Vývoj nových adaptivních strategií algoritmu SOMA

Bc. Marcela Matušíková

Diplomová práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Marcela Matušíková**
Osobní číslo:	**A18268**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Vývoj nových adaptivních strategií algoritmu SOMA**
Téma práce anglicky:	**The Development of New Adaptive Strategies of the SOMA Algorithm**

Zásady pro vypracování

1. Vypracujte literární rešerši.
2. Popište principy a varianty optimalizačního algoritmu SOMA.
3. Analyzujte adaptivní mechanismy v moderních variantách algoritmu SOMA.
4. Navrhněte vlastní adaptivní strategie pro řízení populační dynamiky a parametry algoritmu ve zvoleném programovacím prostředí.
5. Otestujte vytvořené varianty na vybraných testovacích funkcích.
6. Proveďte statistické srovnání s existujícími verzemi SOMA algoritmu a výsledky přehledně graficky a tabulkově zobrazte.

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

Seznam doporučené literatury:

1. ZELINKA, Ivan. *Evoluční výpočetní techniky: principy a aplikace*. Praha: BEN – technická literatura, 2009, 534 s. ISBN 9788073002183.
2. KACPRZYK, Janusz a Witold PEDRYCZ, ed. *Springer handbook of computational intelligence*. Dordrecht: Springer, 2015, lvi, 1633 s. ISBN 9783662435045.
3. ZELINKA, Ivan, Václav SNÁŠEL a Ajith ABRAHAM, ed. *Handbook of optimization: from classical to modern approach*. Berlin: Springer, c2013, xii, 1100 s. Intelligent systems reference library. ISBN 9783642305030.
4. DAVENDRA, Donald a Ivan ZELINKA, ed. *Self-organizing migrating algorithm: methodology and implementation*. [Cham]: Springer, [2016], xviii, 289 s. Studies in computational intelligence. ISBN 9783319281599. Dostupné také z: http://www.loc.gov/catdir/enhancements/fy1608/2015958861-d.html .
5. DIEP, Quoc Bao. Self-Organizing Migrating Algorithm *Team To Team Adaptive–SOMA T3A*. 2019 IEEE Congress on Evolutionary Computation, IEEE, 2019, 1182-1187.
6. MELOUN, Milan a Jiří MILITKÝ. *Statistická analýza experimentálních dat*. Vyd. 2., upr. a rozš. Praha: Academia, 2004, 953 s. ISBN 8020012540.

Vedoucí diplomové práce: **doc. Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **15. ledna 2021**
Termín odevzdání diplomové práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne

Marcela Matušíková v.r.
podpis studenta

ABSTRAKT

Diplomová práca sa zaoberá algoritmom založeným na inteligencii roja a to konkrétne algoritmom SOMA. Cieľom diplomovej práce bol vývoj nových adaptívnych stratégii pre spomínaný algoritmus. Prvá časť práce sa venuje evolučným výpočetným technikám. Následne sa detailne venuje algoritmu SOMA a jej moderným variantám. Vývojom, následnou implementáciou a mnohými experimentami boli získané cenné poznatky, ktoré sú zdokumentované v práci a môžu napomôcť ďalším ľudom, ktorí sa budú venovať tomuto algoritmu, či budú chcieť využiť algoritmus alebo jeho modifikácie pre reálne optimalizačné problémy. Získané výsledky diplomovej práce sú zosumarizované v praktickej časti práce.

Kľúčové slová: SOMA, moderné varianty algoritmu SOMA, adaptívne stratégie, optimalizácia

ABSTRACT

The master's thesis deals with an algorithm based on swarm intelligence, specifically the SOMA algorithm. The aim of the master's thesis was the development of new adaptive strategies for the mentioned algorithm. The first part of the thesis deals with evolutionary computing techniques. Subsequently, it deals in detail with the SOMA algorithm and its modern variants. Throughout development, subsequent implementation and many experiments the valuable knowledge which is documented in the thesis has been gained. This knowledge can also help people who will focus on this algorithm, whether they want to use the algorithm or its modifications for real optimization problems. The obtained results of the diploma thesis are summarized in the practical part.

Keywords: SOMA, modern variants of SOMA algorithm, adaptive strategies, optimization

POĎAKOVANIE A MOTTO

Touto cestou by som predovšetkým veľmi rada poďakovala môjmu konzultantovi Ing. Tomášovi Kadavému a taktiež srdečná vďaka patrí aj vedúcemu mojej diplomovej práce, doc. Ing. Romanovi Šenkeříkovi, Ph.D., za ich odborné vedenie a užitočné rady, podporu, zhovievavosť a v neposlednom rade ochotu, pevné nervy či poskytnutý čas, ktorý je pre nás všetkých najcennejší.

Tvoj čas je obmedzený. Preto ho nestrácaj tým, že budeš

žiť život niekoho iného.

Steve Jobs

Ak si myslíš, že to zvládneš, tak to zvládneš.

Ak si myslíš, že nie, máš pravdu.

Mary Kay Ash

OBSAH

ÚVOD

Algoritmy založené na inteligencii roja a ich rôzne modifikácie sa v priebehu rokov neustále vyvíjajú a tešia sa veľkej pozornosti. Výnimkou nie je ani algoritmus SOMA (celým znením Self-Organizing Migrating Algorithm, slovensky Samoorganizujúci sa migračný algoritmus), ktorému bola venovaná táto diplomová práca. Tento typ algoritmov sa využíva na riešenie optimalizačných problémov. Riešiť tieto problémy sa dá tromi základnými spôsobmi. Prvým je možnosť exaktného riešenia. Veľkou výhodou je síce získanie optimálneho riešenia, avšak vo väčšine prípadov je to na úkor veľkej výpočetnej doby. Druhým spôsobom je riešiť optimalizačné problémy úplne náhodne, napríklad pomocou algoritmu Random Search. Posledný spôsob je pomocou metaheuristík, pod ktoré spadá aj SOMA.

Cieľom tejto práce bolo preskúmať oblasť okolo algoritmu SOMA a všeobecne oblasť používaných adaptívnych techník v metaheuristikách, preskúmať existujúce adaptívne stratégie pre algoritmus SOMA a v konečnom dôsledku navrhnúť aj vlastné adaptívne stratégie. Následne získané výsledky bolo treba vhodne vyhodnotiť.

V teoretickej časti sú stručne popísané základné princípy evolučných výpočetných techník. Je rozobratá história, centrálna dogma, obecné princípy a základné pojmy. Priestor je venovaný typickým zástupcom evolučných výpočetných techník ako aj cieľom a ich aplikáciám v reálnom svete. Ďalej práca detailne popisuje princípy a jednotlivé základné varianty algoritmu SOMA. Samostatná kapitola je zameraná na moderné varianty algoritmu SOMA, kde sa popisujú vzniknuté a publikované modifikácie za posledný rok. Práca sa tiež zameriava na testovanie a spôsoby porovnávania výkonu algoritmov.

Hlavným cieľom praktickej časti diplomovej práce bolo navrhnúť originálne modifikácie algoritmu SOMA, konkrétne návrh vlastných adaptívnych stratégií pre riadenie populačnej dynamiky a parametrov algoritmu. Pred tým než bolo možné navrhnúť originálne adaptívne stratégie, bolo potrebné preskúmať už existujúce adaptívne stratégie v moderných verziách SOMA a zistiť ich vplyv na daný algoritmus. Až po tomto prieskume nasledoval návrh vlastných adaptívnych stratégií. Popisy jednotlivých návrhov sa nachádzajú v kapitole 7. Navrhnuté adaptívne stratégie sa následne implementovali, vykonalo sa množstvo experimentov a následne bolo potrebné zistiť a stanoviť ich vplyv na priemerný výkon algoritmu SOMA. Všetky algoritmy boli testované na benchmarku IEEE CEC 2020, na ktorom bolo možné navzájom porovnať vlastné modifikácie a vybrané už existujúce modifikácie

spoločne so základnými verziami algoritmu SOMA. Všetky výsledky testovania sú prehľadne zobrazené v tabuľkách, grafoch, prílohách a v závere práce celkovo zosumarizované.

