovou síť. Zkoumání neuronových sítí bylo od 60. let pozastaveno, neboť se v té době domnívalo, že neuronové sítě nemají budoucnost. Až od poloviny 80. let se díky několika průkopníkům opět nastartoval divoký rozvoj. Došlo k renesanci neuronových sítí, jejichž význam a užití stoupá každým dnem, viz [3] a [4].

1.2 Neuronová síť

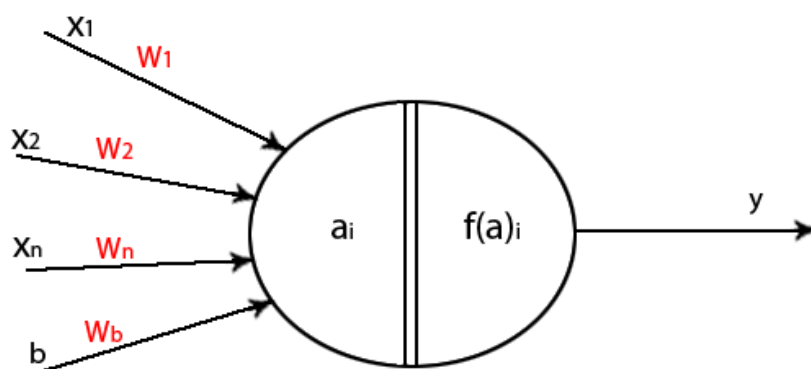
Neuronová síť v umělé inteligenci matematicky odpovídá neuronové síti živých organismů (tzv. matematický model).

Snaží se napodobovat funkce nervových soustav a mozků živých organismů (biologického neuronu). Dá se říci, že mozek je synonymem gigantického počítače.

Umělý neuron vznikl na základě biologického neuronu a rovněž obsahuje dendrity (několik vstupů), axon (1 výstup), synapse (váhy) – Obr. 1 je zobrazen umělý neuron.

Odumře-li neuron, v lidském těle probíhá tzv. reorganizace neuronové sítě. Nedochází tedy k nahrazení, ale pouze k přepojení (navazování nových spojů) na jinou živou neuronovou buňku [5]. Tomuto procesu říkáme učení.

Ke správnému učení je důležité vhodně sestavit trénovací množinu, která zaručuje úspěch či neúspěch neuronové sítě [6].



Obrázek 1 - Umělý neuron

Kde

- X_i – vstupy
- W_i – váhy
- b – práh
- a_i – aktivační funkce
- $f(a)_i$ – přenosová funkce. Je potřeba se zamyslet nad typem přenosové funkce a k tomu charakteristickým typem dat.

Algoritmus učení:

- S učitelem – je snaha nastavit váhy tak, aby se dostavil požadovaný výstup.
- Bez učitele – v rámci samo organizace si vytváří třídy sama a snaží se seskupovat podobné výsledky.

Neuronové sítě lze využít např. při rozpoznání písma, rozpoznání mince vložené do automatu podle váhy a průměru, rozpoznání nepřátelských letadel podle radarového snímku aj. [7].

1.3 Aplikace umělé inteligence

Umělá inteligence se uplatňuje v mnoha odvětvích. Lze se s ní setkat například v ekonomické sféře v oblasti marketingu při identifikaci preferencí zákazníků, v lékařské sféře, vojenských systémech apod. [5].

Neuronové sítě je možno využít při predikci časových řad na burze nebo erupcí na Slunci, při rozpoznávání písma, klasifikaci, optimalizaci, autoasociaci, stegoanalýzy, optimalizaci letadla a mnoha jiném (detekce spamu, detekce pohybu, rozpoznávání komprese obrazů, písma, zvuků...) [4].

Podle [2], co se týče výhod a nevýhod, je na tom umělá inteligence následovně:

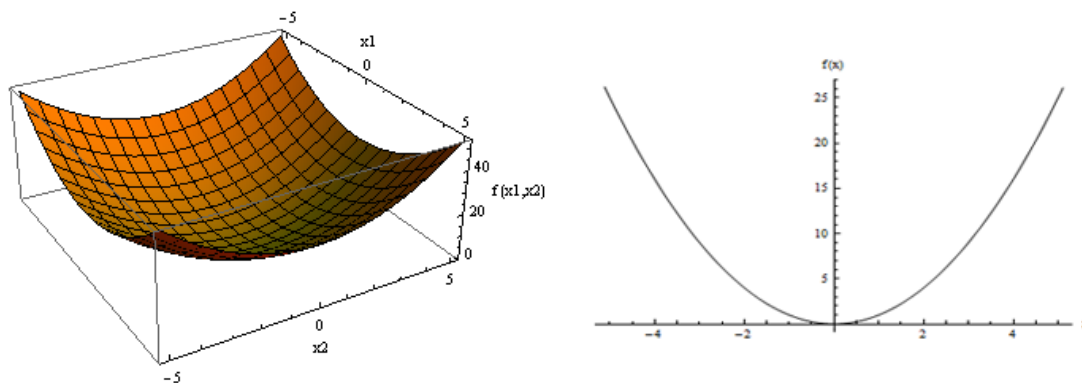
- Výhody: získání lepších výsledků, nezávislost na počátečních podmínkách.
- Nevýhody: není zaručeno, že bude řešení lepší a zda bude vůbec nalezeno, za další může být trénování sítě obtížné, ne-li nemožné.

1.4 Optimalizace a účelové funkce

Jak je uvedeno v [8], optimalizace je proces, který vede k nalezení vhodné účelové funkce. Existují ovšem jisté komplikace, které plynou z optimalizace některých problémů. Můžeme zmínit například příliš velký prostor možných řešení nebo složitost samotného problému (matematický model neodpovídá reálnému) apod.

Účelová funkce je funkce, jejíž optimalizace (nalezení minima či maxima) povede k nalezení optimálních hodnot jejích argumentů. Účelovou funkci můžeme promítnout do geometrické plochy a zobrazit si minimum/maximum ležící v $N+1$ rozměrném prostoru (prostor možných

řešení daného problému). Například na Obrázku 2 je zobrazena testovací funkce 1st De Jong, jejíž předpis je $\sum_{i=1}^D x_i^2$. O testovacích funkcích se dá dozvědět více v kapitole 7.



Obrázek 2 - Zobrazení testovací funkce 1st De Jong ve 3D a 2D

1.4.1 Stochastické algoritmy

Nemožnost nalézt deterministický algoritmus, který řeší úlohu globální optimalizace obecně, vedla k využití stochastických algoritmů, jenž sice nemohou garantovat nalezení řešení v konečném počtu kroků, ale často pomohou nalézt prakticky použitelné řešení v přijatelném čase.

Stochastické algoritmy využívají heuristiky, tzn. postup prohledávání prostoru, ve kterém se využívá náhoda, intuice, analogie a zkušenost.

Hlavní rozdíl mezi heuristikou a deterministickými algoritmy je ten, že heuristika nezaručuje nalezení řešení.

V posledních desetiletích se pro hledání globálního minima funkcí se značným úspěchem využívají stochastické algoritmy, zejména evolučního typu (viz dále) [9].

2 SOFT COMPUTING

Rozhodujícím aspektem umělé inteligence a soft computingu je řešení komplexních problémů, jako je najít správnou rovnováhu mezi výpočetní složitostí a aproximační přesností [10].

Soft computing je nový multidisciplinární obor, který navrhl Dr. Lotfi Asker Zadeh, jehož cílem bylo vytvořit novou generaci umělé inteligence, známou jako výpočetní inteligenci. Myšlenka a současný vývoj soft computingu vznikl v roce 1981. Dr. Zadeh definoval soft computing jako fúzi oblastí fuzzy logiky, neuro-výpočtů (metaheuristiky a inteligence hejna), evolučních a genetických algoritmů (kapitola 3.1 Genetické algoritmy) a pravděpodobnostních výpočtů do jednoho multidisciplinárního systému. Později byl Soft computing spojen s dalšími oblastmi, jako je teorie chaosu a perceptron. Hlavním cílem soft computingu je vyvinout inteligentní stroje a řešit nelineární a matematicky obtížně se modelující problémy systému.

Obecně lze říci, že techniky soft computingu se podobají biologickým procesům lépe než tradiční techniky, které jsou z velké části založeny na formálních logických systémech, jako jsou větné logiky a predikátové logiky, nebo se spoléhají na počítačové podpory numerické analýzy. Techniky soft computingu jsou určeny k tomu, aby se vzájemně doplňovaly [11].

Na rozdíl od tradičního hard computingu je soft computing zaměřen na akceptování všudypřítomné nepřesnosti reálného světa. Základním principem je využití tolerance nepřesnosti, nejistoty a částečné pravdivosti pro dosažení ovladatelnosti, robustnosti, nízké ceny řešení a lepšího vztahu s realitou. Je možno říci, že soft computing nedá úplně přesné řešení, ale řešení, které je pro náš případ dostačující, kdežto hard computing by z časových důvodů ani takové řešení nenašel. Funkčním modelem soft computingu je lidská mysl [12].

2.1 Fuzzy logika

Fuzzy logika pracuje s vágními (nepřesnými) výrazy a je schopna je převádět do čísel, se kterými se pak provádí logické operace. Vede k výsledkům, k nimž by došla i uvažující inteligentní bytost.

Fuzzy logika je téměř synonymem k teorii fuzzy množin. Fuzzy množina je množina, která kromě úplného nebo žádného členství připouští i členství částečné (prvek patří do množiny s jistou pravděpodobností).

Dokáže zpracovávat nepřesnosti a poměrně jednoduchým způsobem umožňuje vyhodnocovat přirozený jazyk.

Zakladatel fuzzy logiky, Lotfi A. Zadeh, pronesl následující výrok o přesnosti: „*S rostoucí složitostí přesný výrok ztrácí smysl a smysluplný výrok přesnost*“ [13].

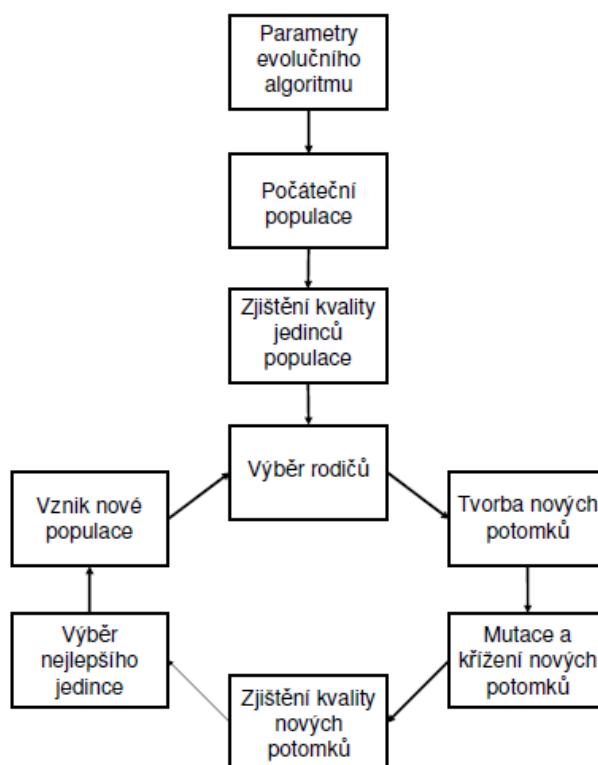
Klasické logické uvažování vyžaduje pouze hodnoty ano a ne, kdežto fuzzy logika může pracovat s pojmy jako možná, skoro, či velmi. Matematicky umožňuje vyjádřit pojmy jako trochu, dost nebo hodně [14].

2.2 Úvod do evolučních algoritmů

Dle Darwinovy evoluční teorie a Mendelových principů křížení (dědičnosti) se jednotlivé druhy (potomci) vyvíjejí z rodičů. Tito potomci při svém vzniku podléhají mutacím, kde se vybírají pouze ti nejvhodnější zmutovaní potomci (nebo i rodiče) a ti nejslabší (méně vhodní) zanikají a nepostupují do další generace (zanikají po tzv. generacích). Na Obrázku 3 je možno vidět obecný cyklus evolučních algoritmů [15].

Mutace je stejně důležitá vlastnost jako křížení a zajišťuje lepší nalezení globálního extrému. Pokud by mutace neprobíhala, je větší pravděpodobnost, že algoritmus uvázne v lokálním extrému. Algoritmus tzv. zdegeneruje.

Je také vhodné si uvědomit, že ne všechny algoritmy lze použít na veškerou třídu problémů. Každý z algoritmů je použitelný na specifické třídy problémů. Např. algoritmus ACO je nejlépe využitelný pro kombinatorické problémy typu obchodní cestující (více v kapitole 4.2 Optimalizace mravenčí kolonií), kdežto genetické algoritmy lze použít na poměrně širokou třídu problémů. Některé z evolučních algoritmů (dále jen EA) jsou uvedeny v kapitole 3. Vybrané evoluční algoritmy [9].



Obrázek 3 - Obecný cyklus evolučního algoritmu

Prohledávat celý prostor řešení je někdy nemožné a s tím je i spojena komplikace nalezení optimálního řešení. Protože se EA řadí mezi stochastické algoritmy, je jeho výhodnou schopnost nalezení globálního extrému na velké ploše řešení, kde využívá právě stochastických prvků. Další využití EA je u problémů, kdy je znám požadovaný výstup, ale již nejsou známy hodnoty na vstupu. EA dokáže tyto hodnoty bez problémů nalézt [16].

Na druhou stranu je nevýhodou neschopnost dopředu předvídat výsledky, právě díky zmiňované částečné práci s náhodou [17].

2.3 Metaheuristika

Ve vědě o počítačích metaheuristika označuje výpočetní metodu, která optimalizuje problém a snaží se iteračně zlepšit vhodného kandidáta řešení s ohledem na výslednou kvalitu. Při použití těchto metod nemusí být k dispozici žádné (nebo jen minimální) informace o problému, který je optimalizován, a je možné prohledávat velmi velké prostory kandidátských řešení. Nicméně, metaheuristika nezaručuje, že optimální řešení je vždy nalezeno. Mnoho metaheuristik provádí optimalizaci pomocí stochastických optimalizačních metod (do algoritmu je zanesen prvek náhody) [18].

Stochastické heuristické metody se někdy také označují jako metaheuristiky, protože poskytují pouze obecný rámec a vlastní operace algoritmu je třeba zvolit (např. operaci křížení a mutace u genetických algoritmů, operaci sousedství u simulovaného žihání a tzv. tabu-search atd.) v závislosti na zkoumaném problému. Protože tyto metody se často inspiřují přírodními procesy, jsou také nazývány evolučními algoritmy [8].

Stručně řečeno, metaheuristika představuje takové metody, které jsou schopny nalézt (třebaže ne nejlepší) vhodné řešení v únosném čase (př. problém obchodního cestujícího). U deterministických algoritmů při optimalizaci daného problému nejsme schopni nalézt ideální řešení v přiměřeném čase. Mezi problémy metaheuristik patří možnost uvážnutí v lokálním extrému. Některé algoritmy jsou úspěšnější v řešení dané úlohy než jiné.

3 VYBRANÉ EVOLUČNÍ ALGORITMY

Do této třídy se již řadí velké množství algoritmů inspirovaných se evoluční teorií, a proto zde budou popsány pouze vybrané algoritmy.

3.1 Genetické algoritmy

Patří do podmnožiny evolučních algoritmů a byly představeny Johnem Hollandem v roce 1975. Genetické algoritmy (dále jen GA) využívají Darwinovy teorie vývoje populace. Terminologie je převzata z biologie, jako je populace, jedinec, generace, mutace [8].

Jako jiné evoluční algoritmy slouží i GA k optimalizování úloh. Jak se píše v [19], mezi základní prvky GA patří chromozóm a gen. Právě v tomto se odlišuje od ostatních evolučních algoritmů. Chromozóm je řetězec informací, nejčastěji je to pozice v prostoru možných řešení, který je zakódován do binárního kódu. Nemusí se jednat o celé číslo, jde použít reálná čísla, matice, vektory i křivky. Oproti tomu gen je nejmenší část chromozómu.

GA používají selekci jako nástroj pro výběr rodičů z populace. Většinou se vybírá na základě hodnoty CV (dále jen Cost Value – hodnota účelové funkce). Mezi metody výběru patří například ruletový výběr, turnajová metoda, náhodný výběr aj. Jednou ze základních operací, kterou GA používají, je křížení. Jedná se o výměnu části chromozómů mezi rodiči (jednobodové či vícebodové). Další operací je mutace, kde se s určitou pravděpodobností zamění hodnoty některých genů v chromozómu (1 za 0 a obráceně) [19].

3.2 Diferenciální evoluce

Jak se píše v [15], diferenciální evoluce (dále jen DE) vznikla v roce 1995 a vyvinuli ji Ken Price a Rainer Storm. Mezi DE a GA je velká podobnost. Podobají se například ve tvorbě potomků (pomocí 4 rodičů), opakováním se v generacích apod.

Nastavení řídicích parametrů hraje velkou roli v kvalitě diferenciální evoluce.

Cílem je nalézt nejlepší populaci jedinců (hodnoty účelové funkce) v jednotlivých generacích. Postupuje se následovně:

- Stanovení parametrů
- Tvorba populace

- Započítí cyklu generace
- Evoluční cyklus
- Testování naplnění ukončovacích parametrů
- Vyhodnocení

3.3 SOMA

V roce 1999 vytvořil Ivan Zelinka, algoritmus SOMA (samo organizující se migrační algoritmus), jenž je založen podobně jako DE nebo PSO (Particle Swarm Optimization, kapitola 4.1 Optimalizace hejnem částic) na vektorových operacích.

Původní myšlenkou bylo napodobení chování skupiny inteligentních jedinců, kteří kooperují při řešení společného problému, jako je např. hledání zdroje potravy apod. Proto tento algoritmus můžeme zařadit mezi algoritmy memetické či hejnové (swarm intelligence). Může být rovněž klasifikován jako algoritmus evoluční, navzdory faktu, že nejsou vytvářeni žádní potomci v průběhu prohledávání plochy, ale pouze se mění jejich pozice, tím „vznikne“ nový potomek [15]. Dalším rozdílem mezi genetickým algoritmem a algoritmem SOMA je názvosloví jednoho evolučního cyklu. U genetických algoritmů se nazývá generace, kdežto SOMA používá migrační kolo (jedinci se přesouvají - migrují) [20] a [15].

Vysvětlení daných pojmů vychází z [15].

3.3.1 Potomci

Tvorba nových potomků (řešení) neprobíhá křížením rodičů, ale je založena na kooperativním prohledávání (migraci) prostoru možných řešení. V reálném světě lze najít kooperaci mezi mravenci, včelami nebo i smečkami.

3.3.2 Parametry

Tak jako DE má i SOMA své vlastní parametry, které ovlivňují běh algoritmu. Parametry jsou uvedeny v Tabulce 1.

V závislosti na nastavení parametrů je více či méně ovlivňována úspěšnost algoritmu. Parametr *Step* říká, jak dopodrobna se bude prozkoumávat daná cesta. Čím větší ohodnocení parametru, tím bude prohledávání kvalitnější (větší šance nalezení globálního

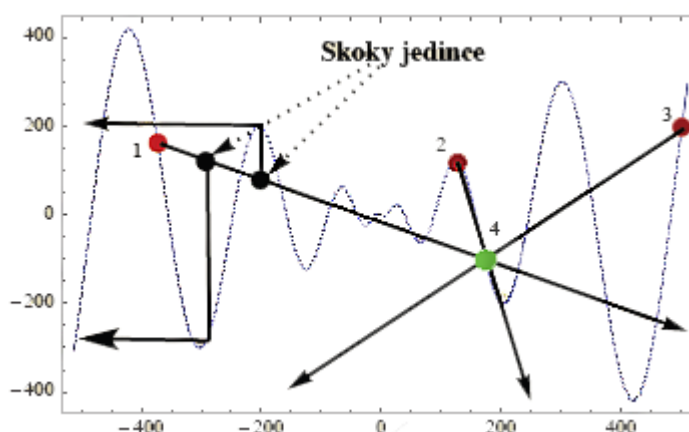
extrému), ovšem za cenu větší výpočetní náročnosti. Jedinec se pohybuje po diskrétních skocích.

Parametr *PRT* (perturbace) znamená, jakou měrou půjde aktivní jedinec přímo k vedoucímu jedinci (leader).

Parametr	Význam
PathLength	Jak daleko se zastaví aktivní jedinec od leadra. (před\za)
Step	S jakým krokem se jedinec bude pohybovat po poli.
PRT	Ovlivňuje, zda se jedinec bude pohybovat přímo k leadrovi či ne.
D	Počet argumentů účelové funkce.
PopSize	Kolik jedinců bude tvořit populaci.
Migrace	Migrace = generace, kolikrát se populace přesune.
MinDiv	Maximální rozdíl mezi nejlepším a nejhorším jedincem.

Tabulka 1 - Parametry algoritmu SOMA

Velikost populace se nastavuje na základě počtu argumentů účelové funkce. Při multimodální funkci (více extrémů) se doporučuje větší počet jedinců v populaci. Bude-li více jedinců v populaci, je větší šance, že alespoň některý z nich nalezne globální extrém. Jak je možno vidět na Obr. 4 (obrázek převzat z [17]) je nalezení globálního extrému obtížné, a proto je možno nastavit $PopSize = D$. U unimodálních funkcí se nastavuje menší počet jedinců.



Obrázek 4 - Chování SOMA – funkce Schwefel

3.3.3 Mutace a křížení

Mutace je proces, při kterém dochází k náhodné změně některých vlastností. Jak již bylo zmíněno, mutace je pro každý algoritmus důležitá, aby nedocházelo k jeho degeneraci. V SOMA způsobuje mutaci za následek parametr *PRT* a generátor náhodných čísel

(mutace = perturbace, viz předchozí kapitola). Na základě tohoto parametru se generuje perturbační vektor (*PRTVector*), který je pro každého jedince generován zvlášť, a je platný jen pro jeden aktuální běh aktivního jedince.

V kapitole 3.3.2 bylo zmíněno, že každý jedinec se pohybuje po ploše v diskrétních skocích (parametr *Step*). Pro každý skok se počítá nová hodnota účelové funkce (teoreticky, z rodiče každým skokem vzniká více potomků). Na základě této nejlepší hodnoty je s danými parametry vytvořen nový jedinec, hodnoty v ostatních skocích (ty horší) se neberou v úvahu.

3.3.4 Princip SOMA

Zde bude ve stručnosti popsáno, jak algoritmus SOMA funguje.

Nejdříve se náhodně vygenerují jedinci na prohledávané ploše. Pro každého z těchto jedinců se vypočítá hodnota účelové funkce. Ten nejlepší jedinec se označí za leadera. Na základě parametrů *Step*, *PathLength*, *PRT* se jedinci pohnou po ploše (většinou směrem k leaderovi) a opět se určí nejlepší jedinec v daném kole. Tomuto jednomu kolu se říká migrace. Tento postup se opakuje tak dlouho, dokud se nedosáhne maximálního počtu migrací nebo dokud jej nezastaví parametr *MinDiv*.

Popsaný postup je pro strategii All to One. Strategii je více, například All to All, All to All Adaptive atd. Záleží na volbě uživatele.

O algoritmu SOMA se dá říct, že ve smyslu nalezení globálního extrému se vyrovná mnohým evolučním algoritmům.

4 INTELIGENCE HEJNA (SWARM INTELLIGENCE)

Intelligentní roj (též hejno) je skupina využívající kolektivní inteligenci jednoduchých samostatných agentů (subjektů). Tito agenti se nechávají ovlivňovat svým okolím, které pravděpodobně obsahuje i jiné agenty, ale přesto jednají nezávisle. Nedodržují příkazy žádného vůdce nebo se nedrží nějakého globálního plánu [21].

Právě touto inteligencí lze řešit rozdílné problémy, jejich řešení se inspirované v hmyzu žijícím ve společenství. Například mravenci, včely, ptáci, atd.

Samotný mravenec nebo včela se nepovažují za inteligentní jedince, zato jejich kolonie již inteligentní chování vykazují. Studie inteligence hejna poskytují poznatky, kterých mohou lidé využít k řízení složitých systémů od jednoduchého řízení vozíku až po vojenské roboty. Tato disciplína se rovněž využívá v optimalizaci, zejména v telekomunikačních systémech nebo v dopravě.

A jak inteligence hejna/roje funguje? Jednoduchá stvoření se řídí jednoduchými pravidly. Nikdo nikomu neříká, co viděl nebo co má dělat. Některé druhy jsou sice sofistikovanější než druhé, ale faktem je, že není nutné žádné vedení. Dokonce i složité chování může být řízeno pomocí relativně jednoduchých interakcí [22].

Při pozorování kolonie hejna podle [23] zjistíme jeho:

- Flexibilita – kolonie může reagovat na vnitřní odlišnosti a vnější změny.
- Robustnost – úlohy jsou splněny, i když se některým jedincům nedaří.
- Decentralizovanost – neexistuje žádné centrální řízení v kolonii.
- Samo-organizovanost – cesty k řešení nejsou předdefinované.

V následujících podkapitolách jsou uvedeni někteří zástupci hejnové inteligence. Včelímu algoritmu je věnována celá jedna kapitola.

4.1 Optimalizace hejnem částic (Particle Swarm Optimization)

PSO neboli Particle Swarm Optimization (optimalizace hejnem částic, rojení částic, dále již PSO) byla vyvinuta v roce 1995 Russellem Eberhartem a Jamesem Kennedym.