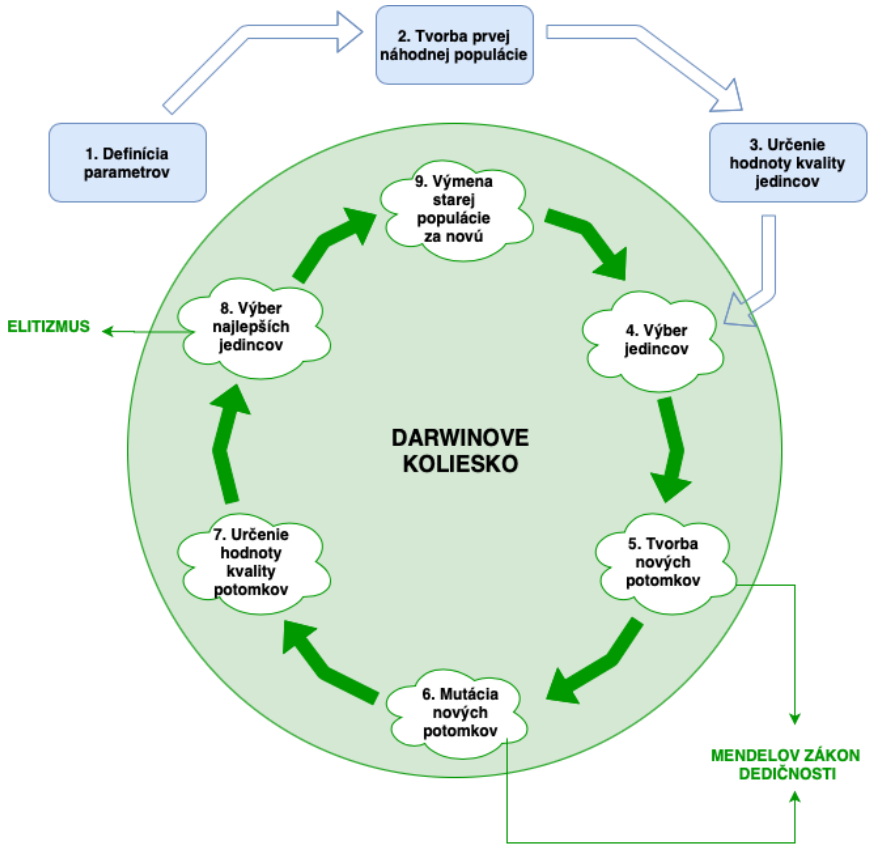
I. TEORETICKÁ ČASŤ

1 EVOLUČNÉ VÝPOČTOVÉ TECHNIKY

Evolučné výpočtové techniky (ďalej už len EVT) sa z hľadiska informatiky radia medzi podobory soft-computingu či umelej inteligencie. Veľkou inšpiráciou EVT sú Darwinove princípy evolúcie, a to konkrétne princíp prirodzeného výberu, ktorý hovorí o tom, že iba tí najlepší (elitní) jedinci by sa mali podieľať na tvorbe nových populácii alebo inak povedané, na tvorbe kandidátnych riešení. Taktiež spolu s Darwinovými princípmi sa EVT opiera aj o Mendelov zákon dedičnosti, ktorý hovorí o krížení a mutácii. [1]

1.1 Centrálna dogma

Ako už aj z názvu evolučné algoritmy (ďalej už len EA) vyplýva, hlavný princíp týchto algoritmov je *evolúcia.* Evolúcia je proces, pri ktorom sa jedinci v priebehu času menia, a to v dôsledku zmien dedičných, fyzických či zmien v správaní. Toto všetko sú zmeny, ktoré umožnia jedincovi lepšie sa prispôsobiť prostrediu, umožnia mu prežiť a mať viac potomkov. [2] EA využívajú mechanizmy inšpirované biologickou evolúciou, ako napríklad reprodukcia, mutácia, rekombinácia a selekcia. Kandidátne riešenia optimalizačného problému hrajú rolu jedincov v populácii a ich vhodnosť určuje kvalitu jednotlivých riešení.



Obrázok 1: Centrálna dogma EVT

Ako už bolo vyššie spomínané, EA sa opierajú o Darwinovu teóriu prirodzeného výberu. Kroky samotného evolučného cyklu sú znázornené na Obrázku 1 a ich popis je nasledujúci:

1. **Definícia parametrov –** na začiatku sa definujú parametre EA, ako sú napríklad riadiace parametre (napr. veľkosť populácie, dimenzia problému,...), ukončovacie podmienky (napr. počet generácií, časová podmienka ukončenia,...), definícia účelovej funkcie a iné.
2. **Tvorba novej náhodnej populácie –** druhým krokom je vygenerovanie náhodnej prvotnej populácie.
3. **Určenie hodnoty kvality jedincov –** v ďalšom kroku sa nová náhodná populácia ohodnotí, a teda určí sa kvalita jedincov buď pomocou účelovej funkcie alebo pomocou hodnoty fitness (modifikovanej účelovej funkcie, obvykle normalizovanej). [1]
4. **Výber jedincov –** výber rodičov, pre tvorbu nových jedincov na základe účelovej hodnoty alebo iných kritérií.
5. **Tvorba nových potomkov –** z vybraných rodičov, ktorí boli vybraní v 4. bode evolučného cyklu, sa vytvárajú noví potomkovia a to pomocou kríženia. Každý algoritmus má svoje špecifické kríženie.
6. **Mutácia nových potomkov –** v 5. bode sa vytvorili noví potomkovia, ktorí následne prejdú mutáciou podľa danej pravdepodobnosti.
7. **Určenie hodnoty kvality potomkov -** rovnako ako na začiatku evolučného cyklu, kedy sa ohodnotila kvalita jedincov z prvotnej náhodnej populácie, tak rovnako sa ohodnotia aj novo vytvorení potomkovia pomocou účelovej funkcie.
8. **Výber najlepších jedincov –** v predposlednom bode sa vyberú tí najlepší jedinci z množiny rodičov, ako aj z množiny potomkov. Počet vybraných najlepších jedincov sa rovná počtu jedincov z prvotnej náhodnej populácie.
9. **Výmena starej populácie za novú –** najlepší jedinci budú tvoriť novú populáciu, a táto nová populácia sa vymení za pôvodnú populáciu. Pôvodná (stará) populácia zaniká.

Evolučný cyklus ďalej pokračuje znova od bodu 4 až pokiaľ nie sú splnené ukončovacie podmienky. Od 4. bodu až po 9. bod prebieha tzv. Darwinove koliesko alebo inak povedané,

Darwinove princípy prirodzeného výberu. V 8. bode je vidieť, ako do cyklu zasahuje *elitizmus* (výber najlepších jedincov).