Tato metoda je založena na chování společenstev včel, ptáků nebo například ryb. Jedná se o kolektivní chování jednotlivců ve vztahu s jejich okolím a se sebou navzájem. A proto ji nazýváme inteligencí hejna nebo roje [24].

Stejně jako evoluční výpočetní techniky je i PSO založena na hledání optimálního řešení pomocí populace jedinců (nalezená řešení). Cílem je nalézt co nejlepší populaci (generaci), nejlepší hodnoty účelové funkce. Na rozdíl od genetických algoritmů ale nemá žádné evoluční operátory, jako je mutace a křížení. Potenciální řešení, částice (jedinci), následují v řešeném prostoru ty částice, které jsou právě optimální [25].

Díváme-li se na chování hejna ptáků, mnohdy nám jejich pohyby připadají jako velmi dobré taneční choreografie. Jak jsou tedy schopni tak elegantně koordinovat své pohyby? Na tuto otázku se snaží odpovědět biologové a vědci zabývající se umělou inteligencí.

Mezi výhody patří (zmíněny jsou jen některé z nich v [24]):

- Rychlé řešení při velkém počtu argumentů účelové funkce.
- Rozlehlý prohledávací prostor není problém.
- Nezávislý na počátečních podmínkách.

Nevýhody:

- Neručí nalezení optimálního řešení.
- Velké množství nastavitelných parametrů.

4.1.1 Princip PSO

PSO algoritmus simuluje chování ptačího hejna. Pokud tedy skupina ptáků prohledává oblast a pátrá po nejvyšším vrcholku, tak neví, kde vrchol je, ale po každé iteraci ví, kdo našel zatím nejvyšší místo. Čili nejlepší strategií by bylo následovat toho jedince, který je v nejvyšší nadmořské výšce.

PSO tuto strategii využívá a využívá ji k řešení optimalizačních problémů. Každé jednotlivé řešení je jako jedinec ve vyhledávacím prostoru [24].

Nejlepší nalezená pozice (CV) se uloží do společné paměti populace. Ostatní pak ví, kde se nejlepší pozice nachází. Dále si každý z jedinců prohledávající prostor uloží svou nejlepší dosud nalezenou pozici k sobě. Ukládá se i nejlepší řešení definované v sousedství. Právě s těmito proměnnými se pracuje a každá částice si počítá svou novou rychlost a pozici.

Postup dle [26] je následující:

- Inicializace populace v prohledávaném prostoru (náhodná pozice a rychlost).
- Výpočet CV u každého jedince.
- Změnit rychlost na základě předchozí nejlepší globální pozice nebo nejlepší pozice v sousedství a nejlepší pozice jedince.
- Přesunout se na novou pozici.
- Pokračovat od kroku 2.

4.2 Optimalizace mravenčí kolonií (Ant Colony Optimization)

V originále nazýván Ant Colony Optimization (optimalizace mravenčí kolonií, dále jen ACO). Jak již název sám napovídá, algoritmus se inspiroje principy spolupráce jedinců patřících ke stejné mravenčí kolonii. Původně navržen pro řešení složitých kombinatorických optimalizačních problémů. V případě problému obchodního cestujícího podává algoritmus excelentní výsledky.

V této práci již bylo zmíněno, že pouhý jedinec nic nezmůže. Oproti tomu celá kolonie dokáže obrovské věci. Z dostupných informací víme, že za určitých okolností můžou některé druhy mravenců usmrtit i člověka. Taková je síla mravenčí kolonie.

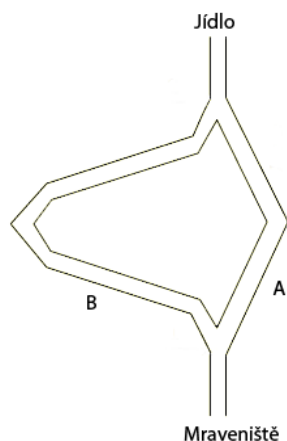
Kolonie může řešit problémy, které jsou nemyslitelné pro jednotlivce, jako například hledání nejlepšího zdroje potravy a její nejkratší cesty, rozdělení jedinců na různé úkoly, nebo bránit území od sousedních nebezpečí. Právě inteligence roje umožňuje kolonii rychle a účinně reagovat na své okolí [22].

4.2.1 Princip ACO

V algoritmu ACO bylo snahou napodobit chování reálných mravenců těmi umělými (jedinci v populaci). Díky jejich vzájemné spolupráci a samo-organizaci lze snadněji nalézt optimální řešení.

A jak tedy mravenci postupují? Každý mravenec produkuje feromony, což jsou látky, podle kterých se řídí ostatní jedinci. Vede-li k potravě více cest, tak si mravenci zvolí cestu náhodně. Jak je vidět na Obr. 5, lze si zvolit mezi cestami A a B (cesta A je vhodnější, kratší). Mravenec z kratší cesty dorazí k jídlu první, kde si stejnou cestu zpět začne

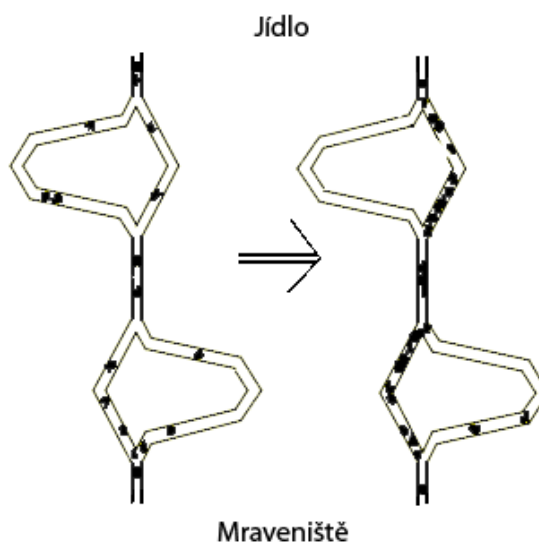
značkovat feromonem. Ostatní mravenci jsou ovlivněni ve prospěch této cesty. Čím víc je daná cesta projita, tím víc je na ní zanechaných feromonů a větší pravděpodobnost, že ostatní jedinci si vyberou právě tuto cestu.



Obrázek 5 - Cesta k potravě bez mravenců

Pomalejší mravenec (delší cesta) konečně dorazí k jídlu a cestou zpět si již vybere tu vhodnější cestu díky většímu množství feromonů. Takovýmto způsobem jednají ostatní jedinci a po nějakém čase je nalezena optimální cesta (viz Obr. 6).

Podle [17] je v ACO algoritmu feromon zastoupen vahou, která je přiřazena k dané cestě. Pro větší účinnost algoritmu se zohledňuje tzv. vypařování feromonů u mravenců. Tento fakt je zastoupen slábnoucími vahami. Díky této skutečnosti je algoritmus robustní a má větší pravděpodobnost nalezení globálního extrému.



Obrázek 6 - Optimalizovaná cesta mravenci

5 OPTIMALIZACE VČELÍM ROJEM

Tato metoda zahrnuje dva základní způsoby, jak algoritmy dělit z hlediska podobnosti s biologickými včelami. A to dle rozmnožování a shánění potravy.

Algoritmus se vyznačuje jednoduchostí stejně jako PSO nebo DE. Aby dosáhl nejlepšího řešení, je třeba si pohrát s nastavením parametrů. Jednotlivé parametry budou rozepsány až v příslušných podkapitolách.

Jako předchozí případy je i včelí algoritmus inspirován chováním skutečných včel. Dříve, než bude vysvětlena implementace včelího algoritmu, je vhodné vysvětlit, jakým způsobem včely pracují a jak se chovají.

5.1 Biologické včely

Včela medonosná je jedním z nejstarších živočichů využívaných člověkem a patří mezi nejznámější zástupce společenského hmyzu. Včela a informace o jejím chovu se objevují již na sumerských tabulkách či v textech starých Egyptů.

Jsou důležitou součástí našich životů, prospívají převážně sadařům, zemědělcům a včelařům. Je to také jediná chovaná skupina zvířat, kterým se pro užitek berou zásoby krmiva. Mezi jejich hlavní náplň života patří opylování květů (ze 100 druhů užitečných plodin opylují kolem 70 %) a tvorba medu a vosku.

Včely žijící v přírodě se dělí na matky, dělnice a trubce. Pro přežití včel je důležitá dělba práce. Každá včela má na starosti svůj specifický úkol, kde by jedna bez druhé nepřežily.

Pro tento konkrétní případ umělých včel se budou brát do úvahy pouze dělnice, ty se zabývají mimo jiné, procesem shánění potravy [27], [28].

Po dlouhou dobu nebylo známo, jak včely komunikují. Mezi první teorie patřilo řízení veškeré společnosti královnou. Tato teorie se nepotvrdila a ihned následovala další teorie o naprogramovaných algoritmech, kterými se jednotlivé včely řídí. I tato teorie byla mylná. Ve 20. letech 20. století upozornil rakouský vědec Karl von Frisch na souvislost včelích tanců s předáváním informace o zdroji potravy ostatním obyvatelkám úlu. Svou domněnku však Frisch s konečnou platností potvrdil až r. 1943. Ukázal zároveň, podle jakého klíče je možné z tanečních kreací létavky zjistit, kde je spousta květů s množstvím sladoučkého

nektaru. Důležitost tohoto objevu ukazuje i to, že za svou práci výzkumu včelích tanců v roce 1973 získal Nobelovu cenu [29].

5.1.1 Včelí tanec

Včelí mozek se skládá z pouhého milionu nervových buněk. To je miliontina množství neuronů ukrytých v lidském mozku [29]. A i přes tuto skutečnost se dokáží efektivně dorozumívat pomocí tzv. „tanců“.

Prostřednictvím těchto tanců včely jednoznačně říkají, kde se nalezená potrava nachází. Nejprve se předpokládalo, že včely používají pouze 2 druhů tanců. Kruhový a kývavý. Kruhové tance informují o výskytu bohatého zdroje potravy v blízkosti úlu, kývavé (ve tvaru osmičky) naopak informují o potravě na vzdálenějších pastvách. V prostřední části této osmičky kýve včela celým tělem ze strany na stranu a zároveň třepotá křídly. Třepotavý úsek tance je jeho nejdůležitější částí, v níž je obsažena veškerá informace. Právě tím směrem, kterým se létavka třepotá, leží kýžená potrava. Rychlost, s jakou se včela během nejdůležitější části tance třepotá, určuje vzdálenost potravy. Rychlejší kývání tělem znamená bližší potravu, pomalejší třepot potravu vzdálenou [27].

Na základě vícero těchto tanců si vyčkávající včely vyberou pro ně nejvýhodnější zdroj potravy. Větší pravděpodobnost výběru mají ziskovější zdroje.

Později byly odhaleny navíc další dva typy tanců. První tzv. mobilizační (natrásavý tanec) pro vyslání více dělnic za potravou a druhý naopak pro povolání dělnic zpět do úlu [29].

5.1.2 Chování včel

V této kapitole bude stručně uvedeno, jak včely komunikují a jak si vybírají, na který zdroj potravy se zaměří. Znázornění chování včel je možno vidět na Obr. 7 (obrázek částečně převzat z [31]).

Ve zdroji [30], [31] se uvádí, že každý jednotlivý hmyz má svoji specializaci a díky tomu se při shromažďování informací a procesu učení nepodílí na všech úkolech. Obecně platí, že každá kolonie sociálního hmyzu se chová podle vlastního dělení. Včelí systém se skládá ze tří základních komponent:

1. Zdroje potravy – Hodnota zdroje závisí na odlišných parametrech, jako je vzdálenost od úlu, bohatství energie, kvalita nektaru a extrahování této energie.

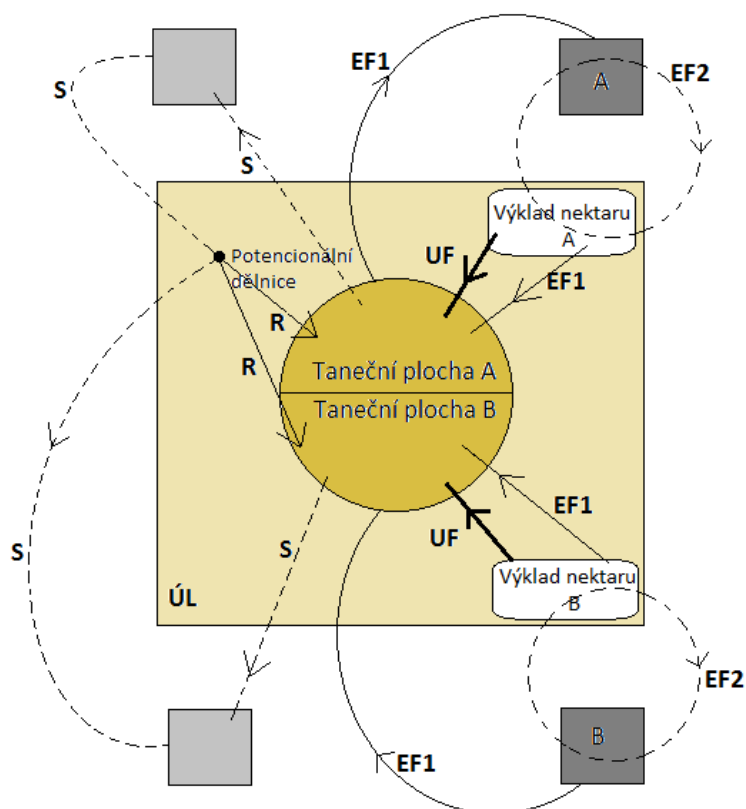
2. Nezaměstnaná dělnice – Včely nemají žádné znalosti o zdroji potravy ve svém okruhu prohledávání. Včely tedy začínají vyhledávat jako nezaměstnané dělnice. Existují dva typy:

- Skaut: včela začne hledat spontánně, bez nějaké znalosti (S).
- Rekrut (vyčkávací včela): účastní se včelího tance prováděného jinou včelou a začne vyhledávat pomocí nově nabytých znalostí (R).

Po nalezení zdroje potravy si včela zapamatuje umístění (lokaci) a potom z něj okamžitě začne těžit. Včela se tak stane zaměstnanou dělnicí (z rekruta nebo skauta). Posbírá náklad nektaru ze zdroje, vrátí se do úlu a vyloží nektar do potravinového skladu. Po vyložení nektaru má včela následující možnosti:

3. Zaměstnaná dělnice

- Jestliže se množství nektaru snížilo na nízkou úroveň, nebo je již vyčerpané, včela opouští zdroj potravy a stává se nezaměstnanou včelou (uncommitted follower, UF).
- Pokud je ještě dostatečné množství nektaru ve zdroji, může z ní nadále těžit bez sdílení informace o zdroji s hnízdem (EF2).
- Nebo může prostřednictvím včelího tance informovat hnízdo o stejném zdroji jídla. Pravděpodobnost této možnosti souvisí s kvalitou zdroje potravy (EF1).



Obrázek 7 - Chování dělnic při shánění potravy

5.2 Umělé včely

Tak jako jiné zmíněné algoritmy se i umělé včely inspiroují svými skutečnými protějšky. V dalších kapitolách jsou shrnuty některé z algoritmů vzešlých od reálných včel a převážně u algoritmů ABC a BA jsou popsány jednotlivé fáze životního cyklu.

5.2.1 Studie o umělých včelích systémech

Schopnost a chování skutečných biologických včel inspirovalo mnoho vědců k vyvinutí algoritmu právě na jejich principu. V Tabulce 2 jsou zmíněny některé z nich. (Pozn. Pro některé názvy algoritmů vyskytujících se v této tabulce neexistuje český ekvivalent, a proto jsou názvy ponechány v originálním znění. Názvy algoritmů tedy jsou: český název (existuje-li), anglický název, zkratka.)

Včelí vlastnosti, jako je chování při hledání potravy, učení, schopnost pamatování si a sdílení informací, patří v poslední době k jedné z nejzajímavějších oblastí studií výzkumu inteligence roje (hejna). Po podrobném prozkoumání těchto studií se algoritmy inspirovaly následujícím chováním [30].

- Chování při shánění potravy
- Chování při páření
- Pojetí včelí královny

Všechny algoritmy uvedené v Tabulce 2 byly navrženy pro vylepšení schopností genetických algoritmů. Jak se uvádí v [30], jsou v dalších odstavcích uvedeny pouze některé z algoritmů inspirovaných se chováním včel. Jako první budou zmíněny:

Sato a Hagiwara navrhli lepší genetický algoritmus založený na chování včel (Bee System). Jak je známo, genetické algoritmy mají dobrou schopnost globálního vyhledávání, nicméně postrádají schopnost lokálního vyhledávání. Na druhou stranu je nízká pravděpodobnost, že Bee System spadne do lokálního minima/maxima díky kombinaci lokálního a globálního vyhledávání. Cílem tohoto algoritmu je zlepšit lokální vyhledávací schopnosti genetického algoritmu, aniž by došlo ke snížení kvality globálního vyhledání. V experimentálních studiích bylo zjištěno, že Bee System je ve srovnání s konvenčním genetickým algoritmem ukazuje lepší výkon, zejména u velmi složitých vícerozměrných funkcí.

Karaboga analyzuje potravní chování včel a navrhuje nový algoritmus simulující toto chování pro řešení multi-dimenzionálních a multi-modálních optimalizačních problémů, tzv. Artificial Bee Colony (ABC).

Yang prezentuje virtuální včelí algoritmus (VBA), který je účinný na funkční optimalizační problémy. Je podobný s genetickým algoritmem, ale je více účinnější díky paralelismu více nezávislých včel. Pro lepší představu byl VGA testován na dvou funkcích o dvou parametrech. Jeden multimodální a druhý unimodální. Ve všech směrech byl algoritmus lepší než genetický.

Wedde a kolektiv představili směrovací protokol (nazván včelí úl, BeeHive), který je odolný proti chybám, adaptivní a robustní. Protokol je inspirován včelím tancem a schopností sehnat si potravu a je využíván pro telekomunikační sítě. Algoritmus byl testován a výsledky ukazují podobný nebo lepší výkon ve srovnání s ostatními algoritmy.

Bozorg Haddad a kolektiv navrhli Honey-Bees Mating Optimization (HBMO) algoritmus, založený na práci profesora H. A. Abbasse, který jako první představil algoritmus inspirovaný pářícím procesem u včel. Algoritmus HBMO je vhodný k řešení nelineárních matematických modelů. HBMO byl testován na několika omezených matematických

funkcích a byl porovnán s genetickými algoritmy. Výsledky ukázaly mírné zlepšení u algoritmu HBMO.

Typ	Autor	Název Algoritmu	Aplikace
Shánění potravy	Sato a Hagiwara (1997)	Včelí systém (Bee System, BS)	Vylepšení genetických algoritmů
	Karaboga (2005)	Umělá včelí kolonie (Artificial Bee Algorithm, ABC)	Průběžné optimalizace
	Yang (2005)	Virtuální včelí algoritmus (Virtual Bee Algorithm, VBA)	Průběžné optimalizace
	Pham a kolektiv (2006)	Včelí algoritmus (Bee Algorithm, BA)	LVQ-Neuronové sítě
	Lucic a Teodorovic (2001)	Včelí systém	Problém obchodního cestujícího (TSP)
	Teodorovic a Dell'Orco, Lucic (2001)	Optimalizace včelím rojem (Bee Colony Optimization, BCO)	Kombinatorické problémy
	Wedde a kolektiv (2004)	Úl (BeeHive)	Směrování telekomunikačních sítí
Páření	Benatchba a kolektiv (2005)	Marriage in Honey-Bees Optimization Based (MBO)	Data Mining
	Chang (2006)	MBO	Stochastické dynamické programování
	Bozorg Haddad a kolektiv (2006)	Honey-Bees Mating Optimization -HBMO	Nelineární optimalizace
Královna	Sung (2003)	Queen-Bee Evolution Algorithm(QBE)	Vylepšení genetických algoritmů
	Kara (2004)	Včelí křížení (Bee Crossover)	Vylepšení genetických algoritmů
	Azeem a Saad (2004)	Modifikovaný QBE	Vylepšení genetických algoritmů

Tabulka 2 - Algoritmy založené na principu chování včel

Sung představil algoritmus evoluce včelí královny (Queen-Bee Evolution) opět s cílem vylepšit genetický algoritmus. Rozdíl oproti genetickým algoritmům je v křížení. U algoritmu Evoluce včelí královny je to křížení královny s ostatními včelami (vybraných jako rodiče podle jiného výběrového algoritmu), kdežto u genetických algoritmů se používá ruletový výběr. Pro lepší výsledky se používá mutace pouze některých jedinců místo všech.

5.3 Umělá včelí kolonie (Artificial Bee Colony)

Umělá včelí kolonie je v originálním znění známá jako Artificial Bee Colony (dále již ABC), patří mezi metaheuristické algoritmy a byla vyvinuta Dervisem Karabogou v roce 1995. Jak

již název sám napovídá, algoritmus se inspiroje chováním skutečných včel v přírodě při shánění potravy.

Pár faktů ohledně ABC, viz [32].

- Kolonie umělých včel se skládá ze tří skupin. Tím jsou dělnice, vyčkávací včely a skauti.
- Pravidlem je, že první půlka kolonie patří dělnicím a druhá půlka jsou vyčkávací včely.
- Počet dělnic by měl odpovídat počtu zdrojů kolem úlu.
- Ty dělnice, které vyčerpají své zdroje, se stávají skauty.

Mezi nastavitelné parametry patří:

- Velikost roje (populace).
- Poměr dělnic a vyčkávacích včel v roji.
- Limit (dělnice opustí své řešení, tzv. míra zužitkování)
- Skauti

5.3.1 Životní cyklus algoritmu

Obecné kroky algoritmu jsou podle [31], [32]:

- Inicializační fáze
- OPAKUJ
 - a. Fáze dělnic
 - b. Fáze vyčkávacích včel
 - c. Fáze skautů
 - d. Aktualizace dosud nejlepšího řešení do paměti
 - e. Cyklus++
- DOKUD (cyklus = maximum nebo vypršení času)

Každá z fází bude detailně popsána v nadcházejících kapitolách, uvedeno v [31] a [33].

5.3.2 Inicializační fáze

Co se týče populace, ve většině případů se poměr dělnic a vyčkávacích včel dělí rovným dílem (uživatelsky nastavitelný parametr). Jako první je potřeba náhodně vygenerovat vektor počátečních jedinců. V závislosti na tom, kolik je uživatelem nastavených dělnic, je i vygenerován takový počet řešení.

Počáteční vektor \vec{x}_m ($m = 1 \dots SN$, SN představuje velikost populace dělnic), je vektor řešení optimalizačního problému. Každý vektor \vec{x}_m má další proměnnou D (\vec{x}_{mi} , $i = 1 \dots n$), která značí optimalizační parametry (dimenzi). Pro tuto počáteční inicializaci je použit následující vztah (1).

$$x_{mi} = l_i + rand(0,1) * (u_i - l_i) \quad (1)$$

Kde l_i je nejnižší a u_i je nejvyšší hranice definičního oboru.

5.3.3 Fáze dělnic

Každá dělnice hledá ve svém okolí další zdroje potravy (\vec{v}_m), dle vztahu (2). Snaží se nalézt lepší zdroj potravy (lepší CV). Po nalezení sousedního řešení počítá novou hodnotu účelové funkce.

$$v_{mi} = x_{mi} + \phi_{mi} (x_{mi} - x_{ki}) \quad (2)$$

Kde $\phi_{mi} = rand(-1,1)$,

$k = rand(1,n) \neq m$ (k se nesmí rovnat stejnému vektoru, vztah v závorce by se rovnal nule a díky tomu by se náhodnost neprojevila).

Jestliže hodnota účelové funkce \vec{v}_m je lepší než jeho rodič \vec{x}_m , pak $\vec{x}_m = \vec{v}_m$. Pokud je tomu naopak, je ponecháno \vec{x}_m nezměněné. Toto hledání sousedního řešení se dá brát jako mutace algoritmu, která je důležitou součástí.

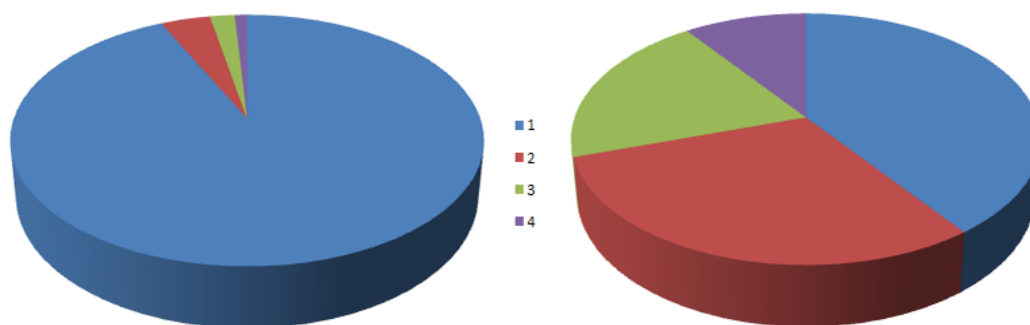
Po nalezení všech řešení nastává fáze vyčkávacích včel.

5.3.4 Fáze vyčkávacích včel

Aby se vědělo, kolik včel vyslat k jednotlivým řešením, je nutné k těmto řešením přiřadit pravděpodobnosti v závislosti na jejich kvalitě. Čím kvalitnější řešení, tím více včel.

Pro určení pravděpodobností je vhodné nejdřív tyto řešení přerozdělit (nepřerozdělené a přerozdělené pravděpodobnosti lze znázornit na Obr. 8) a převést na kladná čísla, je-li tomu jinak. K tomuto účelu si lze vybrat z více řešení. Například využití formule (3), turnajového výběru nebo jiných možností.

$$fit_m(\vec{x}_m) = \begin{cases} \frac{1}{1 + f_m(\vec{x}_m)} & \text{if } f_m(\vec{x}_m) \geq 0 \\ \frac{1}{1 + Abs[f_m(\vec{x}_m)]} & \text{if } f_m(\vec{x}_m) < 0 \end{cases} \quad (3)$$



Obrázek 8 – Ukázka přerozdělení pravděpodobností

U přerozdělených hodnot pracujeme pouze s kladnými čísly. Lepší hodnota má větší díl koláče na grafu. Pokud by nedošlo k přerozdělení, mohlo by dojít k degeneraci algoritmu a algoritmus by mohl uváznout v lokálním extrému.

Dle vztahu (4) vypočítáme pravděpodobnost každého řešení.

$$p_m = \frac{fit_m(\vec{x}_m)}{\sum_{m=1}^{SN} fit_m(\vec{x}_m)} \quad (4)$$

Poté si vyčkávající včely na základě vypočítané pravděpodobnosti zvolí daný zdroj. Lepší zdroje nalákají více včel. Tyto nově získané včely opět podle vztahu (2) počítají okolí v bodě zdroje. Původní zdroj, na kterém se včela původně nacházela, je nahrazen tím nejlepším okolím.

5.3.5 Fáze skautů

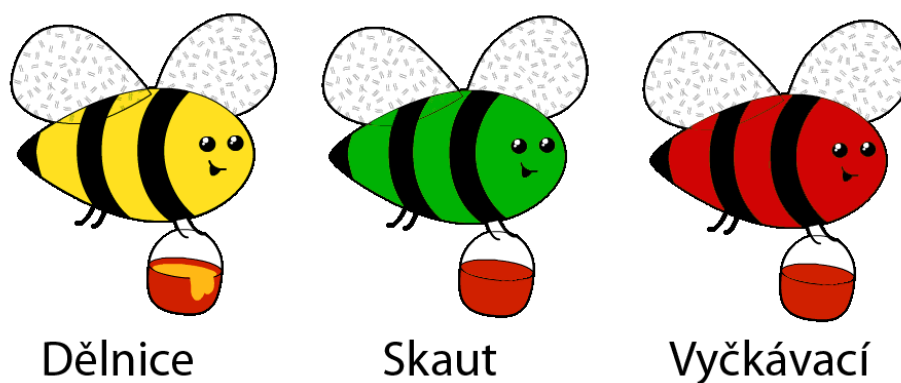
Skaut je nezaměstnaná včela, která hledá zdroj potravy. Dělnice se tedy stává nezaměstnanou včelou (skautem), pokud je hodnota účelové funkce po několika iteracích stále stejná a nelepší se. Počet těchto iterací je uživatelsky nastavitelný parametr (limit). Skaut opět náhodně vyhledává zdroj potravy podle vztahu (1).

To nejlepší řešení se ukládá do globální paměti. Tento cyklus od skautů až po vyčkávající včely se opakuje tak dlouho, dokud nebude dosaženo maximální hodnoty počtu cyklů nebo do vypršení nastaveného času.

5.3.6 Příklad ABC

Uživatel si nastavil 3 dělnice a 10 vyčkávacích včel. Následující obrázky byly částečně převzaty z [41].

Pro následující demonstraci příkladu je možno si představit 3 hlavní včely (Obrázek 9), které vystupují v algoritmu ABC.



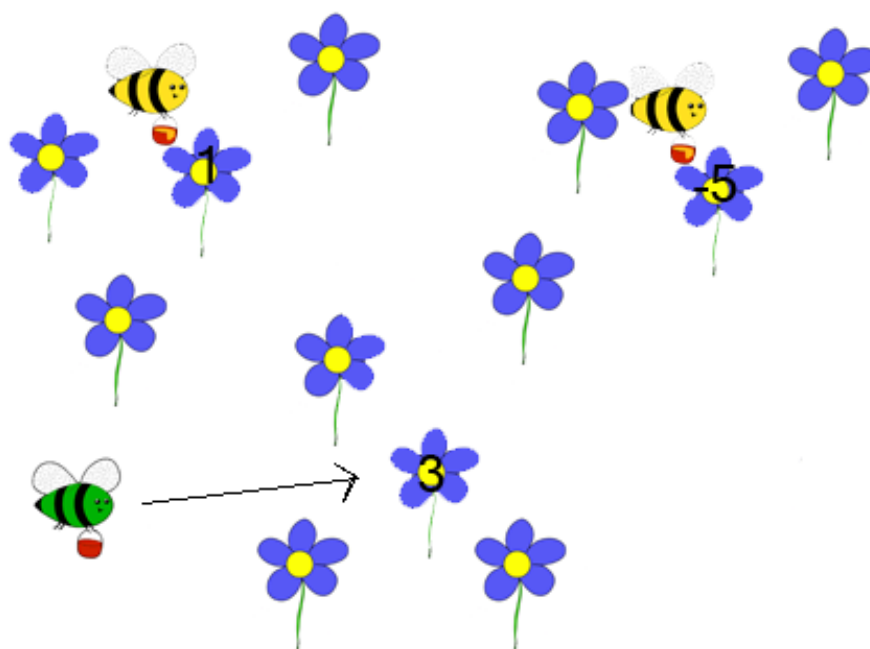
Obrázek částečně převzat z <http://www.acclaimclipart.com>

Obrázek 9 – Základní prvky ABC

1. Provedeme inicializaci počáteční populace podle vztahu (1), Obrázek 10.

Výsledné hodnoty účelové funkce po počátečním vygenerování jsou:

$$f_m(\vec{x}_m) = (1, -5, 3).$$

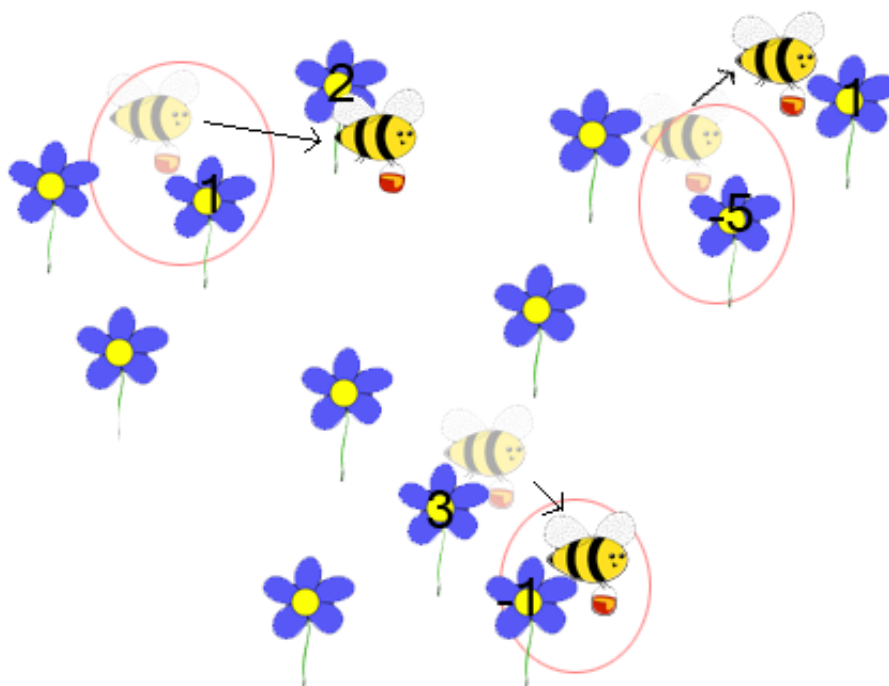


Obrázek 10 - Inicializace počáteční populace ABC

2. Podle vztahu (2) vypočítá dělnice okolí a opět spočítá hodnotu účelové funkce.

$$f_m(\vec{v}_m) = (2, 1, -1)$$

Vyberou se ty nejlepší hodnoty z \vec{v}_m a \vec{x}_m , Obrázek 11.



Obrázek 11 - Výpočet okolí a výběr nejlepších ABC

Nové hodnoty tedy jsou: $f_m(\vec{x}_m) = (1, -5, -1)$.

3. Vypočtou se pravděpodobnosti výběru jednotlivých řešení (pro vyčkávací včely).

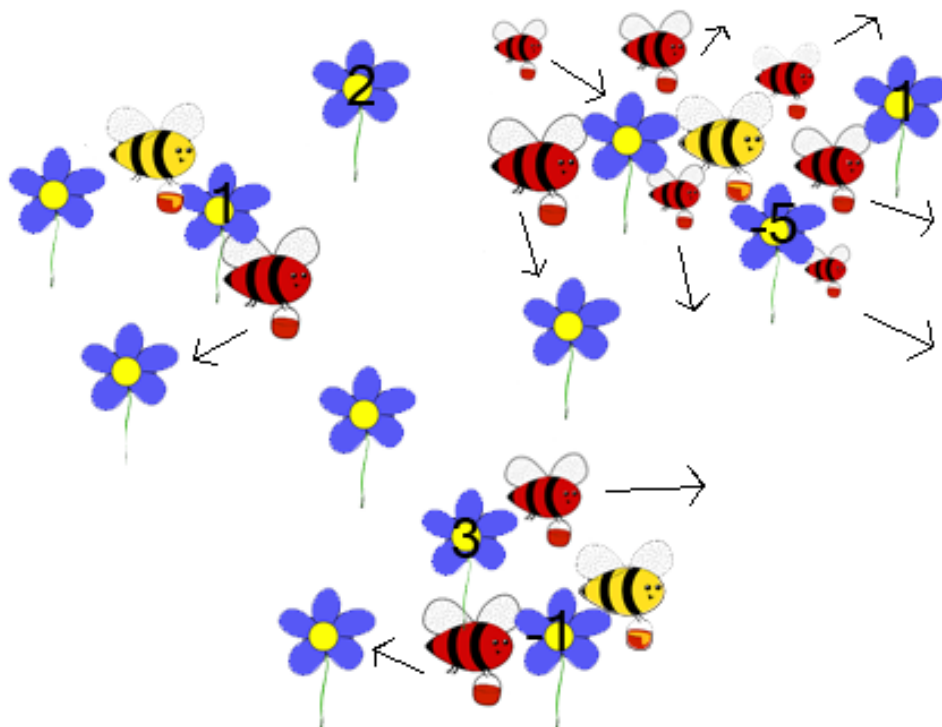
Nejdříve jsou ale hodnoty upraveny podle (3) a vypočítány pravděpodobnosti podle vztahu (4).

$$fit_m(\vec{x}_m) = \left(\frac{1}{2}, 6, 2\right).$$

$$\vec{p}_m = (5.9\%, 70.6\%, 23.5\%).$$

4. Rozdělíme 10 vyčkávacích včel podle vypočítané pravděpodobnosti, Obrázek 12.

5.9 % \rightarrow 1 včela; 70.6 % \rightarrow 7 včel; 23.5 % \rightarrow 2 včely.



Obrázek 12 - Rozdělení včel podle pravděpodobnosti ABC

Každá z vyčkávacích včel (červená) si spočítá hodnotu účelové funkce ve vyhledaném okolí (vztah (2)) a porovná se svou dělnicí. Dělnice se přesune na tu nejlepší hodnotu. Nenaudou-li vyčkávací včely lepší hodnotu, dělnice zůstává tam, kde je. Pokud nedojde ke zlepšení během několika kol (opět nastavitelný parametr uživatelem), z dělnice se stává skaut a podle vztahu (1) se přesunuje na nové místo.

5. Ta dosud nejlepší hodnota účelové funkce se uloží do globální paměti roje. Nyní začíná další cyklus (začátek od bodu 3).

5.4 Včelí algoritmus (Bee Algorithm)

Včelí algoritmus (Bees Algorithm, dále jen BA), který byl vyvinut D. T. Phamem a kolektivem je populační vyhledávací algoritmus, který napodobuje chování roje včel při hledání zdroje potravy. Ve své základní verzi algoritmus provádí určitý druh sousedského vyhledávání v kombinaci s náhodným vyhledáváním. Mohou být použity jak pro kombinatorické, tak i funkční optimalizace. BA byl aplikován na několika optimalizačních problémech, například: školení neuronové sítě pro rozpoznávání, plánování práce výrobních strojů, hledání vícero proveditelných řešení určitého problému, seskupování dat, optimalizace designu mechanických dílů, multi-objektivní optimalizace, ladění regulátoru pro roboty aj [34].

Jak již bylo dříve zmíněno, v reálném životě se včelí skauti snaží nalézt slibné květy s co největším podílem nektaru. Když se vrátí do úlu, vyloží nektar a tzv. včelím tancem začne komunikovat s kolonií. Poté se vrací i se svými následovníky zpět ke květu. Více následovníků je posláno na lepší zdroj. To umožňuje kolonii získat potravu rychle a efektivně. Stejně tak začíná včelí algoritmus, kde se včelí skauti (průzkumnice) náhodně rozmístí do prostoru.

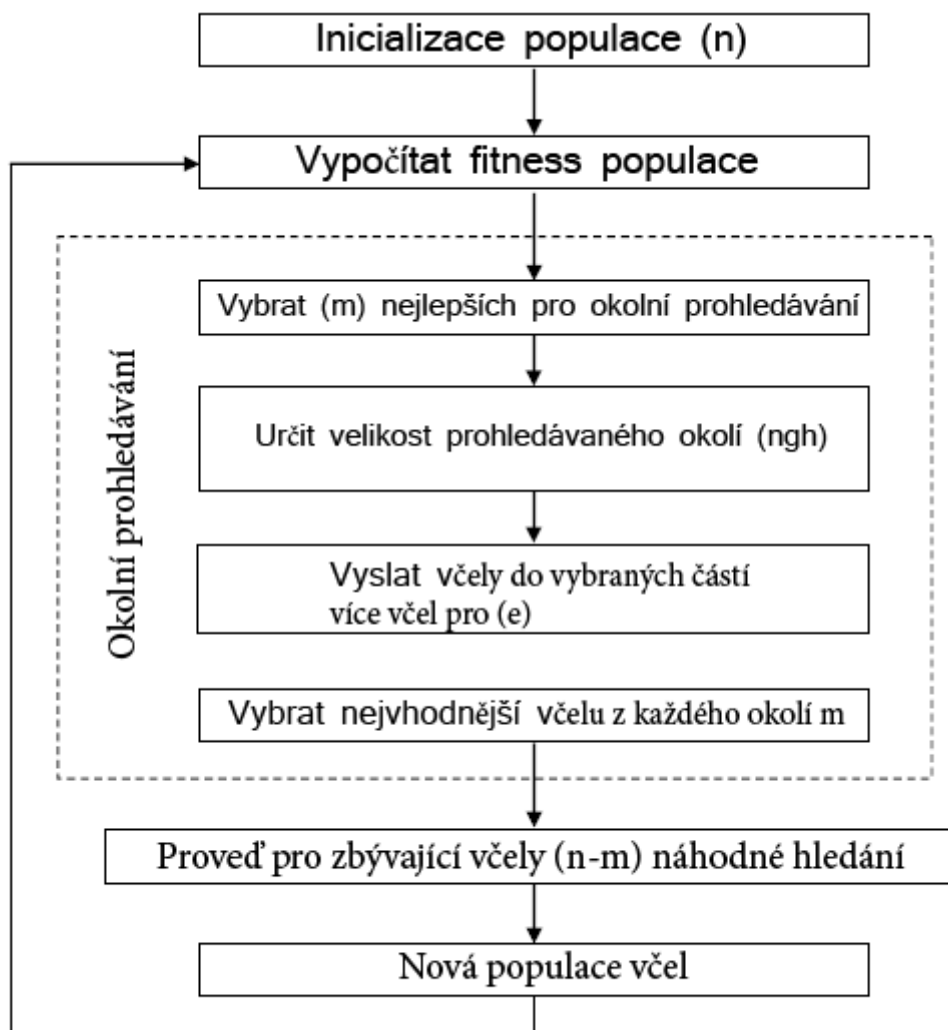
5.4.1 Životní cyklus algoritmu

Mezi základní uživatelsky nastavitelné parametry, které jsou součástí životního cyklu, patří:

- n – Představuje počet jedinců v populaci.
- m – Počet jedinců, kteří budou vybráni pro bližší zkoumání v okolí. Tito jedinci jsou dosud nejlepší z populace.
- e – Další výběr nejlepších, ale tentokrát ne z celé populace, ale již z užšího výběru m .
- n_2 – Počet jedinců, kteří budou zkoumat okolí e , čím víc, tím líp, ale na úkor rychlosti výpočtu algoritmu.
- n_1 – Počet jedinců, kteří budou zkoumat okolí $(m-e)$.
- n_{gh} – Nastavení velikosti okolí, pro výběr včel m , ve kterém se bude nově prohledávat. Určeno pro parametry n_2 a n_1 . Nastaví-li se okolí 0.5, okolí bude zvětšeno o 0.5 na každou stranu.

- *Iterace* – Čím víc, tím líp, ale opět na úkor rychlosti algoritmu.

Hlavní kroky algoritmu jsou znázorněny v následujícím diagramu na Obr. 13:



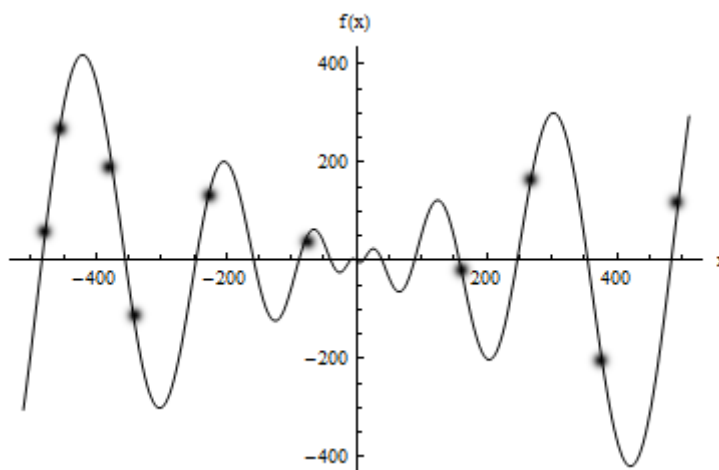
Obrázek 13 - Životní cyklus algoritmu

Nevýhoda včelího algoritmu je příliš mnoho laditelných parametrů.

5.4.2 Příklad BA

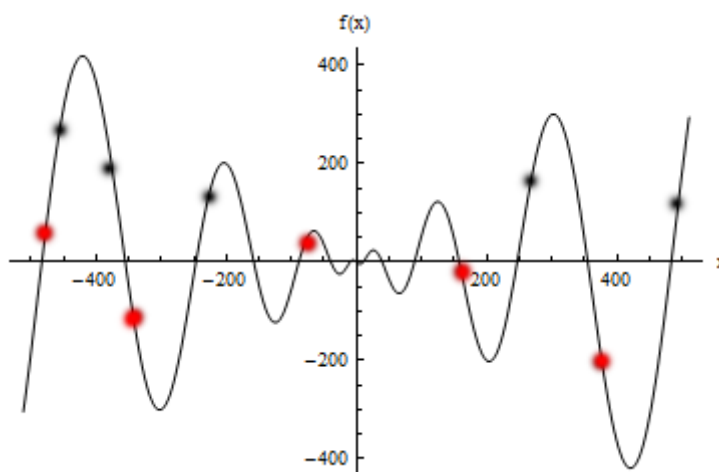
Jako testovací funkce bude zvolena funkce Schwefel, $\langle -512, 511 \rangle$.

1. Náhodně vygeneruji počáteční pozici ($n = 10$), Obrázek 14.

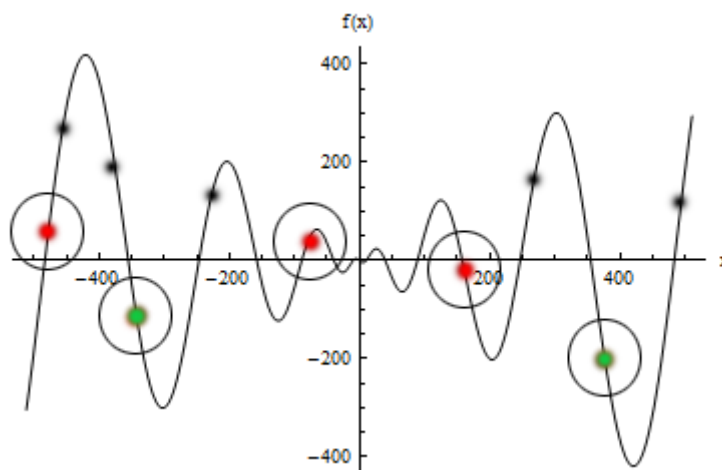


Obrázek 14 - Inicializace populace BA

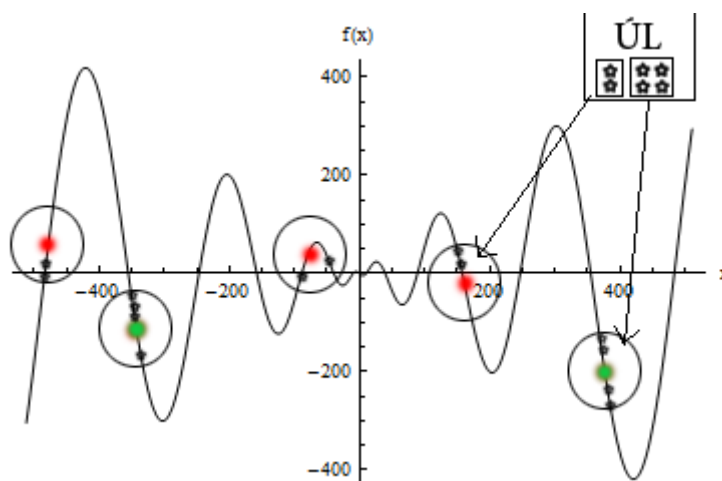
2. Pro každého jedince vypočítám jeho CV a určím m nejlepších. V tomto případě například $m = 5$. Obrázek 15 ukazuje, vybraných m nejlepších včel.

Obrázek 15 - Výběr (m) nejlepších včel BA

3. m nejlepších (červené kolečko) se rozdělí na elitu e (zelené kolečko), kteří představují dosud nejlepší řešení a ty ostatní ($m-e$). Zvolím okolí (ngh), ve kterém se bude nově prohledávat. $e = 2$, $m-e = 3$, $ngh = 1$. Na Obrázku 16 jsou barevně rozděleny jednotlivé skupiny i s vyznačeným okolím.

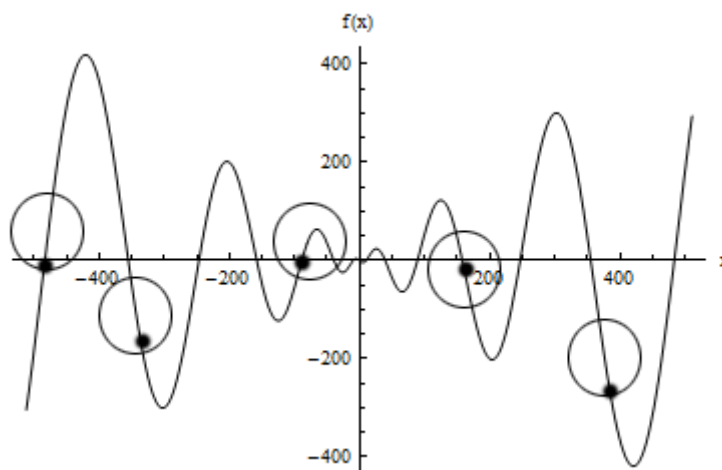
Obrázek 16 - Zobrazení okolí pro (m) nejlepších BA

4. Ke každé vybrané včele pošlu navíc další včely. U lepšího umístění se vyskytne více včel. Pro elitu e pošlu 4 včely ($n2$) a pro ostatní $m-e$ pošlu 2 včely ($n1$). Ostatní včely $n-m$ se zahodí. Znázornění na Obrázku 17.



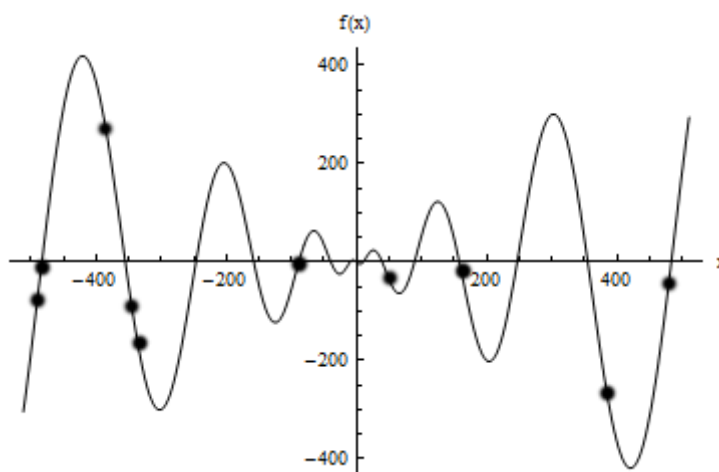
Obrázek 17 – Vyslání více včel do prohledávaného okolí BA

5. Pro každé z okolí se vybere nejlepší včela (nejlepší ohodnocení účelové funkce). Na Obrázek 18 je možno vidět výběr nejlepších jedinců.