1.2 Evolučné princípy

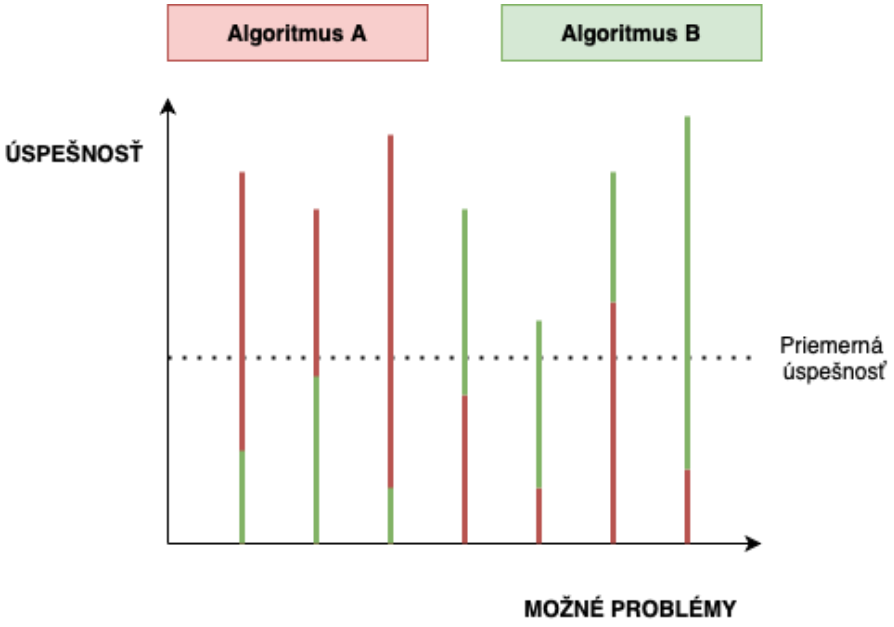
Metódy, ktoré využívajú evolučný cyklus zdieľajú určité evolučné princípy. [1] V najbližšej časti textu sú tie najhlavnejšie popísané:

- **Biologická inšpirácia –** EA sa inšpirujú a snažia napodobovať správanie a postupy biologických systémov.
- **Populačný princíp –** individuálne prehľadávanie jednotlivých jedincov koordinuje s ostatnými jedincami a tým aj s celou populáciou. Vďaka tomuto sa považuje evolučné prehľadávanie za tak úspešné. [1]
- **Opakovanie výpočtov vhodnosti jedincov –** opakovaný výpočet vhodnosti sa vykonáva pre každého jedinca v populácii. Vďaka tomu sa vyhľadávanie riadi tým, že preferuje alebo diskriminuje jedinca s rôznou vhodnosťou.
- **Opakované generačné vyhľadávanie –** generačné vyhľadávanie napomáha tomu, aby sa akumulovali jedinci s vysokou vhodnosťou, inak povedané dochádza ku akumulácii riešení s vysokou kvalitou.
- **Stochastické a deterministické princípy –** v behu EA sa nachádzajú stochastické prvky (napríklad mutácia či náhodné vygenerovanie počiatočnej populácie) rovnako ako aj deterministické prvky.
- **Koordinácia medzi jedincami –** dochádza k určitej koordinácii (komunikácii) medzi jedincami. Určitým spôsobom si predávajú informácie o vhodnosti jednotlivých riešení pri výbere alebo pri rekombinačnom procese.

1.3 „No Free Lunch“ teorém

Pojem „No Free Lunch“ teorém (ďalej len NFL) zaviedol David Wolpert spolu s Williamom Macreadym. Tento teorém je štatisticky významný pre optimalizačné algoritmy, ktorý hovorí, že neexistuje univerzálny algoritmus, ktorý dokáže riešiť všetky problémy lepšie ako iné algoritmy. [1, 3] Ako píše aj Ivan Zelinka vo svojej knihe [4], každý algoritmus je vhodný pre riešenie rôznych typov problémov. Pre niektoré problémy je vhodnejší algoritmus A, pre iné problémy zas algoritmus B.

Obrázok 2 znázorňuje o čom hovorí NFL teorém. Akékoľvek dva optimalizačné algoritmy budú ekvivalentné, ak sa ich úspešnosť spriemeruje napriek všetkými možnými problémami. [5] Inak povedané, pre dva rôzne algoritmy A a B existuje toľko scenárov, kedy algoritmus A bude úspešnejší na daný problém ako algoritmus B a naopak, že konečná priemerná úspešnosť oboch algoritmov aplikovaných na všetky možné problémy bude rovnaká.



Obrázok 2: Graf úspešnosti algoritmu A a B na sadu problémov

1.4 Základné pojmy

V nasledujúcich podkapitolách sú definované jednotlivé pojmy, s ktorými je možné sa stretnúť pri EA. Detailnejšie sú rozobrané pojmy ako *jedinec, populácia, účelová funkcia* či *specimen.*

1.4.1 Specimen

Pri centrálnej dogme EVT v kapitole 1.1 sa spomína, že na začiatku EA v kroku č. 2 sa náhodne vytvorí nová populácia. Vytváranie jedincov tejto populácie sa uskutočňuje na základe určitého vzoru tzv. specimena. Specimen je teda vzorový jedinec, ktorý hovorí o tom ako má jedinec populácie vyzerať, určuje počet parametrov jedinca. [6, 9] Na základe tohto vzoru sa vygeneruje celá populácia. Taktiež sa specimen môže využiť ako korekčné pravidlo u jedincov, ktorí prekročia hranice prehľadávaného priestoru. O tomto využití je napísané v podkapitole 1.4.4.

1.4.2 Jedinec

Základným článkom populácie je jedinec. Jedinec predstavuje jedno riešenie daného problému (kombináciu argumentov účelovej funkcie). Ako už z úloh EA vyplýva, hľadá sa jedinec s ideálnym ohodnotením účelovej funkcie, to znamená, že EA sa snažia nájsť optimálnu kombináciu parametrov jednotlivých jedincov. Definícia jedinca záleží od EA, napríklad v Genetickom algoritme (ďalej len GA) je jedinec definovaný jedným chromozómom a pevnou dĺžkou. [7, 8, 9]

1.4.3 Populácia

Populácia je množina jedincov, alebo inak povedané množina možných riešení. Populáciu môžeme reprezentovať ako maticu o veľkosti $N \times (D + 1)$, kde:

- N – počet jedincov,
- D – dimenzia problému,
- + 1 – predstavuje kvalitu (vhodnosť) daného jedinca (riešenia).

V nasledujúcej Tabuľke 1 je názorná ukážka reprezentácie populácie v matici, kde sú 4 jedinci v 5 dimenzionálnom priestore a teda veľkosť matice je 4 x 6. Kvalita riešenia je predstavená ako vhodnosť daného riešenia a teda hodnota fitness.

Tabuľka 1: Znázornenie ukážkovej populácie v matici

Jedinec	Vhodnosť	D1	D2	D3	D4	D5
J1	**9,86**	2,55	45,99	309,28	-208,33	8,55
J2	**8,44**	-4,87	-209,28	12,91	349,56	-65,98
J3	**45,09**	35,09	3,87	-34,77	378,78	-76,34
J4	**22,14**	120,67	-309,28	3,09	-0,44	398,44

1.4.4 Hranice prehľadávaného priestoru

Obvykle argumenty účelovej funkcie môžu nadobúdať iba určite hodnoty, dôvodom je fyzikálna realizovateľnosť či dostupnosť reálneho objektu. Pre zabezpečenie, aby hodnoty argumentov účelovej funkcie boli v povolenom rozmedzí sa používajú rôzne metódy.