Obrázek 18 - Výběr nové nejlepší včely z každého okolí

6. Vygeneruji nové jedince, aby celkový počet jedinců v populaci byl n . V tomto případě $n-m = 5$. Obrázek 19 – ukázka rozložení jedinců po první generaci.



Obrázek 19 - Generování nových jedinců

7. Postup se opakuje od bodu 2.

5.5 Optimalizace včelích kolonií (Bee Colony Optimization)

V roce 2005 Teodorovic a Dell'Orco navrhli Bee Colony Optimization (dále jen BCO). Základní principy kolektivní včelí inteligence byly prvně použity v algoritmu Bee System (kapitola 5.6 Včelí systém) a právě tento algoritmus byl jeho zobecněním, protože je schopen řešit kombinační optimalizační úlohy. Jedná se o populační algoritmus, kde populace umělých včel hledá optimální řešení. Tyto včely mezi sebou spolupracují a snaží se vyřešit daný problém. Každá umělá včela představuje jedno řešení problému [30].

Algoritmus obsahuje dvě fáze. Dopřednou a zpětnou.

5.5.1 Životní cyklus algoritmu

Na počátku hledání všechny včely začínají v úlu. Životní cyklus podle [35] je následující.

1. Inicializace – Každá včela začíná s nulovým řešením.
2. Dopředná fáze pro všechny včely.
 - a. Nastav $k = 1$ (počítadlo pro skoky v jednotlivých stupních)
 - b. Vypočti všechny možné skoky
 - c. Přes ruletový výběr zvol jeden skok
 - d. $k++$; Pokud $k \leq$ počet stupňů, pak opakuj od kroku 2.
3. Vrať všechny včely zpět do úlu (začátek zpětné fáze).
4. Vyber včely a jejich hodnoty účelových funkcí.
5. Každá včela se náhodně rozhodne, zda bude pokračovat vlastním prozkoumáváním, zda se stane rekrutem (láká ostatní včely na své řešení), nebo se stane následovníkem. (Včely s větší hodnotou účelové funkce mají větší šanci pokračovat ve svém hledání.)
6. Pro každého následovníka vyber ruletovým výběrem od rekrutů nové řešení.
7. Jestliže nejsou splněny podmínky, jdi na krok 2.
8. Nejlepší řešení.

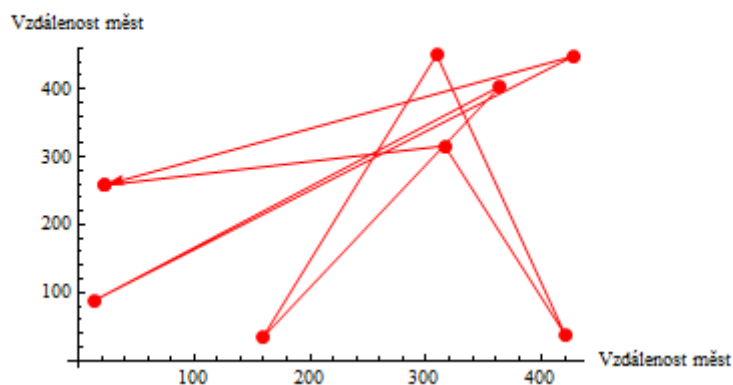
Ukončovací podmínky mohou být například dovršení maxima všech iterací, dovršení maxima iterací bez zlepšení hodnoty účelové funkce, dosažení daného času atd.

5.6 Včelí systém (Bee System)

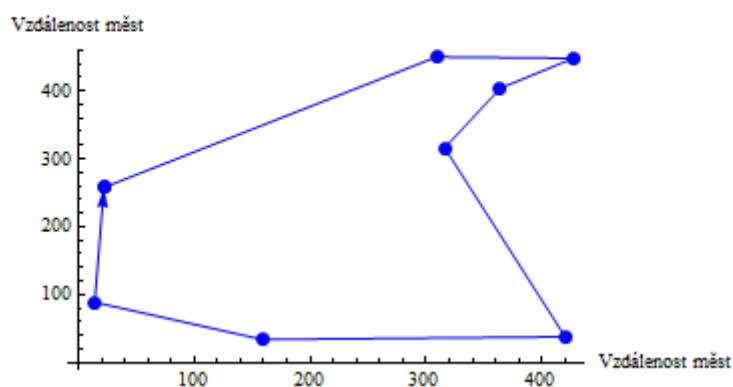
Algoritmus pochází od autorů Lucic a Teodorovic (2001), kteří objevili možné aplikace kolektivní včelí inteligence v řešení komplexních dopravních a přepravních inženýrských problémů. Ve své první studii navrhli algoritmus včelího systému (Bee System), který byl použit k tradičnímu problému obchodního cestujícího (TSP) [34].

TSP se řadí do problémů NP-Hard a který si klade za cíl najít minimum při průchodu každým uzlem alespoň jednou. Např. na Obr. 20 je možno vidět maximální a minimální vzdálenost mezi potenciálními městy [30].

Maximální vzdálenosti mezi městy



Minimální vzdálenosti mezi městy



Obrázek 20 - Problém obchodního cestujícího

II. PRAKTICKÁ ČÁST

6 MATHEMATICA

Software Mathematica je výpočetní software, který je využíván jak vědci, tak i studenty ke svým pracím, který dokáže řešit složité matematické úkony, vykreslovat funkce, grafy a různé interaktivní prvky.

Pomocí softwaru lze řešit projekty libovolného rozsahu od jednoduchých výpočtů až po velkosystémová řešení [36].

Nachází široké uplatnění napříč různých oblastí od vědecko-technických institucí až po lékařství a astronomii [36]. Hlavní využití softwaru spočívá v modelování a simulacích.

Již po 20 let se Mathematica používá od jednoduchých výpočtů mnohých výzkumníků až po ohromné projekty v celosvětových skupinách čítajících až desítky lidí. Mathematicu lze využívat i pro rychlou a přesnou analýzu dat a testování hypotéz [38].

Výhoda softwaru je v technické podpoře, kde je tým odborníků připraven denně odpovídat na potřebné otázky, a v bravurně zvládnuté nápovědě. Nápověda činní kolem 10 000 stránek s matematickými funkcemi a tutoriály a více než 100 000 příkladů [37].

7 TESTOVACÍ FUNKCE

Oba srovnávané algoritmy byly testovány na testovacích funkcích, které slouží k otestování robustnosti evolučních algoritmů. V komunitě lidí zabývajících se evolučními výpočetními technikami jsou tyto testovací funkce uznávány.

U každé testovací funkce bude uveden její analytický zápis i s příslušnými hranicemi. Dále jeho globální minimum v jedno, dvou a n dimenzionálním prostoru. Vykreslení všech testovacích funkcí bude provedeno a) v 2d prostoru, b) ve 3d prostoru a c) zobrazení vrstevnicového grafu v 2d.

Grafy a) a b) byly vytvořeny v programu Mathematica, graf c) byl převzat ze stránky [39].

7.1 Charakteristika testu

Aby byly testy srovnatelné, je potřeba porovnávat jednotlivé algoritmy v počtu ohodnocení účelové funkce (Cost Function Evaluation, dále jen CFE). Takto budou vykreslené testy do grafů mnohem více srozumitelné a na první pohled porovnatelné.

Včelí algoritmus byl porovnáván s algoritmem SOMA.

Výpočet CFE včelího algoritmu: $(n + e * n^2 + (m - e) * n + (n - m)) * \text{iterace}$.

Výpočet CFE algoritmu SOMA:
$$\frac{(\text{PopSize} - 1) * \text{PathLength} * \text{Migrations}}{\text{Step}}$$

K testování včelího algoritmu bylo použito 6 vybraných testovacích funkcí, viz následující kapitola. Postup testu je následující: Každá testovací funkce byla otestována na 2, 5, 10, 20, 30 a 50 dimenzí. U včelího algoritmu je mnoho nastavitelných parametrů a proto je obtížné tyto parametry správně nastavit. Po pár testech a jak bylo uznáno za vhodné, byly parametry nastaveny. Pro každou dimenzi byl výpočet opakován 30 krát.

U algoritmu SOMA, u většiny případů, byl kladen důraz na to, aby počet ohodnocení účelové funkce (CFE) byl víceméně stejný (pro lepší srovnání grafů).

7.2 Vybrané testovací funkce

Rovnice jednotlivých funkcí byly zkontrolovány podle [40].

7.2.1 1st De Jong

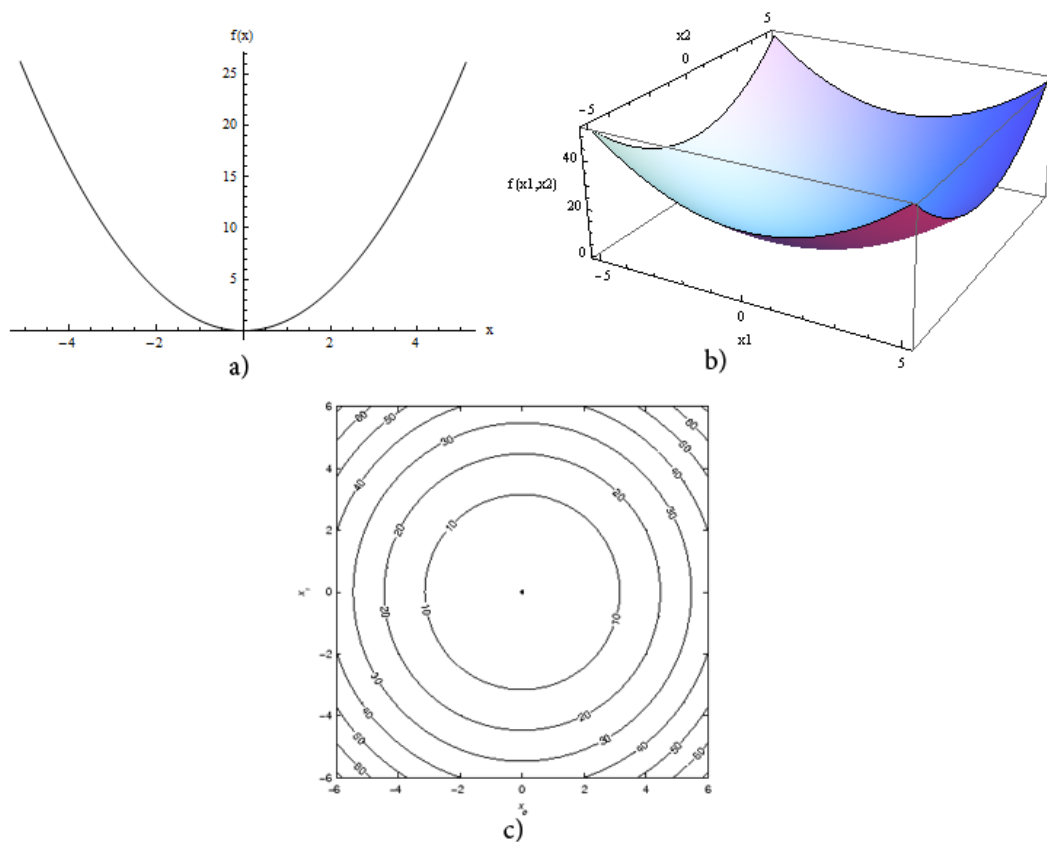
Funkce: $f(\vec{x}) = \sum_{i=1}^D x_i^2$, $-5.12 \leq x_i \leq 5.12$

Globální minimum: D1 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 0\}$.

D2 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 0, x_2 = 0\}$.

Dn $\rightarrow f(\vec{x}) = n \cdot f(x_1) = 0$ na pozici $\{x_1, \dots, x_n = 0\}$.

Příklad(y) vykreslení funkce 1st De Jong je (jsou) zobrazen(y) na Obr. 21.



Obrázek 21 – Zobrazení funkce 1st De Jong

7.2.2 2nd De Jong

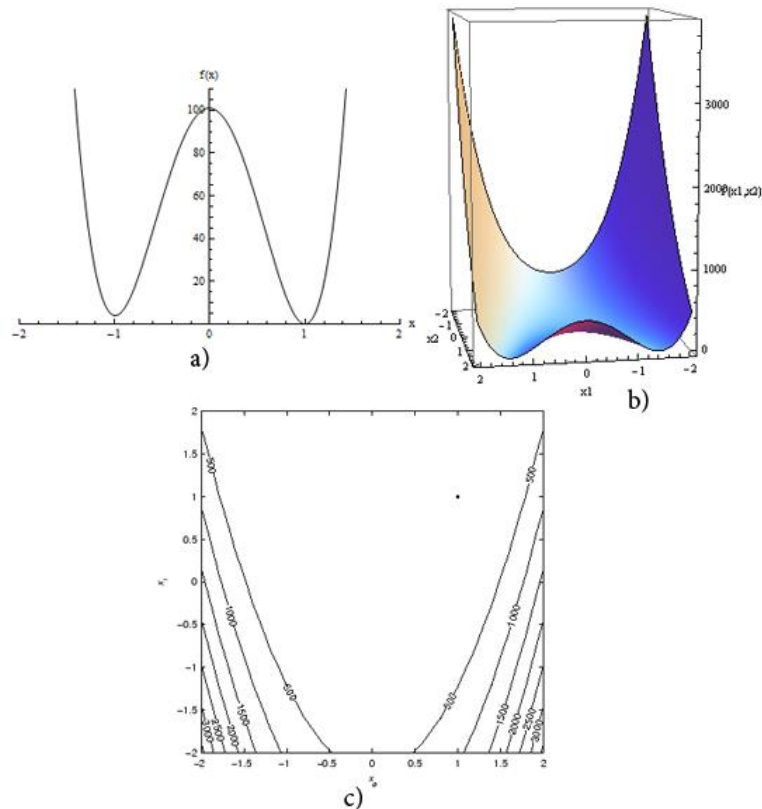
Funkce: $f(\vec{x}) = \sum_{i=1}^{D-1} (100 \cdot (x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$, $-2.048 \leq x_i \leq 2.047$

Globální minimum: D1 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 1\}$.

D2 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 1, x_2 = 1\}$.

$$Dn \rightarrow f(\vec{x}) = n \cdot f(x_1) = 0 \text{ na pozici } \{x_1, \dots, x_n = 1\}.$$

Příklad(y) vykreslení funkce 2nd De Jong je (jsou) zobrazen(y) na Obr. 22.



Obrázek 22 – Zobrazení funkce 2nd De Jong

7.2.3 Schwefel

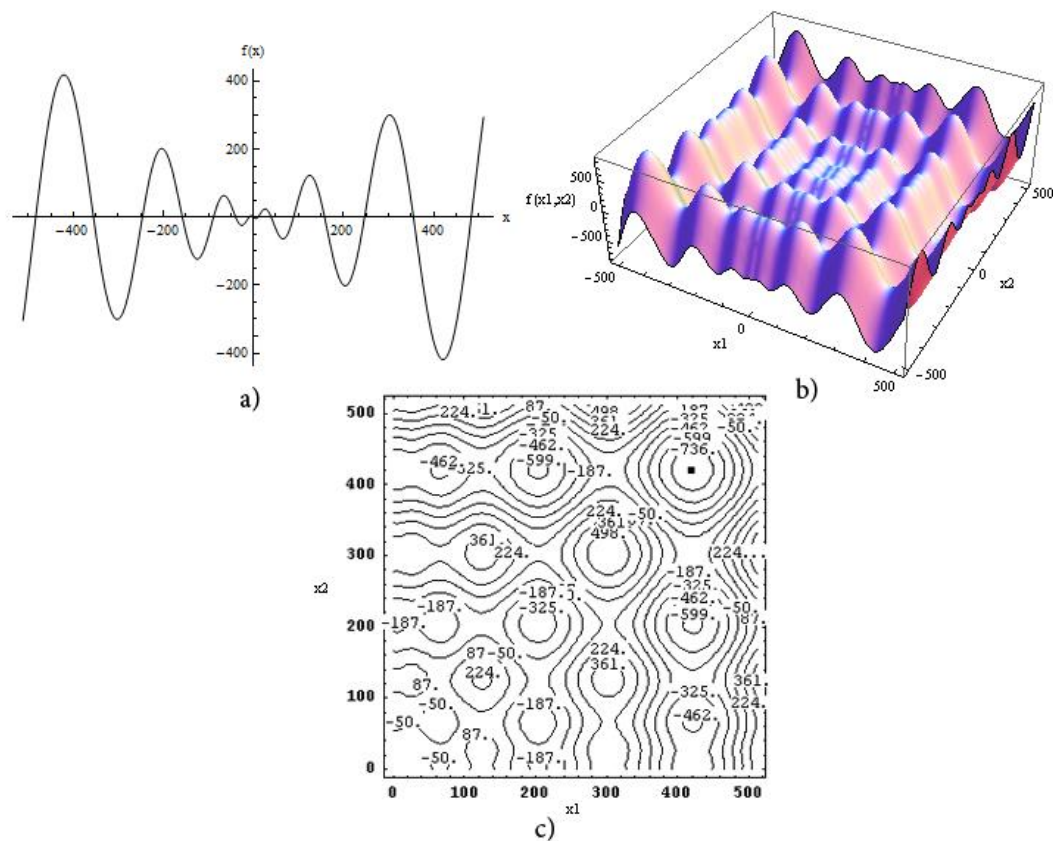
$$\text{Funkce: } f(\vec{x}) = \sum_{i=1}^D -x_i \cdot \sin(\sqrt{|x_i|}), \quad -512 \leq x_i \leq 512$$

$$\text{Globální minimum: } D1 \rightarrow f(\vec{x}) = -418.983 \text{ na pozici } \{x_1 = 420.969\}.$$

$$D2 \rightarrow f(\vec{x}) = -837.966 \text{ na pozici } \{x_1 = 420.969, x_2 = 420.969\}.$$

$$Dn \rightarrow f(\vec{x}) = n \cdot f(x_1) = n \cdot (-418.983) \text{ na pozici } \{x_1, \dots, x_n = 420.969\}.$$

Příklad(y) vykreslení funkce Schwefel je (jsou) zobrazen(y) na Obr. 23.



Obrázek 23 - Zobrazení funkce Schwefel

7.2.4 Rastrigin

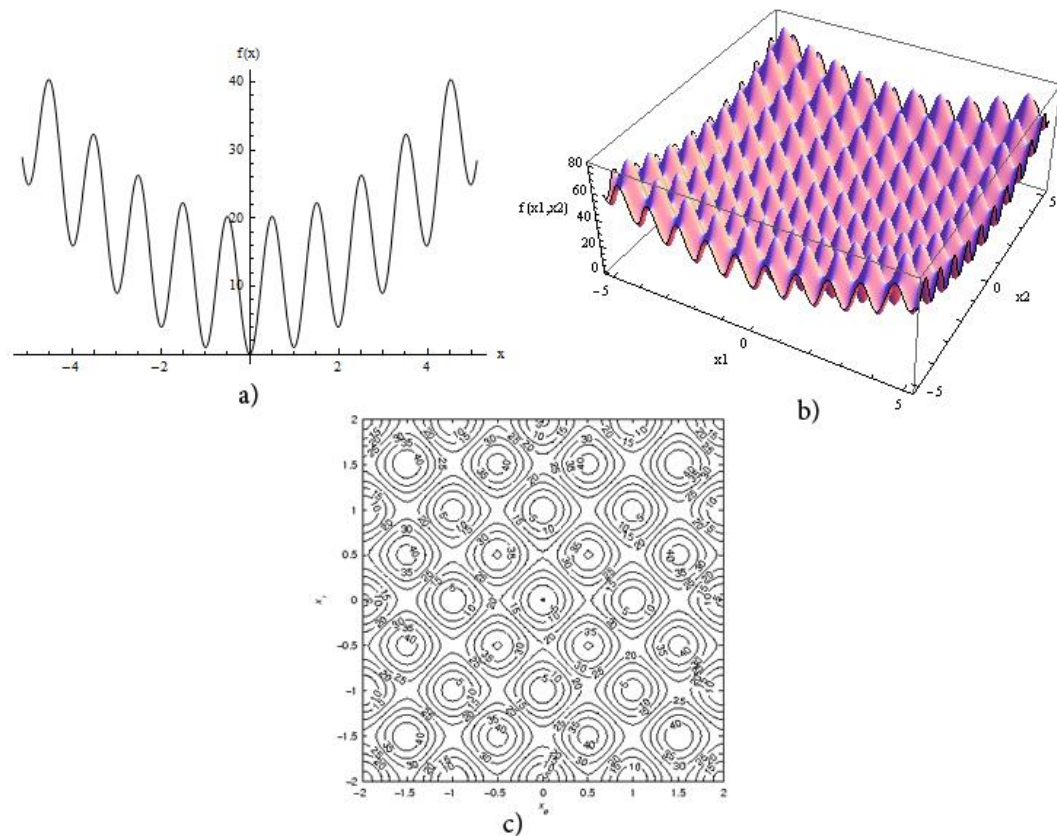
$$\text{Funkce: } f(\vec{x}) = 10D + \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i)), \quad -5.12 \leq x_i \leq 5.12$$

Globální minimum: $D1 \rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 0\}$.

$D2 \rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 0, x_2 = 0\}$.

$Dn \rightarrow f(\vec{x}) = n \cdot f(x_1) = 0$ na pozici $\{x_1, \dots, x_n = 0\}$.

Příklad(y) vykreslení funkce Rastrigin je (jsou) zobrazen(y) na Obr. 24.



Obrázek 24 - Zobrazení funkce Rastrigin

7.2.5 Shifted 1st De Jong

Funkce: $f(\vec{x}) = \sum_{i=1}^D (x_i - \text{shift}_i)^2$, $-5.12 \leq x_i \leq 5.12$

Globální minimum: D1 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = \text{shift}_1\}$.

D2 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = \text{shift}_1, x_2 = \text{shift}_2\}$.

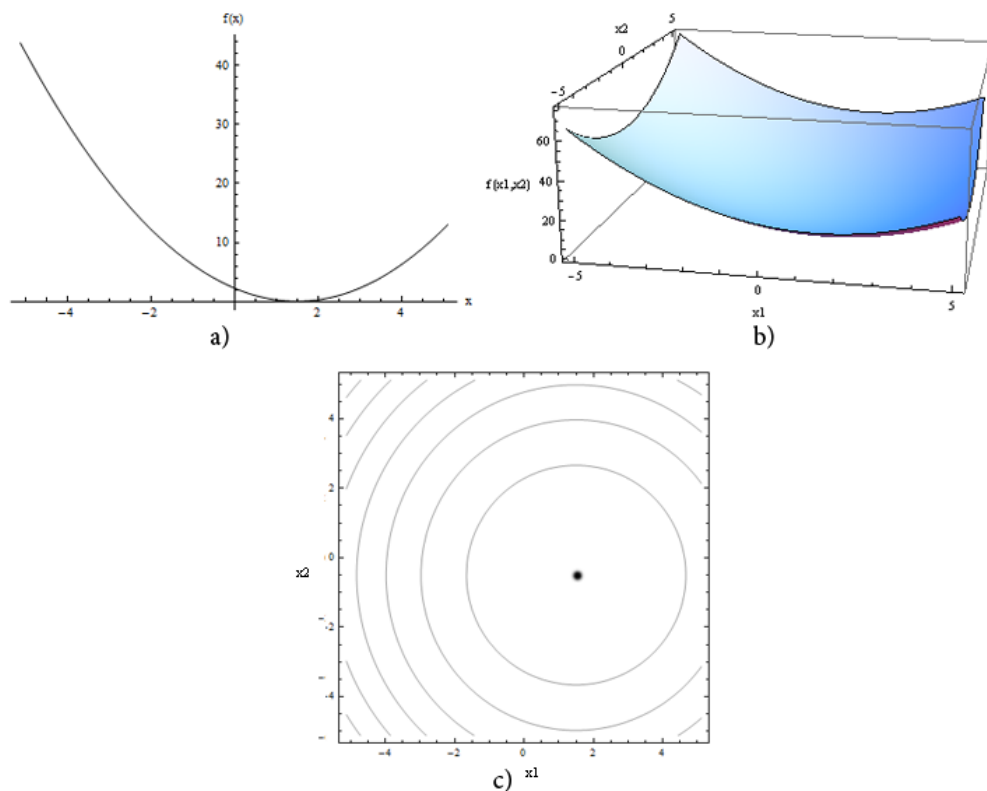
Dn $\rightarrow f(\vec{x}) = n \cdot f(x_1) = 0$ na pozici $\{x_1, \dots, x_n = \text{shift}_1, \dots, \text{shift}_n\}$.

V případě vygenerování $\text{shift}_1 = 1.5$ a $\text{shift}_2 = -0.5$.

Je-li $\text{shift}_1 = 1.5$, minimum se posune o hodnotu 1.5. V případě shifted 1st De Jong bude globální minimum v D1 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 1.5\}$,

D2 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 1.5, x_2 = -0.5\}$.

Příklad(y) vykreslení funkce shifted 1st De Jong je (jsou) zobrazen(y) na Obr. 25.



Obrázek 25 - Zobrazení funkce Shifted 1st De Jong

7.2.6 Shifted Rastrigin

Funkce: $f(\vec{x}) = 10D + \sum_{i=1}^D (x_i - \text{shift}_i)^2 - 10 \cos(2\pi(x_i - \text{shift}_i))$, $-5.12 \leq x_i \leq 5.12$

Globální minimum: D1 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = \text{shift}_1\}$.

D2 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = \text{shift}_1, x_2 = \text{shift}_2\}$.

Dn $\rightarrow f(\vec{x}) = n \cdot f(x_1) = 0$ na pozici $\{x_1, \dots, x_n = \text{shift}_1, \dots, \text{shift}_n\}$.

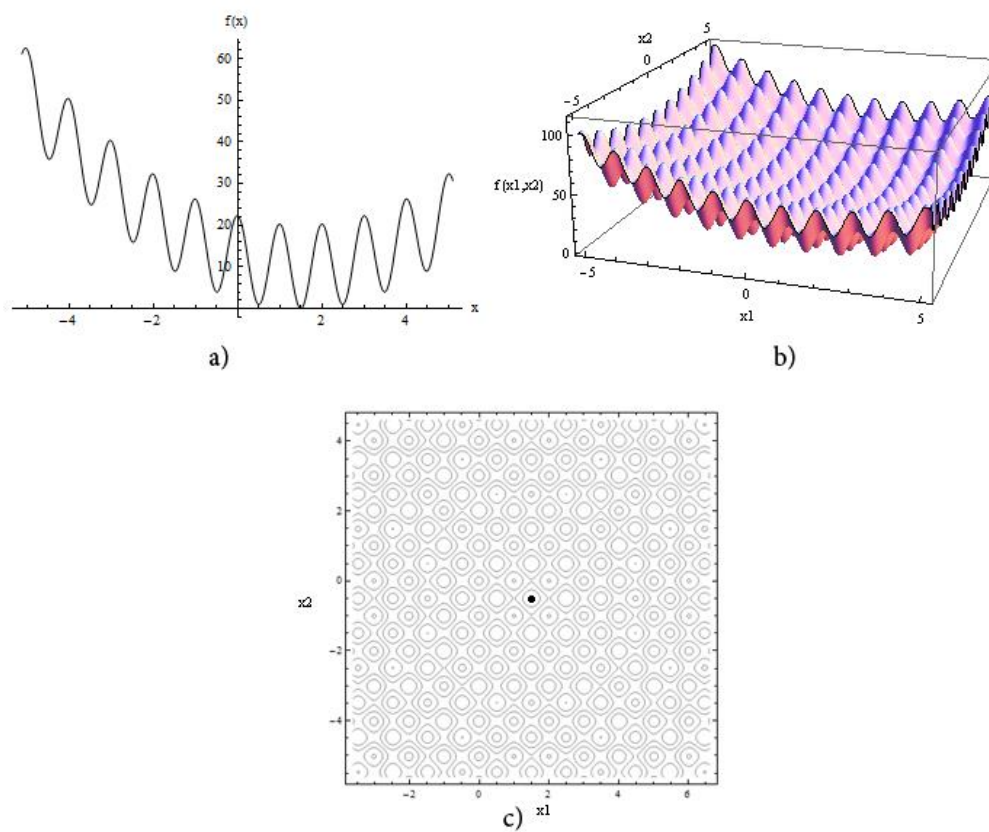
V případě vygenerování $\text{shift}_1 = 1.5$ a $\text{shift}_2 = -0.5$.

Stejný případ jako funkce shifted 1st De Jong. U 1D posunutí na ose x o danou hodnotu.

Posunutí rovněž o 1.5. D1 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 1.5\}$.

D2 $\rightarrow f(\vec{x}) = 0$ na pozici $\{x_1 = 1.5, x_2 = -0.5\}$.

Příklad(y) vykreslení funkce Shifted Rastrigin je (jsou) zobrazen(y) na Obr. 26.



Obrázek 26 - Zobrazení funkce Shifted Rastrigin

8 IMPLEMENTACE VČELÍHO ALGORITMU

Algoritmus je rozdělen na dvě základní části a to tělo algoritmu (Bee Body) a použití algoritmu (Bee Usage). V sekci Bee Body jsou implementovány jednotlivé základní funkce, viz kapitola 5.4.2 Životní cyklus algoritmu. Naproti tomu sekce Bee Usage tyto funkce používá, navíc i s dodatečnými potřebnými informacemi.

Ořezaná ukázka sekce Bee Usage (pouze s hlavními komponenty) na Obr. 27:

```
BeeAlgorithm[]
Initialize[];
For[it = 0, it < iteration, it++,
    SelectBest[];
    NeighborhoodSearchE[];
    NeighborhoodSearchO[];
    SelectNewBee[];
    NewBees[];
]
```

Obrázek 27 – Bee Algorithm v Mathematice

8.1 Rozbor funkcí

`Initialize[]` – Vygenerování počáteční populace ve formátu $\{CV, \{\text{souřadnice}\}\}$.

`SelectBest[]` – Výběr nejlepších včel pro okolní prohledávání.

`NeighborhoodSearchE[]` – Okolní prohledávání výše vybraných včel (elita).

`NeighborhoodSearchO[]` – Okolní prohledávání pro ty méně dobré včely ($m-e$).

`SelectNewBee[]` – Z každého okolního prohledávání jsou vybráni pouze ti nejlepší jedinci.

`NewBees[]` – Vygenerování nových včel ($n-m$), aby se velikost celkové populace opět rovnala n jedinců.

8.2 Adaptive Bee Algorithm

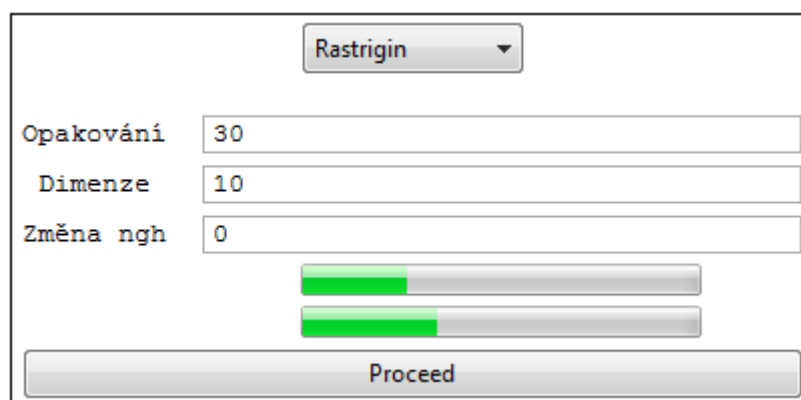
V kapitole 5.4 bylo vysvětleno i s názorným příkladem, jak Včelí algoritmus pracuje. Verze algoritmu (Bee Algorithm) pracuje podle tohoto příkladu. Před spuštěním algoritmu se nastaví parametr *ngh* na určitou mez a během chodu algoritmu se již nemění. Parametr *ngh*

udává, v jak velkém okolí nejlepších jedinců se bude nově prohledávat v naději nalezení lepší CV.

Testy ovšem ukázaly, že ve většině případů je lepší, aby se tento parametr v průběhu výpočtu měnil (zmenšoval). Proto je tato základní verze algoritmu mírně modifikována o tuto úpravu. Tento změněný algoritmus se zde bude nazývat Adaptive Bee Algorithm.

Při testování testovacích funkcí tento upravený algoritmus vykazoval lepší výsledky. A proto bylo při testování pracováno s oběma verzemi algoritmu a následně do jednoho grafu pro srovnání vykresleny oba průběhy výpočtů.

Spuštění samotného algoritmu závisí pouze na výběru testované funkce, počtu opakování, v jaké dimenzi se bude testování provádět a změna *ngh*, která byla zmíněna výše. Toto číslo udává, kolikrát se smí opakovat CV (nejsou-li stále nalezena lepší řešení), než se okolí zmenší o určitou hodnotu. Je-li nastavena nula, jak lze vidět na Obr. 28, jakoby tam změna ani nebyla a jedná se o klasický Bee Algorithm.



The screenshot shows a software interface for the Adaptive Bee Algorithm (BA) in Mathematica. At the top, a dropdown menu is set to 'Rastrigin'. Below it, there are three input fields: 'Opakování' (Iterations) with the value 30, 'Dimenze' (Dimension) with the value 10, and 'Změna ngh' (Change ngh) with the value 0. Under these fields are two horizontal progress bars, both of which are partially filled with green. At the bottom of the interface is a large button labeled 'Proceed'.

Obrázek 28 – Použití BA v Mathematice

Na obrázku lze navíc nalézt dva progress bary, které slouží pouze k informativnímu účelu pro uživatele a říkají, ve které fázi výpočtu se algoritmus nachází. První ukazuje počet opakování a druhý počet generací/migrací.

9 VÝSLEDKY TESTOVÁNÍ

Jednotlivé testovací funkce byly otestovány a jejich výsledky znázorněny. Kvůli velkému počtu grafů, byly vyneseny pouze průběhy z každé testovací funkce pro 5D jako zástupce nižších dimenzí a 30D jako zástupce vyšších dimenzí. Vykreslení ostatních průběhů budou k nalezení na přiloženém CD.

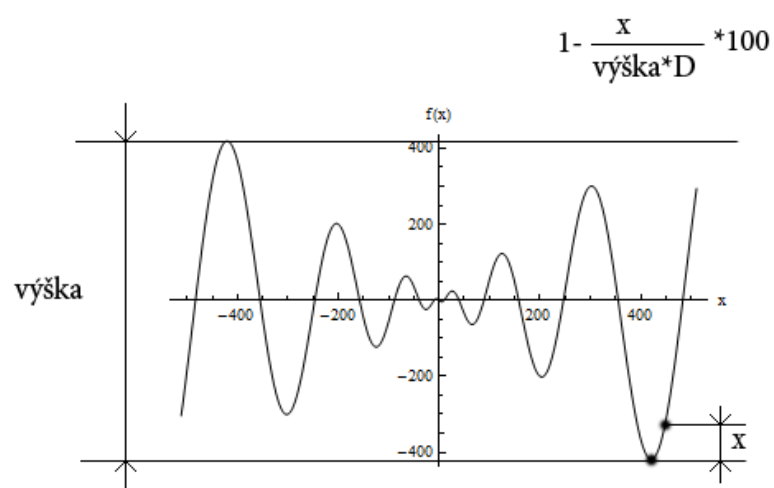
Pro každou testovací funkci je uvedena tabulka s výsledky algoritmů u jednotlivých dimenzí. Nejdříve jsou uvedeny tabulkové výsledky, poté srovnání grafů v 5 a 30 dimenzích.

Před vyobrazením jednotlivých průběhů funkcí je napsáno, s jakými nastavenými parametry (pro obě funkce) se pracovalo u dané testovací funkce a v dané dimenzi. Jako první jsou vyobrazeny parametry pro BA (n , m , e , $n2$, $n1$, ngh , $iterace$) – jednotlivé parametry jsou popsány v kapitole 5.4.1 Životní cyklus algoritmu. Dále jsou zobrazeny parametry pro SOMA ($PopSize$, $Migrations$, $Step$, $PathLength$) – popsáno v kapitole 3.3.2 Parametry.

Co se týče samotných zobrazení průběhů výsledků, levá strana se věnuje včelímu algoritmu a pravá algoritmu SOMA. Včelí algoritmus je navíc rozdělen na klasický včelí algoritmus (černá barva) a adaptivní včelí algoritmus (červená barva), aby lépe vynikly rozdíly mezi těmito dvěma algoritmy. Na ose y jsou ohodnocení účelové funkce (CV) a na ose x je počet ohodnocení účelové funkce (CFE) – více v kapitole 7.1 Charakteristika testu. Právě díky počtu ohodnocení účelové funkce lze snadno porovnat všechny tři typy algoritmu a zjistit, který z nich dává lepší výsledky. Pro každou testovací funkci jsou nejprve vypsáni nejlepší jedinci ze všech 30 opakování a následně zobrazeny grafy: nejlepší nalezený jedinec, průměr z 30 opakování a největší spád.

Ve výsledkové tabulce je pak modře znázorněna nejlepší nalezená hodnota účelové funkce z 30 opakování, žlutě pak procentuální úspěšnost algoritmu. Srovnání se provádí mezi 3 algoritmy: BA, SOMA a Adaptivní BA. U Adaptivního BA je u hodnoty dimenze znázorněna jedna hodnota navíc, jde o hodnotu, která říká, při kolika opakováních se změnila hodnota ngh .

Dále jsou uvedeny nejlepší hodnoty účelové funkce, počet ohodnocení účelové funkce, hodnota teoretického extrému, rozdíl mezi nejlepší a teoretickou hodnotou, průměr, průměr v určitém počtu ohodnocení účelové funkce a úspěšnost v procentech, která říká, jak se procentuálně nejlepší nalezená hodnota vzdaluje od globálního extrému (v rámci dané dimenze). Výpočet je znázorněn na Obr. 29.



Obrázek 29 - Výpočet úspěšnosti

9.1 1st De Jong

Dle výsledků lze usoudit, že nejlepších hodnot dosahovaly algoritmy SOMA a adaptivní BA. BA dosahoval dobrých výsledků pouze u méně dimenzionálních problémů. Klasický BA v 50D se k teoretickému extrému pouze přibližoval. Na této testovací funkci byl nastaven obdobný CFE u všech algoritmů. Jednotlivé výpočty lze zkontrolovat v Tabulce 3 a průběhy na Obrázcích 30 a 31.

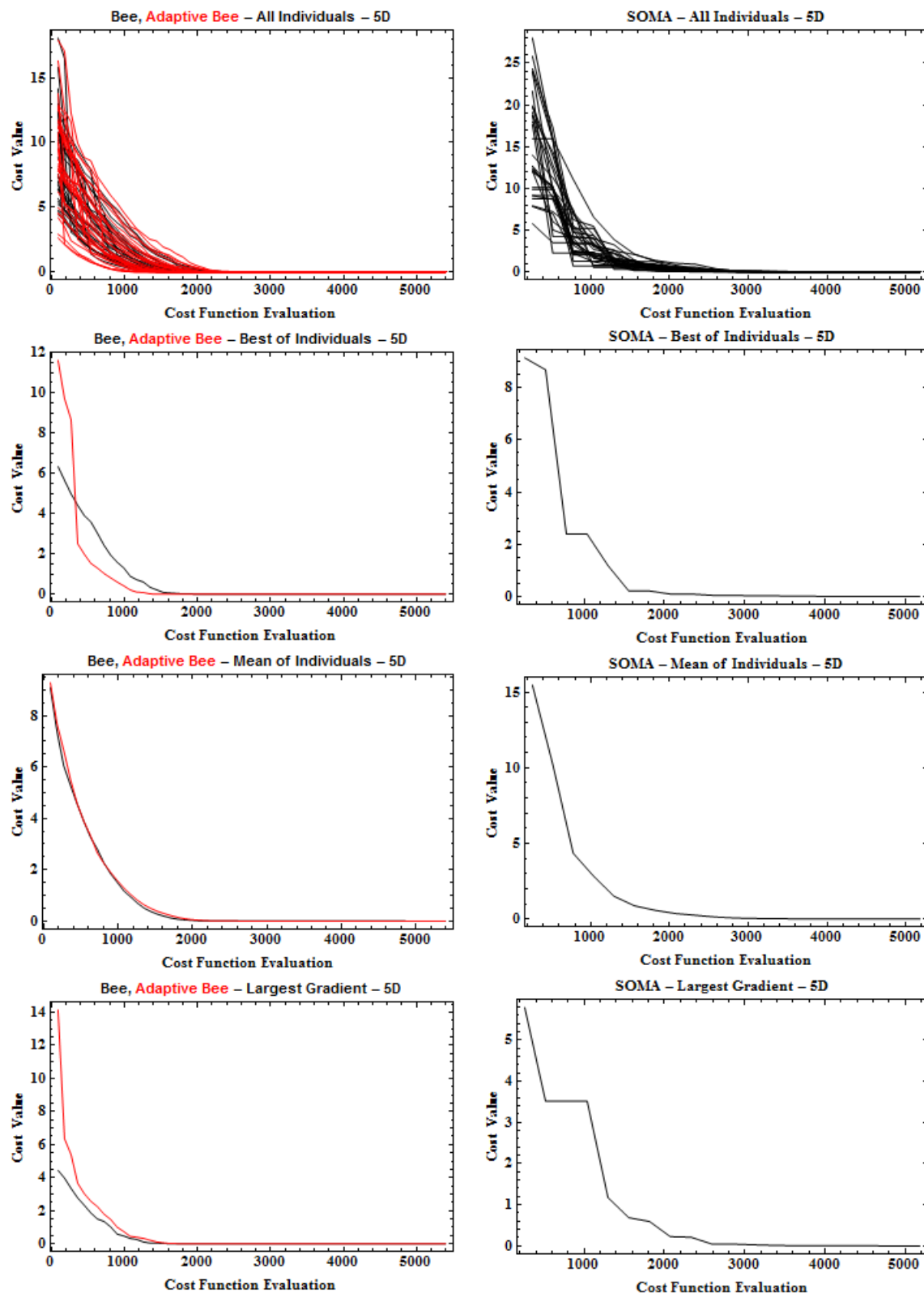
1st De Jong						
	Bee Algorithm					
Dimenze	2	5	10	20	30	50
Nejlepší CV	4,0427E-06	4,8303E-04	0,0745	0,3995	1,1683	3,6335
CFE	1 620	5 400	14 550	19 400	24 250	30 300
Teoretický extrém	0	0	0	0	0	0
Rozdíl	4,0427E-06	4,8303E-04	0,0745	0,3995	1,1683	3,6335
Průměr	1,5725E-04	2,3306E-03	0,1394	0,6507	1,6341	4,7930
Průměr CV v 10 000 CFE	-	-	0,1530	0,7483	2,0683	6,5929
Průměr CV v 20 000 CFE	-	-	-	-	1,7015	5,2372
% úspěšnost [%]	100	99,9996	99,9716	99,9238	99,8514	99,7228
	SOMA					
Dimenze	2	5	10	20	30	50
Nejlepší CV	2,4274E-11	4,6526E-05	2,5949E-07	1,1165E-05	6,0727E-05	4,8719E-04
CFE	1 636	5 182	14 500	19 509	24 255	30 318
Teoretický extrém	0	0	0	0	0	0
Rozdíl	2,4274E-11	4,6526E-05	2,5949E-07	1,1165E-05	6,0727E-05	4,8719E-04
Průměr	1,5785E-07	3,6660E-04	6,7141E-07	3,8878E-05	1,2638E-04	8,7559E-04
Průměr CV v 10 000 CFE	-	-	0,0002	0,0673	0,5952	4,4783
Průměr CV v 20 000 CFE	-	-	-	-	0,0017	0,0685
% úspěšnost	100	100	100	100	100	100
	Adaptive Bee Algorithm					
Dimenze/ngn	2/3	5/3	10/3	20/3	30/3	50/3
Nejlepší CV	9,9188E-09	4,3370E-06	4,1936E-09	1,0775E-07	4,3776E-07	5,1007E-05
CFE	1 620	5 400	14 550	19 400	24 250	30 300
Teoretický extrém	0	0	0	0	0	0
Rozdíl	9,9188E-09	4,3370E-06	4,1936E-09	1,0775E-07	4,3776E-07	5,1007E-05
Průměr	4,3547E-05	3,8324E-05	9,8398E-08	1,7123E-06	4,7512E-06	2,3361E-04
Průměr CV v 10 000 CFE	-	-	2,9821E-05	3,7583E-03	0,0517	1,4278
Průměr CV v 20 000 CFE	-	-	-	-	7,2135E-05	0,0207
% úspěšnost [%]	100	100	100	100	100	100

Tabulka 3 – Vyhodnocení funkce 1st De Jong

9.1.1 5. dimenze

1st De Jong - 5D

$n=30, m=3, e=2, n2=15, n1=3, ngh=0.15, iter=60$
 PopSize=20, Migrations=20, Step=0.22, PathLength=3.

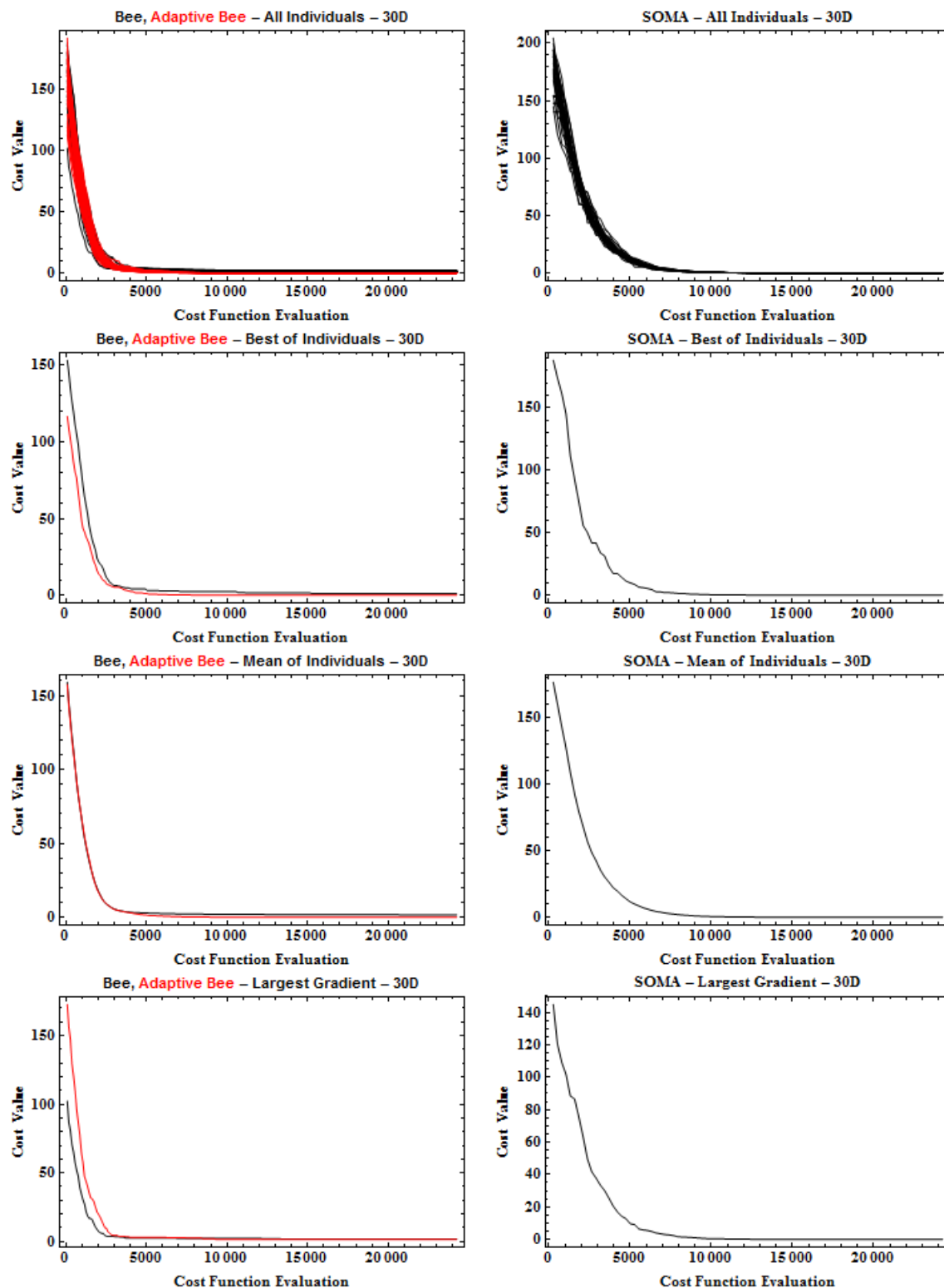


Obrázek 30 - Průběh výsledků pro 1st De Jong - 5D

9.1.2 30. dimenze

1st De Jong - 30D

n=30, m=3, e=1, n2=30, n1=5, ngh=0.5, iter=250
 PopSize=30, Migrations=92, Step=0.22, PathLength=2.



Obrázek 31 - Průběh výsledků pro 1st De Jong - 30D

9.2 2nd De Jong

Z Tabulky 4 i grafů (viz Obr. 32 a 33) lze názorně vidět, že adaptivní BA vykazuje mnohem lepší výsledky u více dimenzionálních problémů než klasický BA. Pokud je porovnán adaptivní BA a SOMA, mírně lepší výsledky jsou zaznamenány u adapt. BA. Opět byl zachován podobný CFE, kromě 50D, kde u SOMA bylo nastaveno vyšší CFE.