V oblasti EA je úplne bežné, že jedinci sa zhlukujú okolo možných riešení, a tým pádom môže nastať uviaznutie v lokálnom optime alebo opúšťajú vymedzené oblasti možných riešení, a tak vznikajú neprípustné riešenia. V EA existuje veľa metód, ktoré nahrádzajú alebo editujú jedincov práve preto, aby zabránili vyššie spomenutým situáciám. [9] Takéto

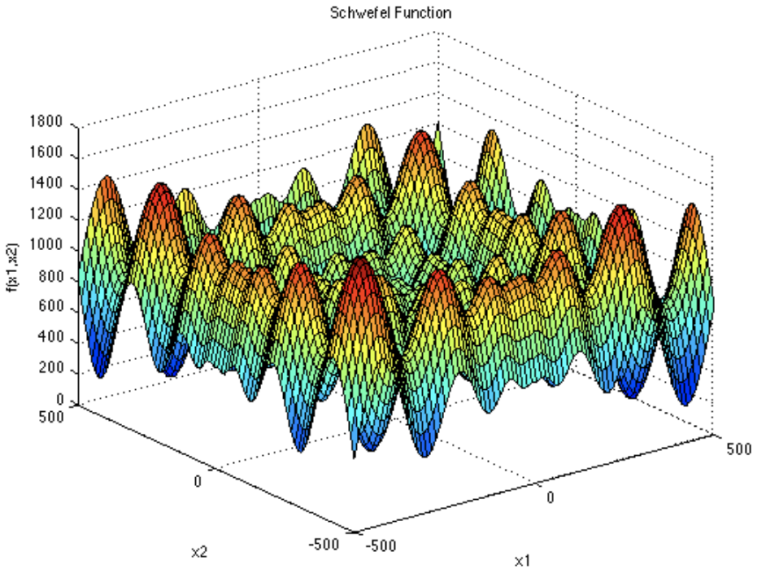
nahrádzanie a editácia jedincov môže zvyšovať stochastičnosť algoritmov, kedy už jedinec nie je úplne výsledkom kríženia. Medzi najpoužívanejšie metódy patria:

- **Random –** jedinec, ktorý prekročí hranice prehľadávaného priestoru, resp. parametre, ktoré prekročia tieto hranice, sú nahradené novými náhodne vygenerovanými parametrami na základe specimena, ktorý bol popísaný v kapitole 1.4.1.
- **Clipping –** jedinec, ktorý prekročil hranice prehľadávaného priestoru, resp. len parametre, ktoré prekročili hranice sú zastavené priamo na hranici. Počas EA môže dochádzať k zhlukovaniu jedincov na hraniciach.
- **Reflection –** ako už aj z anglického názvu vyplýva, jedná sa o odrazenie jedinca. Parameter, ktorý by mal prekročiť hranice je zrkadlovo odrazený naspäť do prehľadávaného priestoru.
- **Periodic -** jedná sa o tzv. simuláciu hyper-gule, kedy parameter, ktorý prekročí hranice z jednej strany, je vrátený o rovnakú vzdialenosť do prehľadávaného priestoru z druhej strany.

V prvom prípade, kedy sú parametre nahradené náhodne vygenerovanými, sa vyskytuje zvyšovanie stochastičnosti, v ostatných troch prípadoch jedinec nie je editovaný stochastickými krokmi, a teda môže byť považované, že je v maximálnej miere zachovaná evolúcia a dynamika populácie.

1.4.5 Účelová funkcia a Fitness

Účelová funkcia môže byť akákoľvek funkcia, ktorá popisuje daný optimalizačný problém a nájdenie jej globálneho extrému (minima alebo maxima) je cieľom optimalizačného procesu. Na účelovú funkciu je tiež možné pozerať ako na geometrický problém, v tomto prípade výstupom účelovej funkcie je jej funkčná hodnota na $D + 1$ rozmernej hyper-ploche. [10] Príkladom môže byť testovacia funkcia Schwefel, zobrazená na Obrázku 3. Funkcia je komplexná a má veľa lokálnych miním. Graf znázorňuje trojrozmerný tvar danej funkcie. Funkcia sa zvyčajne hodnotí na hyper-kocke, kde $x_i \in [-500, 500]$, pre všetky $i = 1, 2, ..., d$, kde d je rovné počtu dimenzií. [11] Viac je o testovaní a testovacích funkciách popísané v kapitole 4.



Obrázok 3: Geometrické znázornenie Schwefel testovacej funkcie [11]

Fitness

Hodnota fitness, alebo inak povedané vhodnosť definuje relatívnu dôležitosť daného riešenia. V prípade, kedy je hľadané globálne minimum, tak jedinec s vyššou hodnotou fitness znamená lepšie riešenie. Hodnota fitness môže byť definovaná rôznymi spôsobmi. [12] Môže byť definovaná na základe hodnoty účelovej funkcie nasledovne:

$$F_i = (1 + \varepsilon) * f_{max} - f_i \quad (1)$$

- F_i – hodnota fitness jedinca i,
- f_i – hodnota účelovej (penalizačnej) funkcie i-teho jedinca,
- f_{max} – najväčšia zaznamenaná hodnota účelovej (penalizačnej) funkcie,
- ε – je veľmi malá hodnota aby sa zabránilo tomu, že hodnota fitness F_i nadobudne hodnotu 0.

1.5 Zástupcovia EVT

Evolučné algoritmy sa inšpirujú prirodzenými procesmi v prírode. Sú to optimalizačné algoritmy, ktorých cieľ je prehľadať priestor a nájsť globálne optimum. Vyznačujú sa vysokou robustnosťou a všeobecne dokážu riešiť veľkú radu problémov. [13] Medzi dve najzákladnejšie skupiny EA patria klasické a rojové EA.

1.5.1 Klasické evolučné algoritmy

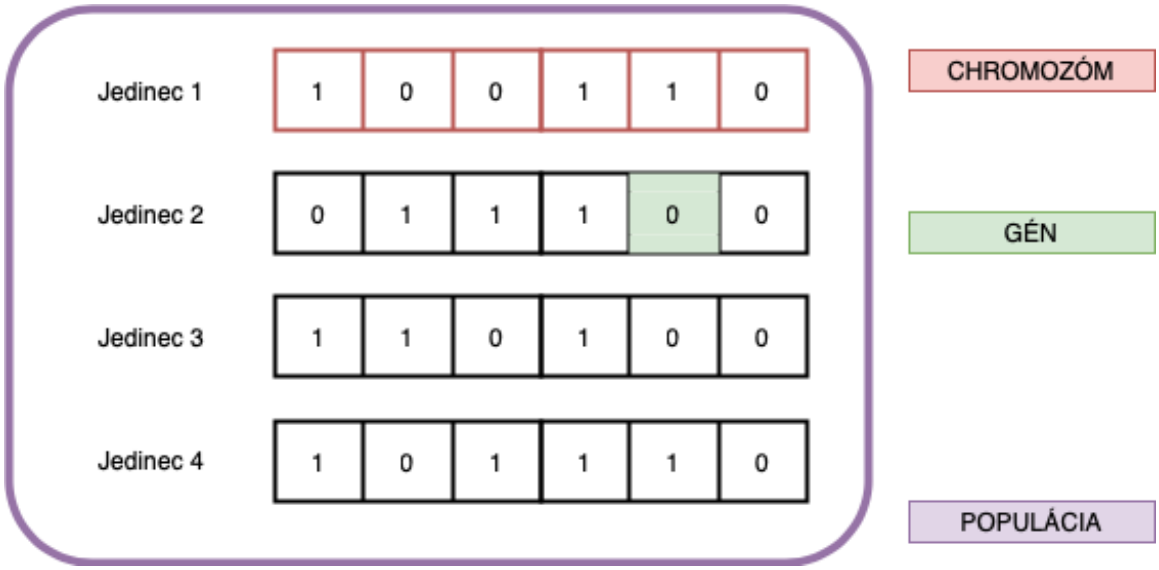
Medzi klasické a najznámejšie EA, okrem iných, patrí s určitosťou aj GA či Diferenciálna Evolúcia (ďalej len DE). V nasledujúcich podkapitolách sa nachádza popis jednotlivých algoritmov, ich základné princípy a existujúce varianty.