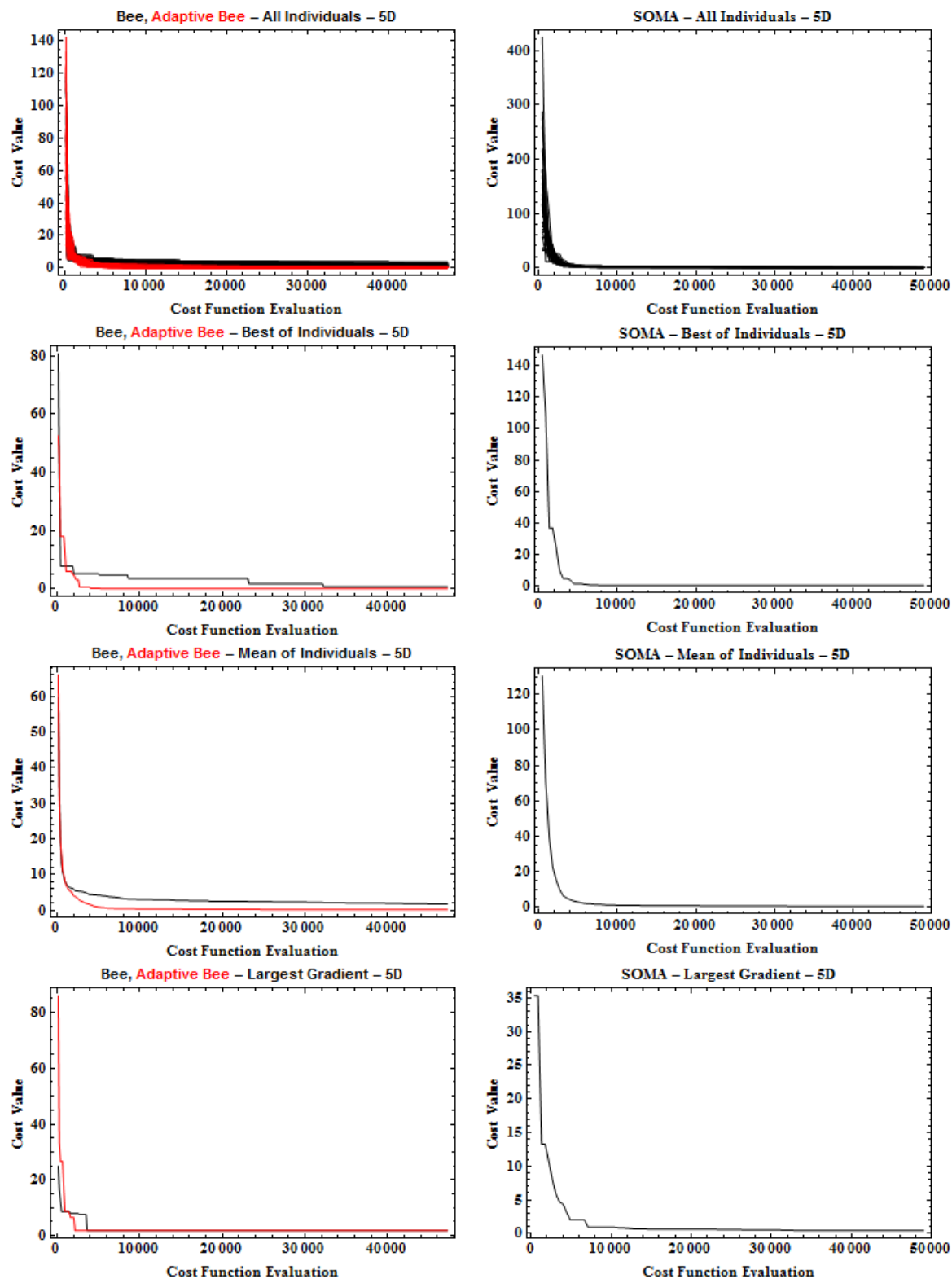
2nd De Jong						
	Bee Algorithm					
Dimenze	2	5	10	20	30	50
Nejlepší CV	0	0,7169	8,9671	52,3078	306,7067	1 233,9851
CFE	2 460	47 250	441 000	759 000	922 500	830 000
Teoretický extrém	0	0	0	0	0	0
Rozdíl	0	0,7169	8,9671	52,3078	306,7067	1 233,9851
Průměr	0,0043	1,7640	12,3531	59,2628	409,0949	1 493,0897
Průměr CV v 400 000 CFE	-	-	12,3589	62,5894	456,0241	1 628,6341
Průměr CV v 800 000 CFE	-	-	-	-	418,0739	1 500,2144
% úspěšnost	100	99,9991	99,9974	99,9965	99,9910	99,9871
	SOMA					
Dimenze	2	5	10	20	30	50
Nejlepší CV	1,4981E-07	0,0489	0,9815	10,8026	20,2026	42,1758
CFE	5 182	49 000	360 000	758 545	925 418	1 041 273
Teoretický extrém	0	0	0	0	0	0
Rozdíl	1,4981E-07	0,0489	0,9815	10,8026	20,2026	42,1758
Průměr	4,5483E-03	0,5367	3,2627	13,0921	23,0409	43,9392
Průměr CV v 400 000 CFE	-	-	-	13,4192	26,7990	45,2735
Průměr CV v 800 000 CFE	-	-	-	-	23,2618	44,1253
% úspěšnost	100	99,9999	99,9997	99,9993	99,9994	99,9996
	Adaptive Bee Algorithm					
Dimenze/ngn	2/1	5/3	10/3	20/3	30/3	50/3
Nejlepší CV	3,7302E-07	1,5083E-05	7,2898E-04	0,3662	0,5611	29,1282
CFE	2 460	47 250	441 000	759 000	922 500	830 000
Teoretický extrém	0	0	0	0	0	0
Rozdíl	3,7302E-07	1,5083E-05	7,2898E-04	0,3662	0,5611	29,1282
Průměr	2,0705E-03	0,1532	0,3398	7,4808	14,3104	37,1682
Průměr CV v 400 000 CFE	-	-	0,4154	10,9231	20,5191	42,2670
Průměr CV v 800 000 CFE	-	-	-	-	15,4729	37,4874
% úspěšnost	100	100	100	100	100	99,9997

Tabulka 4 – Vyhodnocení funkce 2nd De Jong

9.2.1 5. dimenze

2nd De Jong - 5D

n=50, m=10, e=5, n2=7, n1=2, ngh=0.5, iter=350
 PopSize=50, Migrations=110, Step=0.22, PathLength=2.

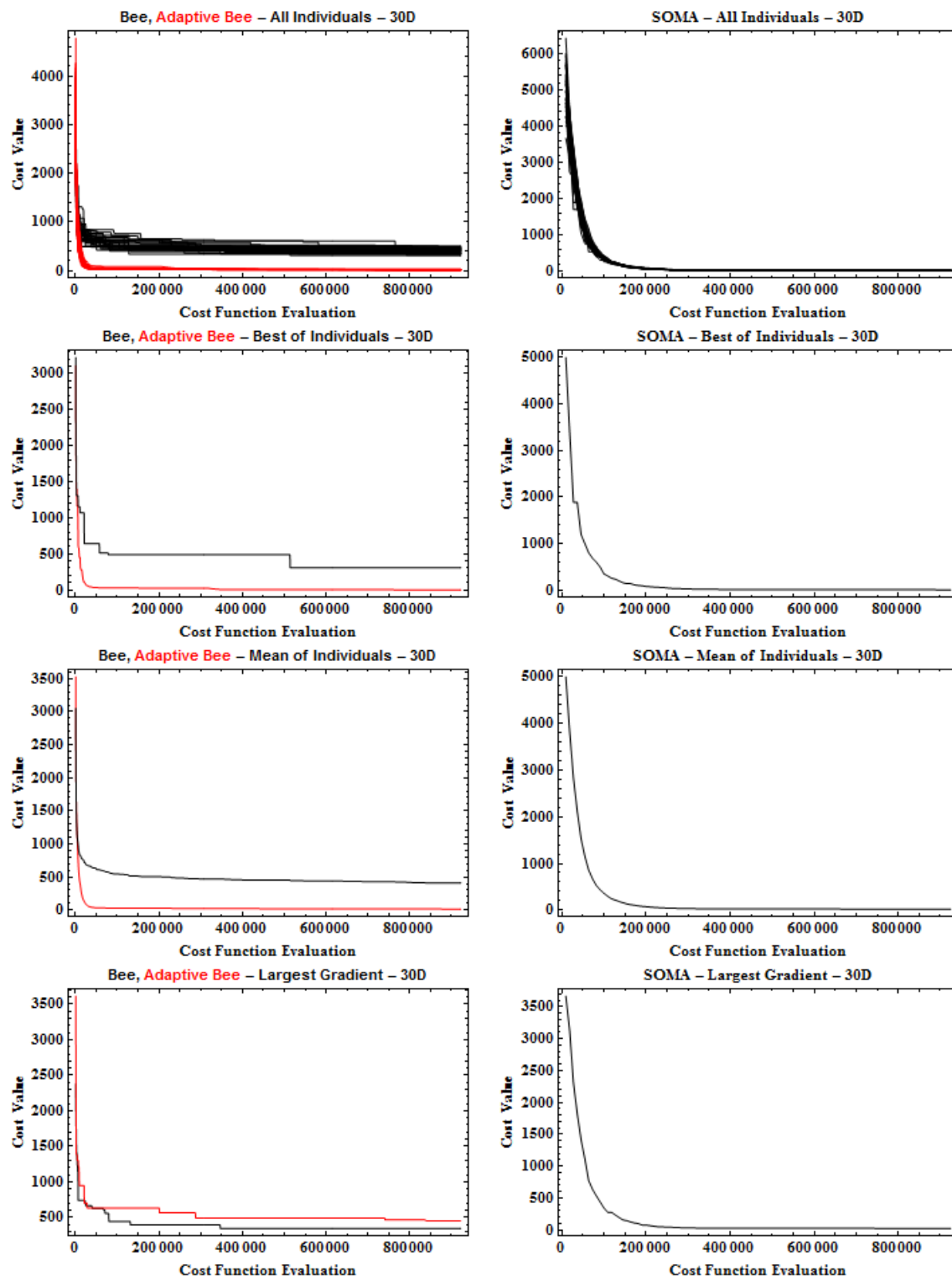


Obrázek 32 - Průběh výsledků pro 2nd De Jong - 5D

9.2.2 30. dimenze

2nd De Jong - 30D

n=50, m=5, e=1, n2=500, n1=5, ngh=1, iter=1500
 PopSize=500, Migrations=102, Step=0.11, PathLength=2.



Obrázek 33 - Průběh výsledků pro 2nd De Jong - 30D

9.3 Schwefel

U multimodální funkce s více extrémy již lze upozorovat, že oba včelí algoritmy nestačí na vícedimenzionální problémy. Pro dimenze 2D a 5D je nalezená hodnota ještě dostačující. Adaptivní BA je o něco lepší než klasický BA, ale i tak na algoritmus SOMA nestačí. Algoritmus SOMA zvládá řešení bez problémů i s mnohem menším CFE (viz Tabulka 5). Jednotlivé průběhy algoritmů lze vidět na Obr. 34 a 35.

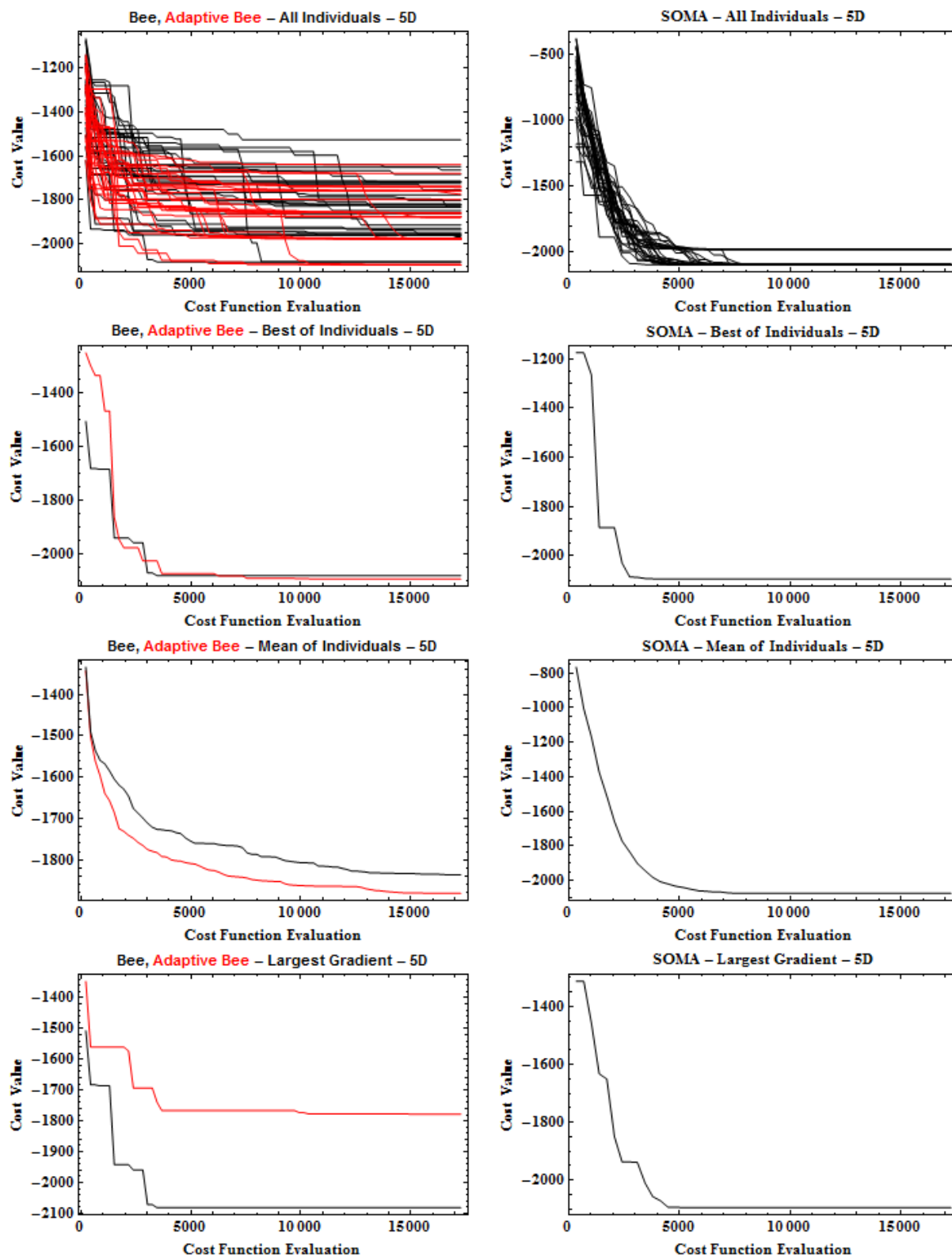
Schwefel						
	Bee Algorithm					
Dimenze	2	5	10	20	30	50
Nejlepší CV	-837,9517	-2 081,6363	-3 830,5366	-5 997,4060	-8 494,2464	-11 631,9204
CFE	1 925	17 280	843 500	2 187 500	9 120 000	4 400 000
Teoretický extrém	-837,9658	-2 094,9144	-4 189,8289	-8 379,6577	-12 569,4866	-20 949,1444
Rozdíl	0,0140	13,2782	359,2922	2 382,2517	4 075,2402	9 317,2240
Průměr	-836,8213	-1 836,5546	-3 474,3548	-5 444,4313	-7 522,8020	-10 696,3958
Průměr CV v 300 000 CFE	-	-	-3 403,3232	-5 299,9159	-7 085,7977	-9 917,4406
Průměr CV v 700 000 CFE	-	-	-3 450,3766	-5 368,3007	-7 186,8350	-10 210,0542
% úspěšnost	99,9983	99,3662	91,4247	71,5710	67,5783	55,5246
	SOMA					
Dimenze	2	5	10	20	30	50
Nejlepší CV	-837,9658	-2 094,9144	-4 189,8289	-8 379,6577	-12 569,4865	-20 949,1444
CFE	3 455	17 273	270 000	542 727	361 818	723 636
Teoretický extrém	-837,9658	-2 094,9144	-4 189,8289	-8 379,6577	-12 569,4866	-20 949,1444
Rozdíl	0	0	0	0	0,0002	0
Průměr	-826,2458	-2 075,1747	-4 189,8289	-8 379,6577	-12 569,4862	-20 949,1444
Průměr CV v 300 000 CFE	-	-	-	-8 379,6577	-12 569,4601	-20 935,3895
Průměr CV v 700 000 CFE	-	-	-	-	-	-20 949,1444
% úspěšnost	100	100	100	100	100	100
	Adaptive Bee Algorithm					
Dimenze/ngn	2/1	5/5	10/50	20/100	30/300	50/300
Nejlepší CV	-837,9658	-2 094,9027	-3 950,9077	-6 771,1706	-9 788,9576	-15 271,1869
CFE	1 925	17 280	843 500	2 187 500	9 120 000	4 400 000
Teoretický extrém	-837,9658	-2 094,9144	-4 189,8289	-8 379,6577	-12 569,4866	-20 949,1444
Rozdíl	0,0000	0,0117	238,9211	1 608,4871	2 780,5290	5 677,9574
Průměr	-836,2903	-1 881,3109	-3 599,7105	-6 163,4259	-8 933,8496	-14 185,6500
Průměr CV v 300 000 CFE	-	-	-3 499,3937	-5 868,3609	-7 188,1338	-10 699,1656
Průměr CV v 700 000 CFE	-	-	-3 576,1231	-6 117,5396	-7 644,0595	-12 286,9756
% úspěšnost	100	99,9994	94,2976	80,8049	77,8787	72,8965

Tabulka 5 - Vyhodnocení funkce Schwefel

9.3.1 5. dimenze

Schwefel - 5D

$n=50, m=4, e=2, n2=50, n1=10, ngh=50, iter=80$
 PopSize=20, Migrations=50, Step=0.11, PathLength=2.

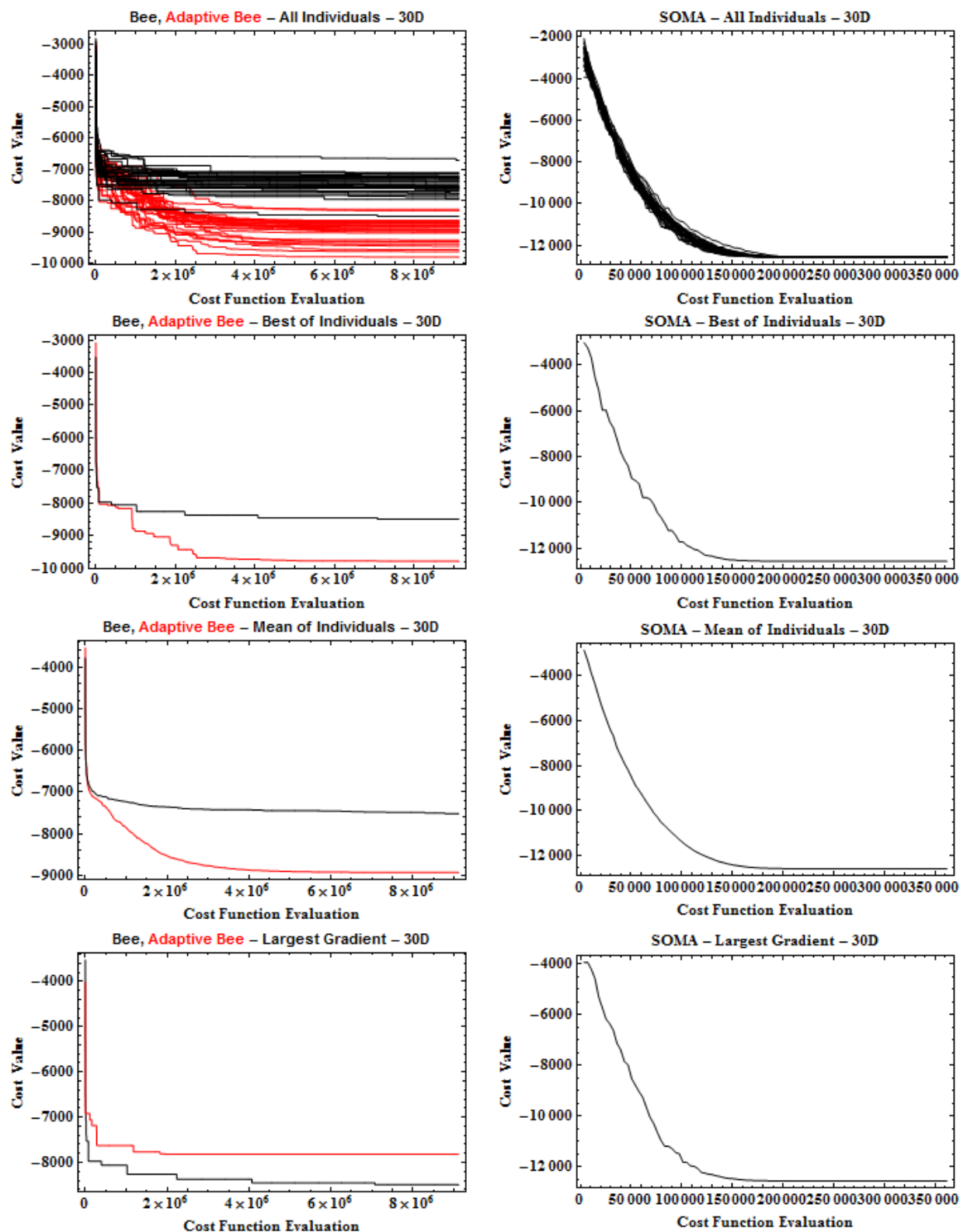


Obrázek 34 - Průběh výsledků pro Schwefel - 5D

9.3.2 30. dimenze

Schwefel - 30D

$n=30$, $m=20$, $e=10$, $n_2=100$, $n_1=10$, $n_{gh}=50$, $iter=8000$
 PopSize=200, Migrations=100, Step=0.11, PathLength=2.



Obrázek 35 - Průběh výsledků pro Schwefel - 30D

9.4 Rastrigin

Adaptivní BA je opět lepší než klasický BA. Názorně to lze vidět na Obr. 36 a 37. Jak již bylo výše zmíněno, oba dva BA nejsou vhodné pro multimodální funkce. I při větším množství počtu ohodnocení účelové funkce uváznou v lokálním extrému. Naproti tomu SOMA přináší velmi dobré výsledky s minimem počtu ohodnocení účelové funkce (viz Tabulka 6).

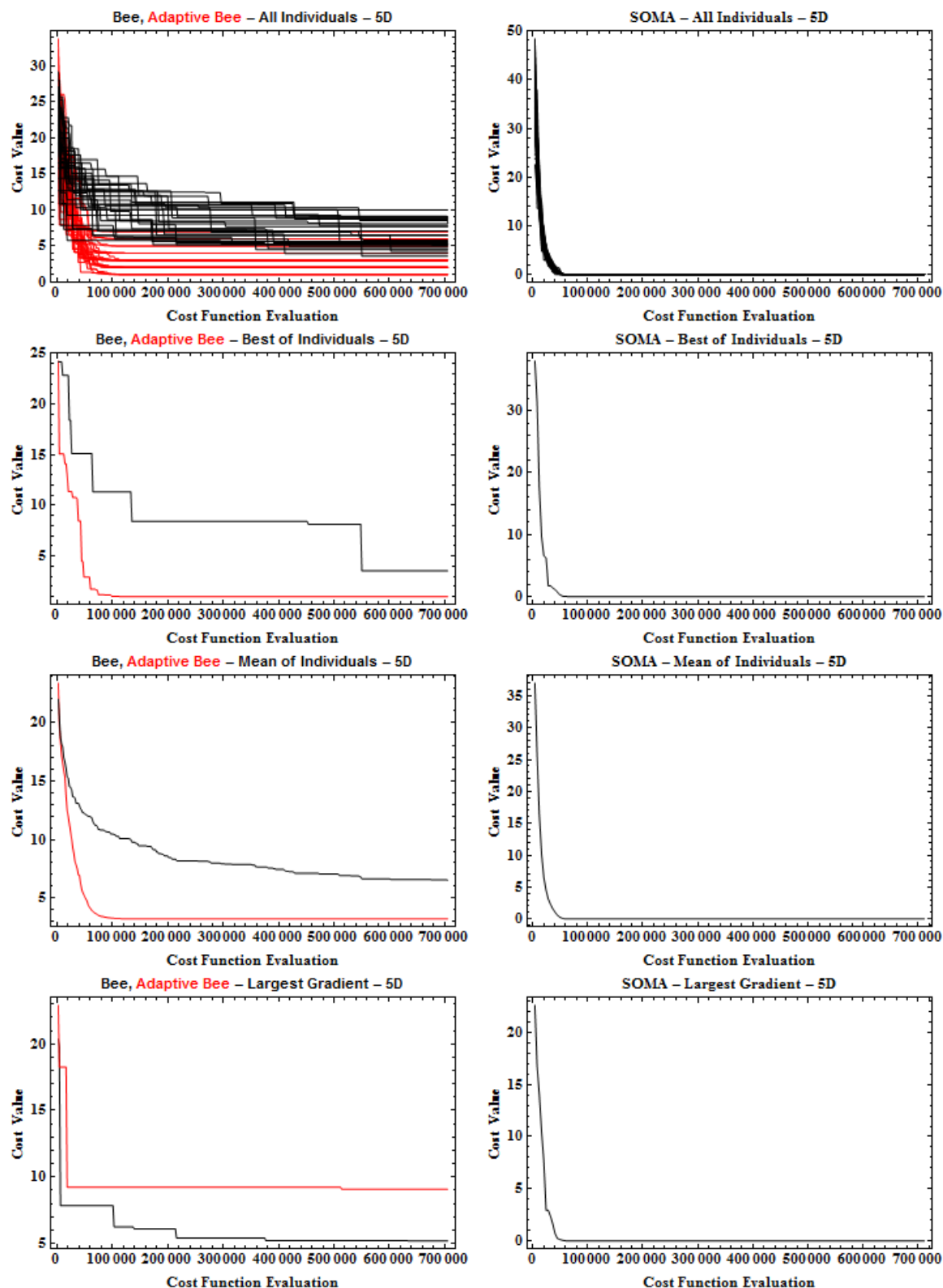
Rastrigin						
	Bee Algorithm					
Dimenze	2	5	10	20	30	50
Nejlepší CV	0,0071	3,5541	17,5622	90,0883	185,9455	384,0322
CFE	2 460	705 600	619 500	4 237 500	3 520 000	4 800 000
Teoretický extrém	0	0	0	0	0	0
Rozdíl	0,0071	3,5541	17,5622	90,0883	185,9455	384,0322
Průměr	0,8403	6,5201	27,9746	109,5480	200,3798	429,3672
Průměr CV v 300 000 CFE	-	7,9157	31,7394	133,2320	225,9932	477,2593
Průměr CV v 700 000 CFE	-	6,5547	-	124,3165	214,9140	458,2223
% úspěšnost	99,9912	98,2340	95,6367	88,8089	84,6008	80,9177
	SOMA					
Dimenze	2	5	10	20	30	50
Nejlepší CV	1,1828E-06	0	0	3,8181E-18	8,6175E-10	2,0829E-08
CFE	3 455	711 136	542 727	542 727	542 727	723 636
Teoretický extrém	0	0	0	0	0	0
Rozdíl	1,1828E-06	0	0	3,8181E-18	8,6175E-10	2,0829E-08
Průměr	0,0727	0	0	9,6470E-18	7,1612E-09	0,3980
Průměr CV v 300 000 CFE	-	0	7,4039E-13	3,6239E-09	0,0219	8,0024
Průměr CV v 700 000 CFE	-	0	-	-	-	0,3980
% úspěšnost	100	100	100	100	100	100
	Adaptive Bee Algorithm					
Dimenze/ngh	2/1	5/3	10/3	20/20	30/250	50/500
Nejlepší CV	2,6670E-07	0,9950	8,9546	39,8066	43,2358	110,9887
CFE	2 460	705 600	619 500	4 237 500	3 520 000	4 800 000
Teoretický extrém	0	0	0	0	0	0
Rozdíl	2,6670E-07	0,9950	8,9546	39,8066	43,2358	110,9887
Průměr	0,5307	3,2159	16,5495	55,2388	62,9434	140,3259
Průměr CV v 300 000 CFE	-	3,2170	16,5968	132,0498	227,6074	473,4079
Průměr CV v 700 000 CFE	-	3,2159	-	98,1812	199,4545	444,3881
% úspěšnost	100	99,5056	97,7752	95,0551	96,4194	94,4850

Tabulka 6 - Vyhodnocení funkce Rastrigin

9.4.1 5. dimenze

Rastrigin - 5D

$n=1000$, $m=6$, $e=2$, $n_2=7$, $n_1=2$, $ngh=0.5$, $iter=350$
 PopSize=150, Migrations=175, Step=0.11, PathLength=3.

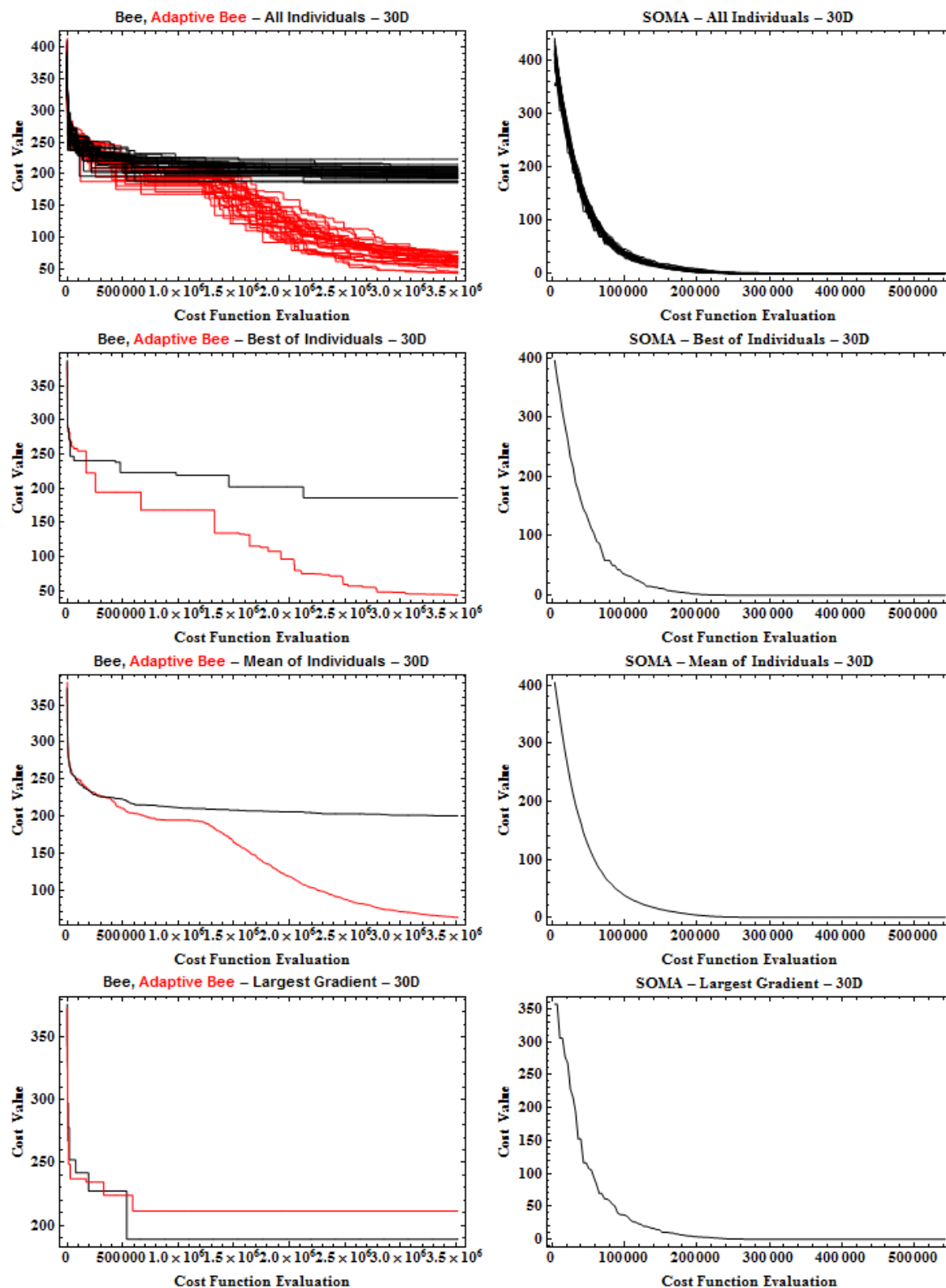


Obrázek 36 - Průběh výsledků pro Rastrigin - 5D

9.4.2 30. dimenze

Rastrigin - 30D

$n=30$, $m=20$, $e=10$, $n2=30$, $n1=10$, $ngh=2$, $iter=8000$
 PopSize=200, Migrations=150, Step=0.11, PathLength=2.



Obrázek 37 - Průběh výsledků pro Rastrigin - 30D

9.5 Shifted 1st De Jong

Jako v případě testování 1st De Jong, nemá posunutí (shifted) žádný vliv na výsledky řešení. Klasický BA je opět o něco horší než zbývající dva algoritmy, ale je možné vidět, že se k řešení bezprostředně přibližuje. SOMA a adaptivní BA jsou srovnatelné (viz Tabulka 7). Výpočty byly prováděny s podobnými CFE. Průběhy 5D a 30D jsou k nahlédnutí na Obr. 38 a Obr. 39.

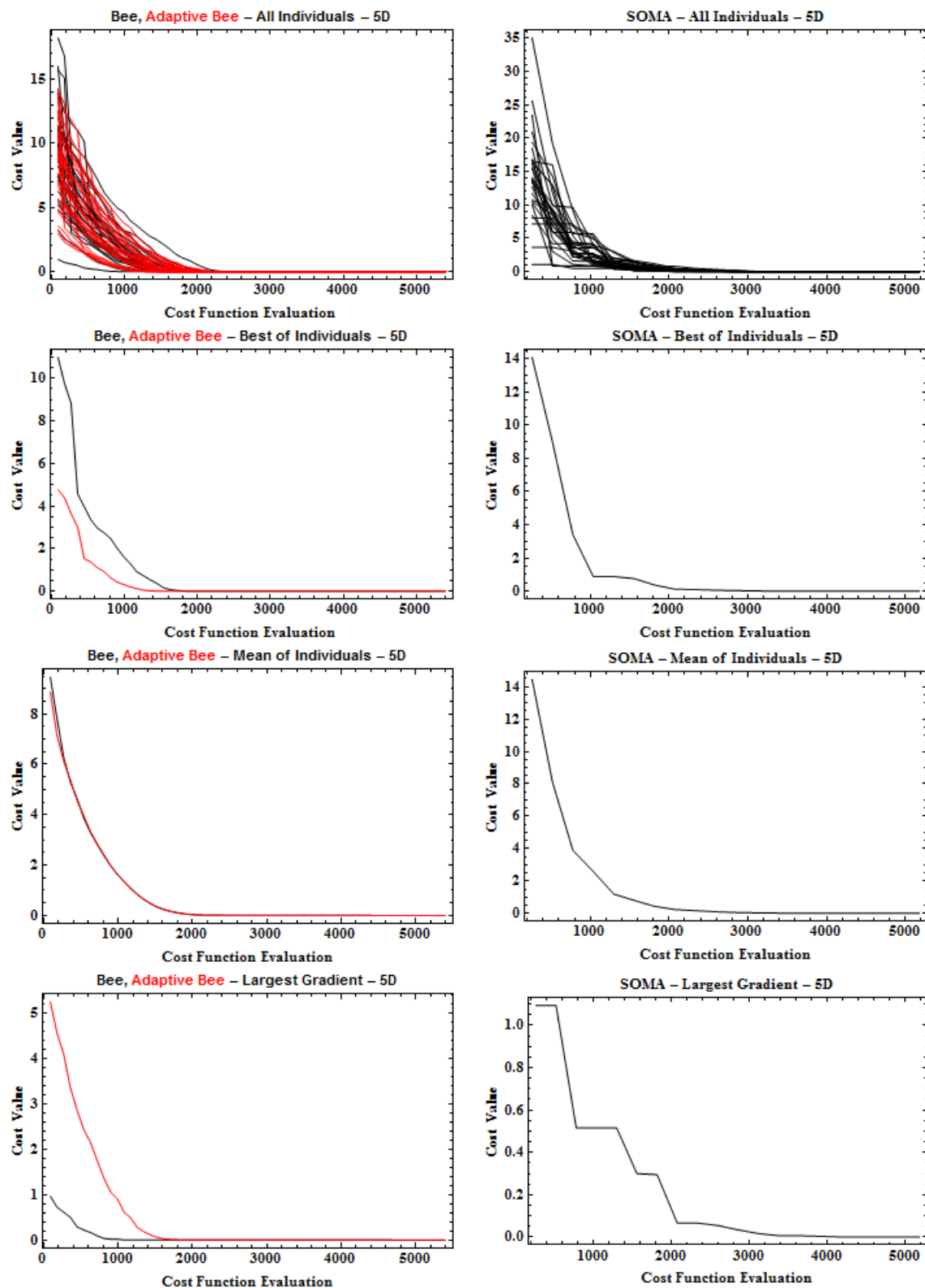
Shifted 1st De Jong						
	Bee Algorithm					
Dimenze	2	5	10	20	30	50
Nejlepší CV	1,1374E-05	2,1325E-04	0,0782	0,4823	1,2725	3,8518
CFE	1 620	5 400	14 550	19 400	24 250	30 300
Teoretické min.	0	0	0	0	0	0
Rozdíl	1,1374E-05	2,1325E-04	0,0782	0,4823	1,2725	3,8518
Průměr	2,9654E-04	2,0266E-03	0,1441	0,6595	1,5212	4,8229
Průměr CV v 10 000 CFE	-	-	0,1563	0,7927	1,9684	6,7165
Průměr CV v 20 000 CFE	-	-	-	-	1,6134	5,3078
% úspěšnost [%]	100	99,9998	99,9702	99,9080	99,8382	99,7061
	SOMA					
Dimenze	2	5	10	20	30	50
Nejlepší CV	3,0324E-06	4,5288E-05	1,7106E-07	2,4243E-05	9,1307E-05	5,0740E-04
CFE	818	5 182	14 500	19 509	24 255	30 318
Teoretické min.	0	0	0	0	0	0
Rozdíl	3,0324E-06	4,5288E-05	1,7106E-07	2,4243E-05	9,1307E-05	5,0740E-04
Průměr	0,0008	0,0003	0	0	0,0001	0,0010
Průměr CV v 10 000 CFE	-	-	0,0003	0,0742	0,6615	5,5567
Průměr CV v 20 000 CFE	-	-	-	-	0,0018	0,0835
% úspěšnost	100	100	100	100	100	100
	Adaptive Bee Algorithm					
Dimenze/ngħ	2/1	5/1	10/1	20/1	30/1	50/1
Nejlepší CV	4,8776E-10	7,2972E-11	1,0548E-17	2,3820E-13	7,5636E-12	1,9795E-08
CFE	1 620	5 400	14 550	19 400	24 250	30 300
Teoretické min.	0	0	0	0	0	0
Rozdíl	4,8776E-10	7,2972E-11	1,0548E-17	2,3820E-13	7,5636E-12	1,9795E-08
Průměr	3,0367E-08	3,1901E-08	2,0329E-15	2,8445E-12	9,1392E-11	1,3198E-07
Průměr CV v 10 000 CFE	-	-	1,6427E-10	1,2691E-05	1,5662E-03	0,2855
Průměr CV v 20 000 CFE	-	-	-	-	1,4811E-08	2,4371E-04
% úspěšnost [%]	100	100	100	100	100	100

Tabulka 7 - Vyhodnocení funkce Shifted 1st De Jong

9.5.1 5. dimenze

Shifted 1st De Jong - 5D

$n=30$, $m=3$, $e=2$, $n2=15$, $n1=3$, $ngh=0.15$, $iter=60$
 PopSize=20, Migrations=20, Step=0.22, PathLength=3.

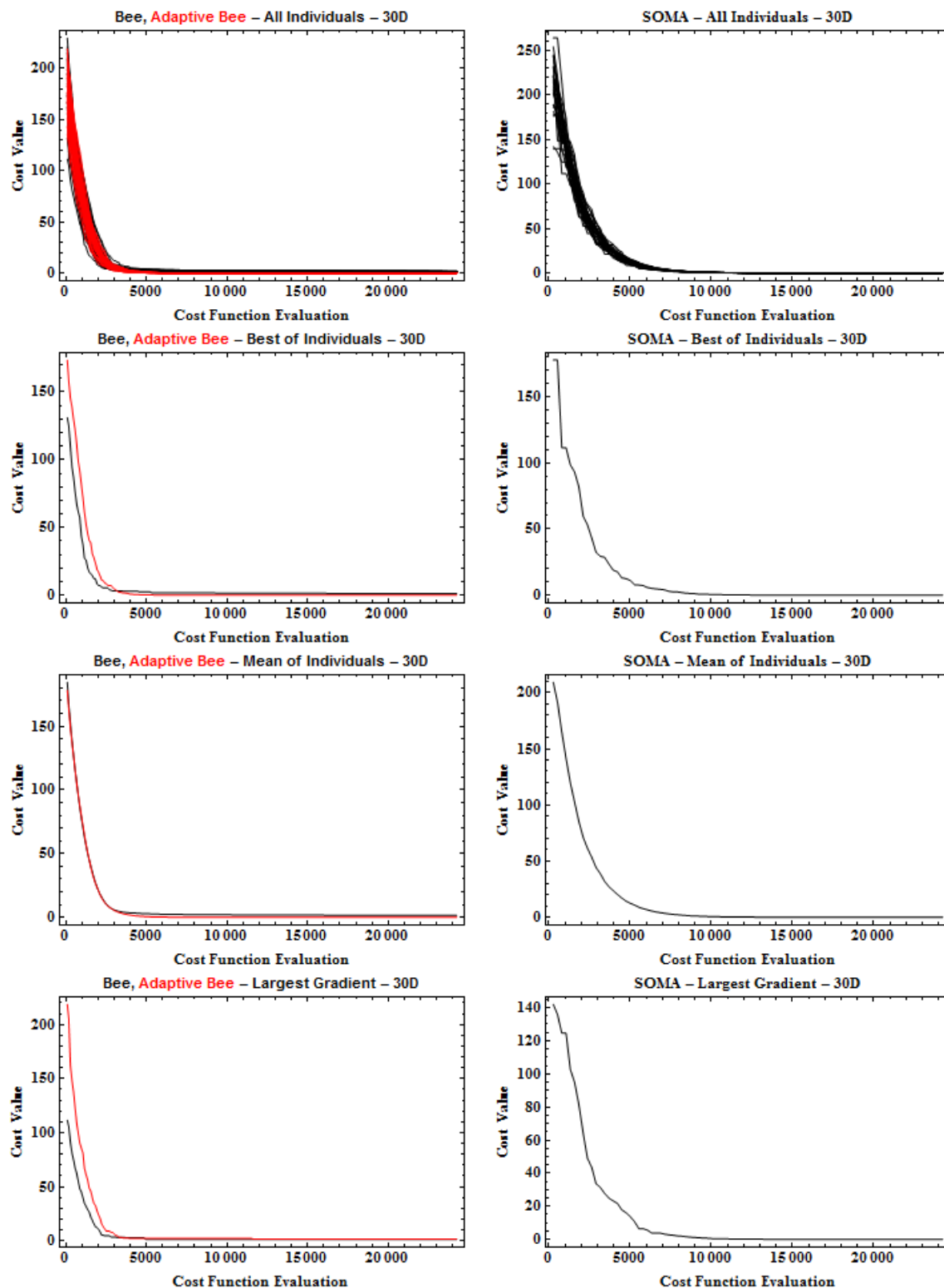


Obrázek 38 - Průběh výsledků pro Shifted 1st De Jong - 5D

9.5.2 30. dimenze

Shifted 1st De Jong - 30D

$n=30$, $m=3$, $e=1$, $n_2=30$, $n_1=5$, $ngh=0.5$, $iter=250$
 PopSize=30, Migrations=92, Step=0.22, PathLength=2.



Obrázek 39 - Průběh výsledků pro Shifted 1st De Jong - 30D

9.6 Shifted Rastrigin

Bylo dosaženo stejného výsledku jako u testování funkce Rastrigin (viz Tabulka 8). Obě BA uvážnou v lokálním extrému. Algoritmus SOMA s mnohem menším CFE je schopna nalézt výborné řešení jak lze vidět na Obr. 40 a 41.

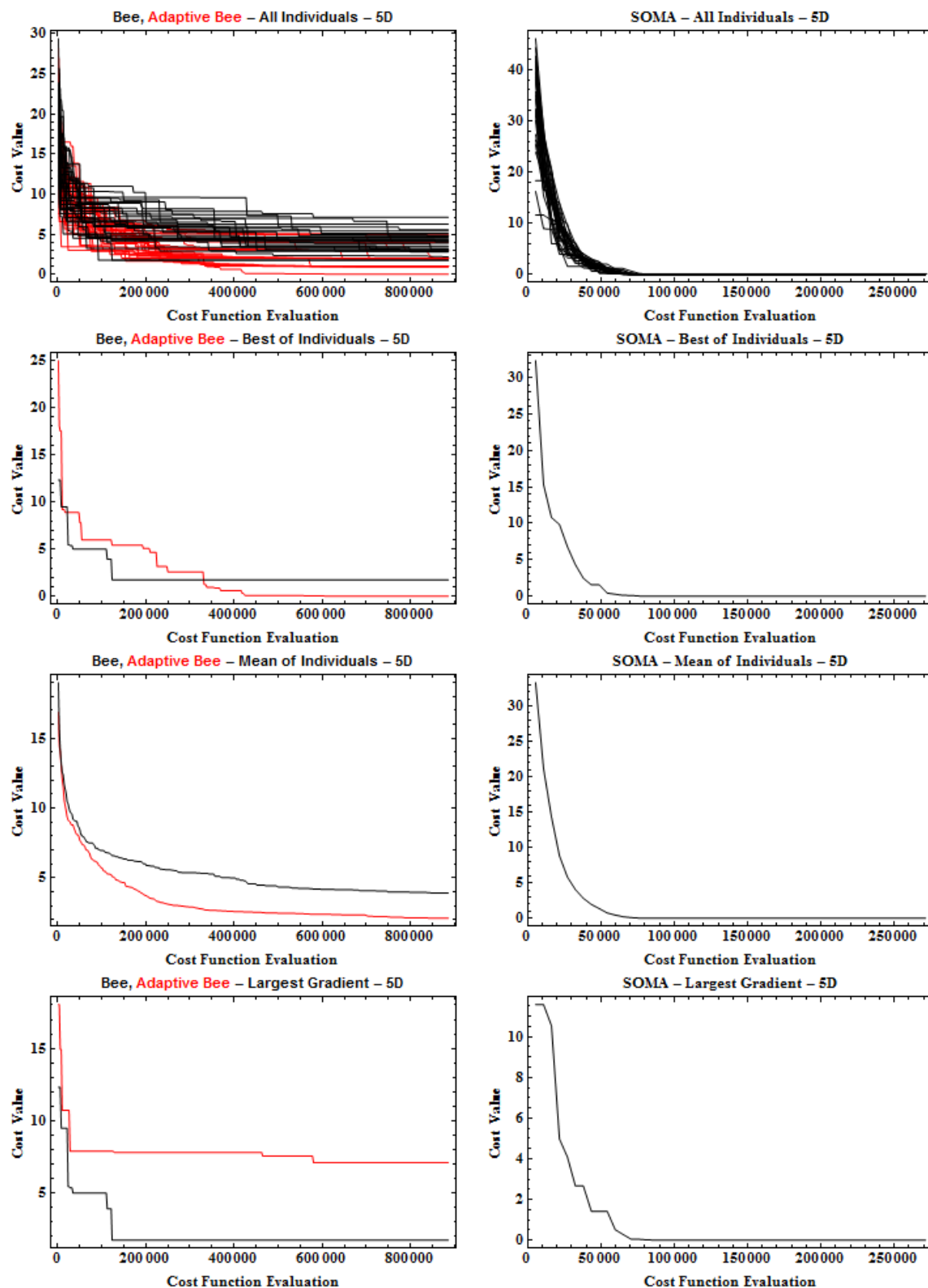
Shifted Rastrigin						
	Bee Algorithm					
Dimenze	2	5	10	20	30	50
Nejlepší CV	0,0179	1,7521	18,4781	91,7573	176,8208	377,2934
CFE	2 460	885 600	2 388 000	950 000	2 320 000	4 800 000
Teoretický extrém	0	0	0	0	0	0
Rozdíl	0,0179	1,7521	18,4781	91,7573	176,8208	377,2934
Průměr	0,8485	3,9008	27,5072	107,2268	200,3849	417,6648
Průměr CV v 300 000 CFE	-	5,3673	34,8961	119,9232	222,4229	471,9938
Průměr CV v 700 000 CFE	-	4,0733	31,3126	111,1539	210,3558	455,2767
% úspěšnost	99,9777	99,1294	95,4092	88,6016	85,3565	81,2525
	SOMA					
Dimenze	2	5	10	20	30	50
Nejlepší CV	1,0613E-06	0	0	0	1,2078E-09	3,8783E-08
CFE	3 455	271 364	325 636	542 727	542 727	723 636
Teoretický extrém	0	0	0	0	0	0
Rozdíl	1,0613E-06	0	0	0	1,2078E-09	3,8783E-08
Průměr	0,0765	0	0	0	0	0,2322
Průměr CV v 300 000 CFE	-	-	0	0	0,0172	8,0627
Průměr CV v 700 000 CFE	-	-	-	-	-	0,2322
% úspěšnost	100	100	100	100	100	100
	Adaptive Bee Algorithm					
Dimenze/ngh	2/1	5/20	10/30	20/150	30/250	50/500
Nejlepší CV	4,5304E-08	1,7629E-04	4,9756	13,9300	45,8213	99,5539
CFE	2 460	885 600	2 388 000	950 000	2 320 000	4 800 000
Teoretický extrém	0	0	0	0	0	0
Rozdíl	4,5304E-08	1,7629E-04	4,9756	13,9300	45,8213	99,5539
Průměr	0,7782	2,0903	13,2999	30,6535	57,6371	127,6216
Průměr CV v 300 000 CFE	-	2,8991	25,0313	52,6942	203,6377	464,2922
Průměr CV v 700 000 CFE	-	2,2932	14,9714	30,8296	117,5637	431,7583
% úspěšnost	100	99,9999	98,7638	98,2696	96,2053	95,0532

Tabulka 8 – Vyhodnocení funkce Shifted Rastrigin

9.6.1 5. dimenze

Shifted Rastrigin - 5D

$n=1000$, $m=6$, $e=2$, $n_2=100$, $n_1=5$, $ngh=0.5$, $iter=400$
 PopSize=200, Migrations=50, Step=0.11, PathLength=3.