1.5.1.1 Genetický algoritmus (GA)

GA je jeden z najznámejších a z historicky najstarších EA, ktorý priamo mimikuje evolúciu potomkov pomocou výberu rodičov a ich kríženia. Typickým rysom GA je, že pracujú s binárnym kódovaním parametrov jedincov. [14, 15]

Základná terminológia

Na Obrázku 4 je možné vidieť základnú terminológiu genetického algoritmu. Zelenou farbou je znázornený gén, ktorý predstavuje jeden parameter jedinca. Červenou farbou je znázornený reťazec jednotlivých génov, ktorý sa nazýva chromozóm a predstavuje jedného jedinca. Populácia, znázornená fialovou farbou je potom množina takýchto chromozómov.

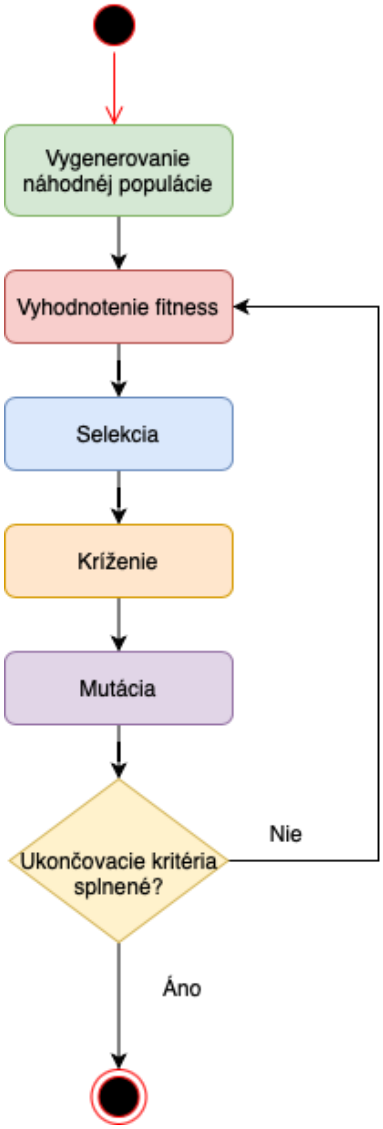


Obrázok 4: Základná terminológia GA

Rovnako ako aj u iných algoritmov, aj tu sa vyskytuje hodnota fitness, ktorá vyjadruje vhodnosť jedinca a je to pretransformovaná hodnota účelovej funkcie.

Podľa stavového diagramu, ktorý je znázornený na Obrázku 5, je možné vidieť základne kroky algoritmu. Na začiatku sa náhodne vygeneruje počiatočná populácia. Ako nasledujúci krok je ohodnotenie týchto riešení. Pod ohodnotením je myslené určenie hodnoty fitness jedincov v počiatočnej populácii. Ďalším krokom je selekcia. Jedná sa o výber jedincov

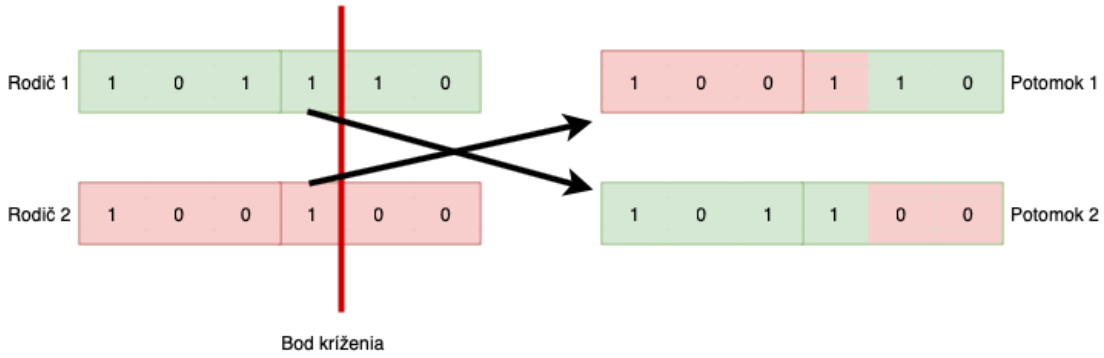
z populácie pre operácie kríženia a mutácie. Selekcia sa môže uskutočniť rôznymi metódami. Medzi tieto metódy patrí *ruletové pravidlo*, *poradová selekcia (rank selection)* či napríklad *selekcia založená na súťaži (tournament selection)*. Tieto pravidlá dajú šancu aj jedincom s nižšou hodnotou fitness, ako keby sa rovno vybrali dvaja jedinci, ktorí majú najväčšiu hodnotu fitness. Aj jedinec s nižšou hodnotou fitness môže mať vhodnú štruktúru parametrov.



Obrázok 5: Stavový diagram GA

Po úspešnom výberu rodičov dochádza k reprodukcii nových potomkov, a to na základne kríženia a mutácie. Znova existuje viacero techník kríženia, ako napríklad jednobodové, dvojbodové, viacbodové či uniformné. Priebeh jednobodového kríženia je znázornený na Obrázku 6. Dvojbodové a viacbodové kríženie prebieha obdobne. Uniformné kríženie funguje na princípe, kedy každý gén z prvého rodiča má 50% pravdepodobnosť, že bude

vybraný do potomka. Ak gén z prvého rodiča nie je vybraný, následne do potomka je vybraný gén z druhého rodiča. Takýmto spôsobom sa postupne prejde všetkými génmi rodičov a vytvoria sa potomkovia.



Obrázok 6: Jednobodové kríženie

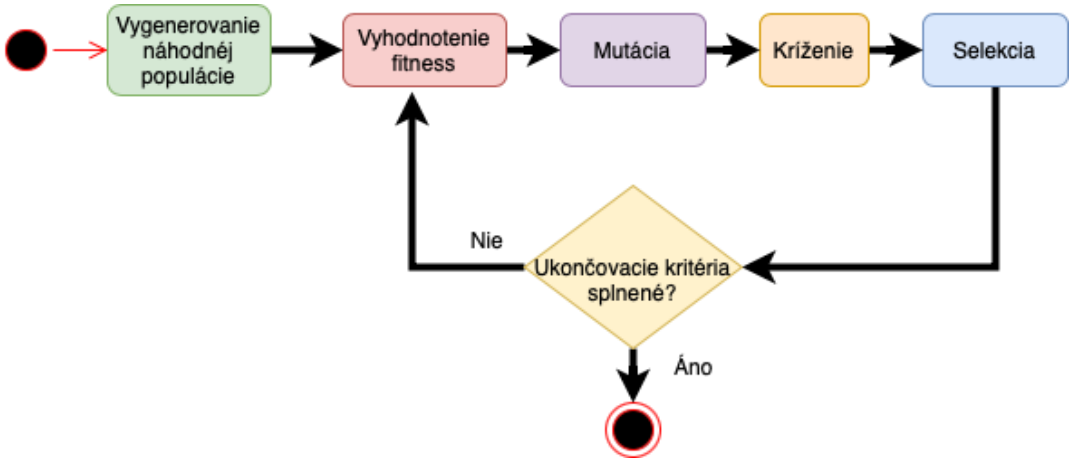
Po krížení prichádza na rad mutácia, čo v tomto prípade znamená len inverzia náhodne vybraných bitov. Rovnako ako kríženie, tak i mutácia sa môže vykonávať rôznymi technikami. Medzi tieto techniky patrí napríklad *jednobodová, dvojbodová či viacbodová* mutácia. K mutácii nemusí nutne dochádzať v každej generácii, či u každého jedinca.

Existujú aj rôzne modifikácie či varianty tohto algoritmu, ako napríklad paralelný GA [16], Messy GA [17], Hybridný [18], Micro GA [19], Steady State GA [20] a rôzne iné. Častým dôvodom vzniku týchto modifikácií je zvýšenie robustnosti, či zlepšenie konvergencie.

1.5.1.2 Diferenciálna evolúcia (DE)

Ďalším veľmi známym algoritmom je Diferenciálna Evolúcia. Od GA sa odlišuje tým, že jednotlivé procesy kríženia a mutácie sú uskutočňované ako vektorové operácie. [21] DE sa vyznačuje tým, že má veľmi netradičný tzv. workflow, pretože na rozdiel od iných EA, kroky selekcie, kríženia a mutácie prebiehajú v opačnom poradí. Ako prvá prebehne mutácia, následne kríženie a ako posledná selekcia.

DE má štyri vstupné parametre, a to *maximálny počet generácií (G_{max}), veľkosť populácie (NP), mutačnú konštantu (F)* a *prah kríženia (CR)*. [22] Na Obrázku 7 je stavový diagram DE, ktorý zobrazuje jednotlivé kroky algoritmu.



Obrázok 7: Stavový diagram DE

1.5.2 Rojové algoritmy

Algoritmy založené na rojoch sa za posledné desaťročie tešia čoraz väčšej popularite. Patria medzi jedny z najskúmanejších heuristík z oblasti evolučných výpočtových techník. [22] Tieto algoritmy majú spoločné to, že sa v nich nenachádza žiadne centralizované riadenie a fungujú čisto len na predávaní si informácii a komunikácii medzi jedincami. V nasledujúcich podkapitolách sú niektoré najznámejšie stručne popísané, ako napríklad Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) a Artificial Bee Colony (ABC) algoritmus. V 2. kapitole je následne dôkladne rozobraný algoritmus SOMA, ktorý je predmetom tejto diplomovej práce a ktorý tiež spadá medzi rojové EA.

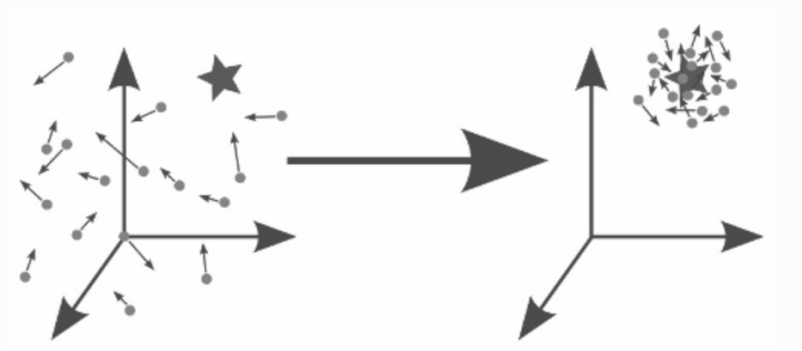
1.5.2.1 Optimalizácia roju častíc (Particle Swarm Optimization – PSO)

PSO sa inšpiruje a simuluje chovanie roju vtákov (rýb). Jedno riešenie algoritmu a teda jedinec sa nazýva častica. Každá častica má nasledujúce parametre: *aktuálnu polohu, rýchlosť* a *pamäť doposiaľ najlepšieho úspechu pri hľadaní* (doposiaľ najlepšiu nájdenú polohu). [23] Tieto častice sa pohybujú po priestore, spolu medzi sebou komunikujú a predávajú si informácie o ich úspechoch. PSO algoritmus sa dá jednoducho zhrnúť v nasledujúcich krokoch:

1. Vygenerovanie náhodnej počiatočnej populácie.
2. Každej častici je v prvej iterácii nastavená rýchlosť na hodnotu 0.
3. Vypočíta sa hodnota fitness (z pozície každej jednej častice).
4. Častica s minimálnou alebo maximálnou hodnotou optima (záleží od optimalizačného problému) uloží svoju aktuálnu polohu do spoločnej pamäte populácie. Táto hodnota nazýva *gBest* (global best).

5. Každá častica zisťuje, či jeho aktuálna poloha nie je lepšia ako jeho doteraz najlepšia nájdená. Ak je tomu tak, aktualizuje svoju doposiaľ najlepšiu nájdenú pozíciu na aktuálnu. Táto hodnota sa nazýva *pBest* (personal best).
6. Každá častica upraví svoj vektor rýchlosti a pozíciu. Objavujú sa tu 2 tendencie chovania, a to kognitívne (častica smeruje k svojmu najlepšiemu riešeniu) a sociálne (častica smeruje ku globálnemu najlepšiemu riešeniu).
7. Znova sa pokračuje bodom 3 až kým nie sú splnené ukončovacie podmienky.

Výhodou PSO je jednoduchá implementácia a rýchla konvergencia k optimu pre širokú škálu účelových funkcií. [24] Konvergencia je znázornená na Obrázku 8. Na ľavej strane obrázku je vidieť rozloženie častíc v priestore v počiatku algoritmu a na pravej strane obrázku rozloženie častíc v priestore po skončení algoritmu.

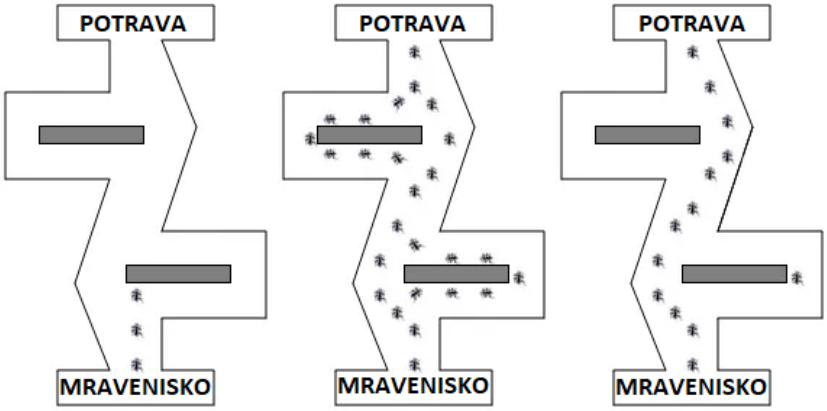


Obrázok 8: Častice v 3D priestore v iterácii 0 až N [23]

1.5.2.2 Optimalizácia mravčej kolónie (Ant Colony Optimization – ACO)

Algoritmus sa inšpiruje a simuluje správanie mravčej kolónie pri hľadaní potravy. Prvý algoritmus mravčej kolónie navrhol v roku 1992 Marco Dorigo a predstavil ho vo svojej dizertačnej práci. [25] Princíp tohto algoritmu spočíva v tom, že mravci si značkujú pomocou feromónov cestu k potrave. Čím viac mravcov prejde po danej trase, tým intenzívnejší zápach feromónov ostáva. Následne ďalší mravci, ktorí hľadajú potravu sa uberajú cestou kde je najintenzívnejší zápach, pretože je to znamenie, že je tam dobrý zdroj potravy. [25, 26]

Mravci sa pri hľadaní potravy musia vyhýbať rôznym prekážkam, z toho dôvodu vzniká viac trás, ktorými sa dokážu dostať ku zdroju potravy, pričom si vyberajú najkratšiu trasu. Z toho dôvodu algoritmus nájde využitie pri hľadaní dobrých ciest v grafoch, napríklad u problému obchodného cestujúceho. Obrázok 9 vyobrazuje popisované správanie mravcov.