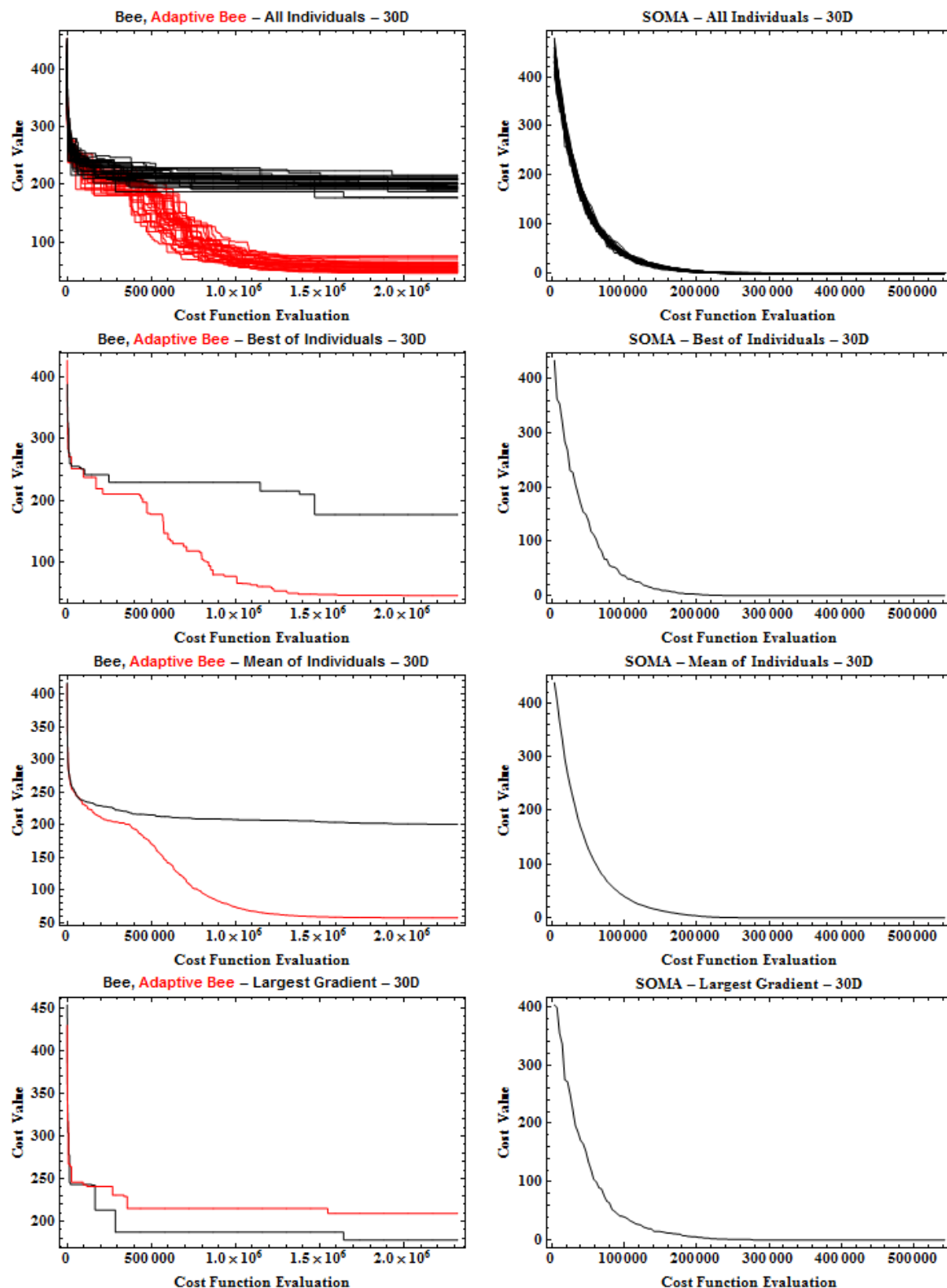


Obrázek 40 - Průběh výsledků pro Shifted Rastrigin - 5D

9.6.2 30. dimenze

Shifted Rastrigin - 30D

$n=30$, $m=20$, $e=10$, $n2=20$, $n1=5$, $ngh=2$, $iter=8000$
 PopSize=200, Migrations=150, Step=0.11, PathLength=2.



Obrázek 41 - Průběh výsledků pro Shifted Rastrigin - 30D

ZÁVĚR

Tato diplomová práce měla za úkol vytvořit a otestovat včelí algoritmus a porovnat jej s jiným evolučním algoritmem. Jako druhý evoluční algoritmus byl vybrán SOMA.

Oba dva algoritmy byly testovány na šesti testovacích funkcích. U každé testovací funkce byla vytvořena přehledná tabulka, která ukazuje, jak si daný algoritmus ve výpočtech vedl. Co se týče vykreslení průběhů, byla zvolena 5. dimenze jako zástupce nižších dimenzí a 30. dimenze jako zástupce dimenzí vyšších. Zbylé průběhy jsou k dostání na přiloženém CD.

Lepší srovnání algoritmů zaručuje vynesení počtu ohodnocení účelových funkcí (dále jen CFE) na osu x a hodnoty účelové funkce na osu y. Pro každý algoritmus se CFE počítá jiným způsobem, viz kapitola 7.1 Charakteristika testu.

Z počátku se algoritmy pro lepší porovnání testovaly s podobným CFE. Ovšem u multimodálních funkcí vyšších dimenzí už SOMA nepotřeboval takové množství CFE jako včelí algoritmus. Proto byl u tohoto algoritmu zvolen mnohem menší CFE, a jak lze vidět v tabulkách, bylo toto množství dostačující.

Včelí algoritmus i SOMA byly spuštěny 30x na každou testovací funkci. Opakování výpočtu se provádí kvůli stochastickému prvku v každém z obou algoritmů, kde se tímto způsobem zaručí nalezení nejlepšího výsledku a eliminuje se jev náhody.

Ke klasickému včelímu algoritmu, který je popsán v kapitole 5.4 Včelí algoritmus, byl ještě přidán tzv. adaptivní včelí algoritmus. Adaptivní proto, že za běhu dokáže měnit velikost okolí v závislosti na průběhu výpočtu. Respektive, pokud se nalezené řešení dlouhodobě nemění a stále se nachází řešení horší, tak se okolí zmenší, což zajistí ve většině případů nalezení lepšího řešení. Testy ukázaly, že tato modifikace je vhodnější a ve všech případech byla nalezena lepší řešení. Ovšem při rychlém zmenšování prohledávaného okolí hrozí riziko uváznutí v lokálním extrému. S rostoucím počtem generací je proto vhodné měnit okolí (parametr *ngh*) až poté, co se budou ve výsledcích vyskytovat stejná řešení vícekrát po sobě. Testování rovněž ukázalo, že z hlediska nastavení parametrů včelího algoritmu je výhodnější nastavit více generací a méně jedinců. Důkladné nastavení jednotlivých parametrů hraje u obou algoritmů klíčovou roli.

Charakteristika srovnávacích grafů je následující: v levé části je umístěn včelí algoritmus (černá barva) a adaptivní včelí algoritmus (červená barva) jenž jsou vykresleny do jednoho

grafu, a v pravé části SOMA. Samotné grafy se skládají ze 4 částí. Těmi jsou vykreslení všech 30 opakování, nejlepší ze všech těchto opakování, průměr ze všech opakování a největší spád.

Po otestování algoritmů na všech testovacích funkcích byly zjištěny následující skutečnosti. Z výsledků bylo patrné, že pro unimodální funkce nebo funkce s málo extrémy, což v tomto případě představuje 1st De Jong, shifted 1st De Jong a 2nd De Jong, vykazuje adaptivní včelí algoritmus mírně lepší výsledky než SOMA. Tento rozdíl je však zanedbatelný. Naproti tomu u multimodálních funkcí ve vyšších dimenzích je rozdíl v řešení mnohem více znatelný, a to ve prospěch algoritmu SOMA. Zatímco SOMA je s dostatečným počtem migrací a jedinců schopna nalézt globální extrém, řešení včelího algoritmu v každém z 30 případů uvázlo v lokálním extrému a to i přes skutečnost, že CFE byl oproti SOMA několikanásobně vyšší. Tento jev je dán skutečností, že SOMA pracuje po skocích a prohledává daný prostor v určitých intervalech, kde je prvek náhody pouze směr cesty. Včelí algoritmus využívá náhody v okruhu hledání prvků, kde náhodně generuje nová řešení. Tudíž není zajištěno prohledávání veškerého možného prostoru. Testováním bylo zjištěno, že při generování nového řešení v tomto okolí byla v mnoha případech nalezena řešení horší než již nalezené nejlepší řešení daného okolí. Algoritmus ve většině případů stál na místě.

Testy neprokázaly, že by posuny (shifted funkce) měly na schopnost algoritmu nějaký vliv.

Závěrem lze říci, že včelí algoritmus pracuje uspokojivě na funkcích s minimálním počtem extrémů, zatímco SOMA zvládá bez problémů všechny případy.

ZÁVĚR V ANGLIČTINĚ

Within this thesis, Bee algorithm was implemented, tested and compared it with other evolutionary algorithm. For this comparison, SOMA was chosen.

Both algorithms were tested on six test functions. For each test function a transparent chart was created that shows how good the algorithm was in calculations. Regarding the charts, 5th dimension was chosen as a representative of the lower dimensions and 30th dimension as a representative of higher dimensions. The rest is enclosed on CD.

For better comparisons of algorithms, cost function evaluation (CFE) is projected on the x-axis and cost value on the y-axis. For each algorithm CFE is calculated differently, see chapter 7.1 Characteristics of the test.

From the beginning, algorithms were tested with similar CFE. However, for multimodal functions with higher dimensions SOMA did not need so many CFE as Bee algorithm. Therefore a much smaller CFE was chosen for SOMA and as it can be seen in the tables, it was sufficient.

Bee algorithm and SOMA were run 30 times for each test function. Repeating the calculation is performed because of stochastic element in each algorithm. This way ensures finding the best results and eliminates the phenomenon of coincidence.

As a complement to the classical Bee algorithm, which is described in section 5.4 Bee algorithm, Adaptive Bee algorithm was furthermore implemented. Notation “Adaptive” is because during runtime, it can change the size of searched area depending on the progress of the calculation. Respectively, if the found solutions remain unchanged for a long time and algorithm keeps finding worse solutions, the searched area is reduced. In most cases, a better solution is found. Tests have shown that this modification is appropriate and in all cases a better solution was found. However, the rapid reduction of the searched area is risky because of threat of being stuck in a local extreme. Therefore an appropriate setting is while increasing number of iterations adjusts the size of the searched area (ngh) only when same solutions occur multiple times. Testing also showed that in terms of parameter settings, multiple generations and fewer individuals are better. Thorough setting of parameters is fundamental and plays a key role for both algorithms.

Characteristics of the comparative charts is following - Bee algorithm (black) and Adaptive Bee algorithm (red) are together in one chart on the left side and SOMA is on the right. The actual graphs consist of four parts - all 30 repetitions, the best progress from those repetitions, mean from those repetitions and the largest gradient.

After testing both algorithms on all test functions, the conclusion is following. From the results it was clear that Adaptive Bee algorithm has slightly better results than SOMA on unimodal functions or function with a few extremes (which is the case of 1st De Jong, shifted 1st De Jong and 2nd De Jong). However, this difference was insignificant. In contrast, on multimodal functions with higher dimensions there was much more noticeable difference in favor of SOMA. SOMA with sufficient amount of migrations and individuals was capable of finding global extreme, while Bee algorithm stucked in local extreme in all cases despite the fact that compared to SOMA, CFE was several times higher. This phenomenon is due to the fact that SOMA is using steps and searches the area at certain intervals where the random element is the direction of travel. Bee algorithm uses random elements in the surroundings of individuals where randomly generates new solutions. Because of this, searching in all possible areas is not ensured. Tests have discovered that newly generated solutions from this area were in many cases worse than the already found best solution. So the algorithm stood still in most cases.

Tests have shown that shifts (shifted test functions) did not affect the algorithm in any way.

In summary, Bee algorithm works satisfactorily on functions with minimum number of extremes while SOMA handles all tested cases with no problems.

SEZNAM POUŽITÉ LITERATURY

- [1] OPLATKOVÁ, Zuzana. *Neuronové sítě: Úvod do umělé inteligence* [online]. Zlín, 2012 [cit. 2013-04-24].
- [2] POPELKA, Ondřej a Michael ŠTENCL. *Možnosti aplikace metod umělé inteligence v ekonomii* [online]. Brno, 2008 [cit. 2013-04-24]. Dostupné z: http://www.mendelu.cz/dok_server/slozka.pl?id=39500;download=40993.
- [3] MEHROTRA, Kishan, Chilukuri K. MOHAN a Sanjay RANKA. *Elements of Artificial Neural Networks*. Cambridge, Mass.: MIT Press, c1997, xiv, 344 s. ISBN 978-0-262-13328-9.
- [4] ZELINKA, Ivan. *Umělá inteligence I* [online]. Zlín, 1997 [cit. 2013-04-24].
- [5] JIRSÍK, Václav a Petr HRÁČEK. *Neuronové sítě, expertní systémy a rozpoznávání řeči* [online]. Brno: Vysoké učení technické, 2002 [cit. 2013-04-24].
- [6] OPLATKOVÁ, Zuzana. *Neuronové sítě: Principy činnosti* [online]. Zlín, 2012 [cit. 2013-04-24].
- [7] IMADA, Akira. *An Introduction to Soft Computing: Neural Networks, Evolutionary Computations and Fuzzy Logic*. Brest State Technical University, 2003, 11 s. Dostupné z: <http://neuro.bstu.by/ai/soft-computing.pdf>.
- [8] ZELINKA, Ivan, Zuzana OPLATKOVÁ a Roman ŠENKERŮ. *Aplikace umělé inteligence*. Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2010, 151 s. ISBN 978-80-7318-898-6.
- [9] TVRDÍK, Josef. *Evoluční algoritmy* [online]. Ostravská univerzita, 2004 [cit. 2013-04-25]. Dostupné z: http://prf.osu.cz/doktorske_studium/dokumenty/Evolutionary_Algorithms.pdf.
- [10] RUTKOWSKI, Leszek a Janusz KACPRZYK. *Neural Networks and Soft Computing: Proceedings of the Sixth International Conference on Neural Networks and Soft Computing, Zakpane, Poland, June 11-15, 2002* [online]. Heidelberg: Physica-Verlag, 2003, 914 s. [cit. 2013-04-25]. ISBN 37-908-0005-8. Dostupné z: <http://books.google.cz/books?id=O74IjSEUucsC>.

- [11] Soft Computing. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 17 April 2013 [cit. 2013-04-25]. Dostupné z: http://en.wikipedia.org/wiki/Soft_computing.
- [12] ZBORIL, František V. *Soft computing: Pojem soft computing* [online]. Vysoké učení technické Brno, 2011 [cit. 2013-04-25]. Dostupné z: https://www.fit.vutbr.cz/study/courses/SFC/private/11sfc_1.pdf.
- [13] TRENZ, Oldřich. *Fuzzy množiny, Fuzzy logika, Fuzzy regulace* [online]. Mendelova Univerzita Brno, 2009 [cit. 2013-04-25]. Dostupné z: https://akela.mendelu.cz/~xkosicek/UI/VUI_P8_LS_09_Fuzzy.ppt.
- [14] SERAFÍN, Čestmír. *Fuzzy logika* [online]. Univerzita Palackého Olomouc [cit. 2013-04-25]. Dostupné z: <http://www.kteiv.upol.cz/uploads/soubory/serafin/mechatronika/10-Fuzzy%20logika.ppt>.
- [15] 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.
- [16] CHALUPNÍK, Vitalij. *Biologické algoritmy - Evoluční algoritmy* [online]. 4. 4. 2012 [cit. 2013-04-25]. Dostupné z: <http://www.root.cz/clanky/biologicke-algoritmy-1-evolucni-algoritmy>.
- [17] ZELINKA, Ivan. *Biologicky inspirované výpočty* [online]. VŠB TU Ostrava, 2008 [cit. 2013-04-25]. Dostupné z: <http://arg.vsb.cz/data/Vyuka/BIV-AUI.pdf>.
- [18] Metaheuristic. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 21 April 2013 [cit. 2013-04-25]. Dostupné z: <http://en.wikipedia.org/wiki/Metaheuristic>.
- [19] OLIVKA, Petr. *Genetické algoritmy* [online]. VŠB TU Ostrava [cit. 2013-04-25]. Dostupné z: <http://poli.cs.vsb.cz/edu/isy/down/ga.pdf>.
- [20] ONWUBOLU, Godfrey C a B BABU. *New optimization techniques in engineering* [online]. Berlin: Springer, 2004, xxii, 712 s. [cit. 2013-04-25]. ISBN 35-402-0167-X. Dostupné z: <http://www.google.cz/books?id=YRsq-Pz0kAAC>.

- [21] LIU, Yang a Kevin M. PASSINO. *Swarm Intelligence* [online]. Ohio State University, March 30, 2000 [cit. 2013-04-25]. Dostupné z: <http://www2.ece.ohio-state.edu/~passino/swarms.pdf>.
- [22] MILLER, Peter. *National Geographic: Swarm Theory* [online]. July 2007 [cit. 2013-04-25]. Dostupné z: <http://ngm.nationalgeographic.com/2007/07/swarms/miller-text>.
- [23] BONABEAU, Eric. *Swarm Intelligence: a presentation* [online]. 2000 [cit. 2013-04-27]. Dostupné z: <http://faculty.washington.edu/paymana/swarm/bonabeau00.pdf>.
- [24] GALDOVÁ, Lucie. *Particle Swarm algoritmus v prostředí Mathematica* [online]. Zlín, 2007 [cit. 2013-04-27]. Dostupné z: https://dspace.k.utb.cz/bitstream/handle/10563/4807/galdov%E1_2007_dp.pdf?sequence=1. Diplomová práce. Univerzita Tomáše Bati ve Zlíně.
- [25] XIAHUI, Hu. About PSO: Introducion. *Particle Swarm Optimization* [online]. 2006 [cit. 2013-04-27]. Dostupné z: <http://www.swarmintelligence.org/index.php>.
- [26] EBERHART, Russell C. *Particle Swarm Optimization* [online]. 2001 [cit. 2013-04-27]. Dostupné z: http://faculty.washington.edu/paymana/swarm/krink_01.pdf.
- [27] HOŠEK, Pavel. O včelích tanečcích, orientaci a robotech. *Vesmír* [online]. 1995 [cit. 2013-04-27]. Dostupné z: <http://www.vesmir.cz/clanek/o-vcelich-taneccich-orientaci-a-robotech>.
- [28] Význam včel. *n-vcelari* [online]. [cit. 2013-04-27]. Dostupné z: <http://www.n-vcelari.sk/sal/VCELY1.html>.
- [29] Nový objev: K čemu slouží včelí tance?. *21stoleti* [online]. 19.3.2007 [cit. 2013-04-27]. Dostupné z: <http://21stoleti.cz/blog/2007/03/19/novy-objev-k-cemu-slouzi-vceli-tance/>.
- [30] BAYKASOGLU, Adil, Lale OZBAKIR a Pinar TAPKAN. *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem* [online]. Trukey [cit. 2013-04-27]. Dostupné z: <http://natcomp.liacs.nl/SWI/papers/artificial.bee.colony.algorithm/abc.algorithm.and.its.application.to.generalized.assignment.problem.pdf>.

- [31] KARABOGA, Dervis, Bahriye AKAY. *A comparative study of Artificial Bee Colony algorithm* [online]. Trukey, 2009[cit. 2013-04-27]. Dostupné z: <http://chern.ie.nthu.edu.tw/gen/comparative-study.pdf>.
- [32] LARIK, Asma Sanam. *Artificial Bee Colony Algorithm* [online]. 2011[cit. 2013-04-28]. Dostupné z: <http://cse659ci.wikispaces.com/file/view/Artificial+Bee+Colony+Algorithm.pptx>.
- [33] KARABOGA, Dervis. Artificial bee colony algorithm. *Scholarpedia* [online]. 2010, 21 October 2011 [cit. 2013-04-28]. Dostupné z: http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm.
- [34] OZBAKIR, Lale, Adil BAYKASOGLU a Pinar TAPKAN. Bees algorithm for generalized assignment problem. *Science direct* [online]. February 2010 [cit. 2013-04-28]. Dostupné z: <http://www.sciencedirect.com/science/article/pii/S0096300309010078?np=y>.
- [35] TEODOROVIĆ, Dušan. *Bee Colony Optimization (BCO)* [online]. 2011[cit. 2013-04-28]. Dostupné z: <http://www.sf.bg.ac.rs/downloads/katedre/oi/1.BCO-Book-Chapter.pdf>.
- [36] Mathematica [online]. 2011 [cit. 2013-04-28]. Wolfram Mathematica. Dostupné z: http://www.mathematica.cz/produkty.php?p_mathematica.
- [37] Reference.wolfram [online]. 2011 [cit. 2013-04-28]. Wolfram Mathematica. Dostupné z: <http://reference.wolfram.com/mathematica/guide/Mathematica.html>.
- [38] Wolfram [online]. 2011 [cit. 2013-04-28]. Solutions for Higher Education. Dostupné z: <http://www.wolfram.com/solutions/education/higher-education/>.
- [39] *Basic functions* [online]. 2007-05-09[cit. 2013-04-28]. Dostupné z: <http://www.it.lut.fi/ip/evo/functions/node1.html>.
- [40] *Non-linear Continuous Multi-Extremal Optimization* [online]. 2004-12-17[cit. 2013-04-28]. Dostupné z: <http://www.maths.uq.edu.au/CEToolBox/node3.html>.
- [41] *Acclaim Clipart* [online]. [cit. 2013-04-28]. Dostupné z: <http://www.acclaimclipart.com/>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
BA	Bee Algorithm
BCO	Bee Colony Optimization
BS	Bee Systém
CFE	Cost Function Evaluation
CV	Cost Value
HBMO	Honey-Bees Mating Optimization
LVQ	Learning Vector Quantization
MBO	Marriage in Honey-Bees Optimization Based
PSO	Particle Swarm Optimization
QBE	Queen-Bee Evolution Algorithm
SOMA	Samo-organizační migrační algoritmus
TSP	Travelling Salesman Problem
VBA	Virtual Bee Algorithm

SEZNAM OBRÁZKŮ

Obrázek 1 - Umělý neuron.....	14
Obrázek 2 - Zobrazení testovací funkce 1st De Jong ve 3D a 2D.....	16
Obrázek 3 - Obecný cyklus evolučního algoritmu.....	19
Obrázek 4 - Chování SOMA – funkce Schwefel.....	23
Obrázek 5 - Cesta k potravě bez mravenců	28
Obrázek 6 - Optimalizovaná cesta mravenci	28
Obrázek 7 - Chování dělnic při shánění potravy.....	32
Obrázek 8 – Ukázka přerozdělení pravděpodobností.....	37
Obrázek 9 – Základní prvky ABC	38
Obrázek 10 - Inicializace počáteční populace ABC	39
Obrázek 11 - Výpočet okolí a výběr nejlepších ABC.....	39
Obrázek 12 - Rozdělení včel podle pravděpodobnosti ABC.....	40
Obrázek 13 - Životní cyklus algoritmu	42
Obrázek 14 - Inicializace populace BA.....	43
Obrázek 15 - Výběr (m) nejlepších včel BA.....	43
Obrázek 16 - Zobrazení okolí pro (m) nejlepších BA.....	44
Obrázek 17 – Vyslání více včel do prohledávaného okolí BA	44
Obrázek 18 - Výběr nové nejlepší včely z každého okolí	45
Obrázek 19 - Generování nových jedinců	45
Obrázek 20 - Problém obchodního cestujícího.....	47
Obrázek 21 – Zobrazení funkce 1st De Jong	51
Obrázek 22 – Zobrazení funkce 2nd De Jong	52
Obrázek 23 - Zobrazení funkce Schwefel	53
Obrázek 24 - Zobrazení funkce Rastrigin	54
Obrázek 25 - Zobrazení funkce Shifted 1st De Jong	55
Obrázek 26 - Zobrazení funkce Shifted Rastrigin	56
Obrázek 27 – Bee Algorithm v Mathematice.....	57
Obrázek 28 – Použití BA v Mathematice	58
Obrázek 29 - Výpočet úspěšnosti.....	60
Obrázek 30 - Průběh výsledků pro 1st De Jong - 5D.....	62
Obrázek 31 - Průběh výsledků pro 1st De Jong - 30D.....	63

Obrázek 32 - Průběh výsledků pro 2nd De Jong - 5D.....	65
Obrázek 33 - Průběh výsledků pro 2nd De Jong - 30D.....	66
Obrázek 34 - Průběh výsledků pro Schwefel - 5D	68
Obrázek 35 - Průběh výsledků pro Schwefel - 30D	69
Obrázek 36 - Průběh výsledků pro Rastrigin - 5D	71
Obrázek 37 - Průběh výsledků pro Rastrigin - 30D	72
Obrázek 38 - Průběh výsledků pro Shifted 1st De Jong - 5D	74
Obrázek 39 - Průběh výsledků pro Shifted 1st De Jong - 30D	75
Obrázek 40 - Průběh výsledků pro Shifted Rastrigin - 5D.....	77
Obrázek 41 - Průběh výsledků pro Shifted Rastrigin - 30D.....	78

SEZNAM TABULEK

Tabulka 1 - Parametry algoritmu SOMA.....	23
Tabulka 2 - Algoritmy založené na principu chování včel	34
Tabulka 3 – Vyhodnocení funkce 1st De Jong.....	61
Tabulka 4 – Vyhodnocení funkce 2nd De Jong.....	64
Tabulka 5 - Vyhodnocení funkce Schwefel.....	67
Tabulka 6 - Vyhodnocení funkce Rastrigin.....	70
Tabulka 7 - Vyhodnocení funkce Shifted 1st De Jong	73
Tabulka 8 – Vyhodnocení funkce Shifted Rastrigin	76

SEZNAM PŘÍLOH

- PŘÍLOHA P I: Zdrojový kód (Mathematica) – včelí algoritmus, SOMA
- PŘÍLOHA P II: Průběhy výpočtů (.nb, .png)
- PŘÍLOHA P III: Tabulky výpočtů (.nb, .xls)
- PŘÍLOHA P IV: Výsledky testování (.txt)