Obrázok 9: Správanie mravcov pri hľadaní potravy

1.5.2.3 Algoritmus umelej včelej kolónу (Artificial Bee Colony – ABC)

ABC je inšpirované správaním včiel pri hľadaní potravy. Funguje na princípe kolektívnej inteligencie včelích rojov. Tento model vyvinula Valery Tereshko. Hovorí o tom, že roj sa skladá zo *zamestnaných včiel*, *vyčkávajúcich včiel* v úle a zo *nezamestnaných včiel* hľadajúcich potravu. [27] Zamestnané včely nosia informácie o zdroji potravy, kde aktuálne ťažia a zdieľajú tieto informácie s ostatnými včelami čakajúcimi v úli. Nezamestnaná včela je taká, ktorá ešte len hľadá zdroj potravy kde by ťažila. Každý zdroj potravy je ohodnotený hodnotou *fitness.* Vhodnosť jednotlivých zdrojov sa počíta na základe vzdialenosti od úľa, bohatstva zdroja, obťažnosti získania bohatstva tohto zdroja a iné. Takýto zdroj potravy v oblasti optimalizačných úloh predstavuje jedno možné riešenie optimalizačného problému. Kvalita riešenia je zhodná s množstvom nektáru v danom zdroji. Veľkosť populácie udáva počet zamestnaných včiel. Nezamestnané včely nemajú priradené súradnice riešenia, ale iba náhodne hľadajú nové riešenia. [28] Rovnako ako ACO aj algoritmus ABC sa hodí na problémy, kde sa hľadajú dobré cesty v grafe.

1.6 Ciele EVT

Na rozdiel od deterministických algoritmov, evolučné algoritmy nedokážu zaručiť nájdenie optimálneho riešenia, avšak aj napriek tomu dokážu dosahovať akceptovateľné výsledky, dokonca aj v prípadoch, kedy deterministické algoritmy alebo aj iné zlyhávajú. Na základe mnohých uskutočnených testov na reálnych účelových funkciách a testovacích funkciách, sa dá konštatovať, že evolučné algoritmy sú veľmi výkonné a vhodné pre globálnu optimalizáciu. [6, 29]

Cieľom EVT je za pomerne krátky čas dosiahnuť optimálne riešenie. Môžu však nastať nežiadúce situácie ako napríklad uviaznutie v lokálnom optime, čo môže mať za príčinu, že nie je možné nájsť globálne optimum.

Pomocou EVT sa môžu riešiť zložitejšie optimalizačné problémy, komplexné problémy, či multi-kriteriálne optimalizačné problémy. Použitie algoritmov, ktoré využívajú evolučné techniky je veľmi široké ako napríklad v odvetví informatiky, priemysle, doprave, lekárstve, atď. [1]

2 SAMOORGANIZUJÚCI SA MIGRAČNÝ ALGORITMUS

SOMA, tzv. samo-organizujúci sa migračný algoritmus vyvinul a publikoval pán Profesor Ivan Zelinka v roku 1999. Rovnako ako ostatné rojové EA spomenuté v kapitole 1.5.2, aj SOMA je založená na modeli sociálnej kooperácie a komunikácie medzi jedincami pri riešení optimalizačného problému. Algoritmus pracuje s vektormi a vektorovými operáciami. Ako každý iný EA aj SOMA si zakladá na princípe evolučného cyklu, ktorý je nazývaný ako migračné kolo, kedy na konci tohto kola vzniká nová populácia. Rozdielom od klasických EA je, že počas migračného kola nevznikajú noví potomkovia, ale pôvodní sa presúvajú pomocou skokov v prehľadávanej oblasti. Vďaka tomu, že napodobuje chovanie skupiny inteligentných jedincov sa SOMA radí medzi rojové EA. [4, 30]

2.1 Princípy

Rovnako ako aj iné EA, aj SOMA sa riadi určitými *riadiacimi* a *ukončovacími* parametrami. V nasledujúcej Tabuľke 2 sa nachádzajú parametre, ktoré ovplyvňujú samotný beh a výsledky algoritmu.

Tabuľka 2: Zoznam parametrov algoritmu SOMA

Parameter	Doporučený rozsah	Druh parametra
PathLength	[1,1; 5>]	Riadiaci
step	[0,11; *PathLength*]	Riadiaci
PRT	[0; 1]	Riadiaci
D	Určené optimalizačným problémom	Riadiaci
PopSize	[10; definuje užívateľ]	Riadiaci
Migrácia	[10; definuje užívateľ]	Ukončovací

Popis jednotlivých parametrov:

- ***PathLength*** – dĺžka cesty,
- ***step –*** veľkosť kroku,
- ***PRT*** – perturbácia,
- ***D –*** dimenzia problému,
- ***PopSize –*** veľkosť populácie,
- ***Migrácia –*** počet opakovaní,

Na začiatku algoritmu je vygenerovaná náhodná populácia podľa rovnice 2, tak aby boli zachované hranice prehľadávaného priestoru.

$$Jedinec_j = x_j^{low} + rand[0,1] * (x_j^{high} - x_j^{low}) \quad (2)$$

- x_j^{low} – najnižšia hodnota hraníc j-teho atribútu
- x_j^{high}– najvyššia hodnota hraníc j-teho atribútu
- $rand[0,1]$ – náhodne vygenerované číslo od 0 po 1 s rovnomerným rozdelením

2.1.1 Základné pojmy

V tejto podkapitole sú popísané základné pojmy a terminológie algoritmu SOMA, ako je napríklad *Leader, perturbácia* a *perturbačný vektor.*

2.1.1.1 Leader

Všetci jedinci v populácii sa ohodnotia a jedinec s najlepšou vhodnosťou je zvolený za Leadra. Leader sa používa len v niektorých variantách SOMA algoritmu, napríklad vo variante All To One a All To One Rand. Medzi varianty, ktoré nemajú Leadra patria varianty All To All a All To All Adaptive. Bližší popis variant SOMA algoritmu sa nachádza v kapitole 2.2.

2.1.1.2 Perturbácia

V klasických EA dochádza počas evolučného cyklu k mutácii. V prípade SOMA algoritmu sa mutácia nazýva tzv. *perturbáciou.* Názov je odlišný z toho dôvodu, že nedochádza doslova k mutovaniu jedincov, ale migrácia jedinca je náhodne prerušovaná. Riadiaci parameter *PTR* vyjadruje to, ako silná perturbácia prebehne a teda, koľko parametrov jedinca bude pozmenených. Ďalším novým terminologickým výrazom je perturbačný vektor *PRTVector*, ktorý sa generuje na základe *PRT* parametra. *PRTVector* sa generuje pre každého jedinca zvlášť a v každom migračnom kole nový. Tento vektor je definovaný na základe nasledujúcej rovnice 3:

$$PRTVector_j = \begin{cases} 1, & ak\ rand_j < PRT \\ 0, & inak \end{cases} \qquad pre\ j = 1, 2, \ldots, D \quad (3)$$

Pre každý parameter jedinca je generované náhodné číslo. Ak toto náhodne vygenerované číslo je menšie ako hodnota *PRT* parametra, ktorá bola nastavená na začiatku pred spustením algoritmu, tak je danému prvku perturbačného vektora priradená hodnota 1. Ak náhodne vygenerovaná hodnota je väčšia ako hodnota *PRT* parametra, tak danému prvku sa priradí hodnota 0 v perturbačnom vektore. [30] Pre ľahšie pochopenie je uvedený

nasledujúci príklad: je daný jedinec, ktorý má 5 parametrov, na začiatku algoritmu parameter *PRT* bol nastavený na hodnotu 0,4. V Tabuľke 3 v ľavom stĺpci sa nachádzajú náhodne vygenerované hodnoty pre jednotlivé parametre jedinca a v pravom stĺpci sú priradené hodnoty perturbačného vektora podľa rovnice 3.

Tabuľka 3: Ukážka perturbačného vektoru

$rand_j$	$PRTVector_j$
0,777	0
0,378	1
0,119	1
0,876	0
0,099	1

2.1.1.3 Kríženie

Rovnako ako aj mutácia tak ani kríženie v algoritme SOMA neprebieha ako u klasických EA. Pri pohybe jedinca po hyper-ploche (pomocou diskrétnych skokov) vzniká tzv. sekvencia potomkov, z ktorej prežije len ten najlepší. Rozdiel od klasických EA je, že vzniká celá sekvencia potomkov od jedného jedinca a nie len dvaja potomkovia ako je to napríklad u GA.

Jedinec sa pohybuje v diskrétnych skokoch smerom k *Leadrovi*, ale zároveň je ovplyvnený hodnotou *PRTVector*. V každom skoku sa vypočítavá aj vhodnosť nových potomkov. Jedinec si pri svojom putovaní potom uchováva súradnice potomka s najlepšou vhodnosťou. Najlepší potomok vstupuje do ďalšieho migračného kola. Takýto pohyb je znázornený aj na Obrázkoch 10 a 11 na základe varianty algoritmu. Pohyb sa riadi na základe rovnice 4.

$$x_{i,j}^{k+1} = x_{i,j}^{k} + \left(x_{L,j}^{k} - x_{i,j}^{k}\right) * t * PRTVector_j \tag{4}$$

- $x_{i,j}^{k+1}$ – nová pozícia i-teho riešenia pre iteráciu k+1
- $x_{i,j}^{k}$ – aktuálna pozícia i-teho riešenia
- $x_{L,j}^{k}$ – pozícia Leadra pre iteráciu k
- t – krok z i-teho riešenia k Leadrovi generovaný z <0, *pathLength*>
- $PRTVector_j$ – generuje sa pre každý nový *t* skok, udáva smer pohybu jedinca

2.1.2 Pseudokód

Samotný algoritmus SOMA je veľmi jednoduchý na implementáciu a dalo by sa ho zhrnúť do týchto 5 krokov:

1. Definícia riadiacich a ukončovacích parametrov.
2. Tvorba počiatočnej populácie.
3. Migračné kolá.
4. Testovanie ukončovacích parametrov.
5. Zastavenie algoritmu. Výstupom algoritmu je najlepšie nájdené riešenie.

Nasledujúce Pseudokód 1 symbolicky popisuje jednotlivé kroky a funkcionality základnej varianty algoritmu SOMA - All To One.

Vstupy:
- **x** – náhodne vygenerovaná populácia na základe *specimena*
- nastavenie riadiacich a ukončovacích parametrov
- definovanie $\mathbf{f}_{\text{cost}}$ – účelovej funkcie, ktorá vracia vhodnosť aktuálneho riešenia

pre $i < Migrace$ **rob:**
  selekcia najlepšieho riešenia z populácie – *Leadra*
  pre $j \leq PopSize$ **rob:**
    vyber j-teho jedinca z populácie
    vypočítaj nové pozície podľa rovnice 4
    ulož si najlepšie nájdene riešenie do novej populácie
  ak je splnená iná ukončovacia podmienka **tak:**
    zastav algoritmus

Výstup: Najlepšie nájdené riešenie.

Pseudokód 1: Algoritmus SOMA

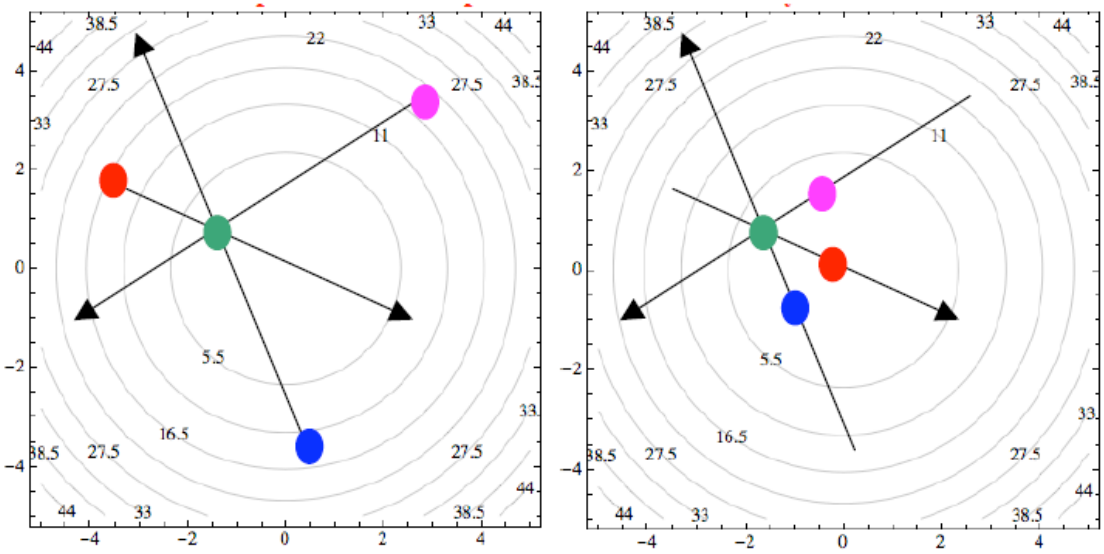
2.2 Varianty

V nasledujúcom texte sú rozobrané štyri základné stratégie navrhnuté v oficiálnom koncepte algoritmu SOMA. Stratégie migrácií sú nasledovné: All to One (ATO), All To One Random (ATR), All To All (ATA) a All To All Adaptive.

2.2.1 Všetci k jednému (All To One)

Na začiatku pred migráciou je zvolený tzv. Leader. Je to jeden jedinec, ktorý sa aktuálne nachádza na najlepšej pozícii na hyper-ploche (v najhlbšej alebo v najvyššej pozícii na

hyper-ploche) oproti ostatným jedincom. Následne ostatní jedinci postupne jeden po druhom migrujú práve k tomuto zvolenému Leadrovi. Migrácia vyzerá nasledovne. Jedinec migruje k Leadrovi tak, že robí skoky o veľkosti *step*. Následne sa vráti na svoju štartovnú pozíciu a zapamätá si miesto, kde našiel najlepšiu pozíciu na hyper-ploche počas jeho migrácie. Rovnako postupujú aj ostatní jedinci. Ak už všetci jedinci vykonali migráciu, na záver sa každý jedinec presunie na svoj najlepší extrém, ktorý pri migrácii našiel. Týmto končí migračné kolo, znova sa volí zo všetkých jedincov Leader a postup sa opakuje ako bol už vyššie popísaný. Názorná ukážka migrácie je na Obrázku 10.



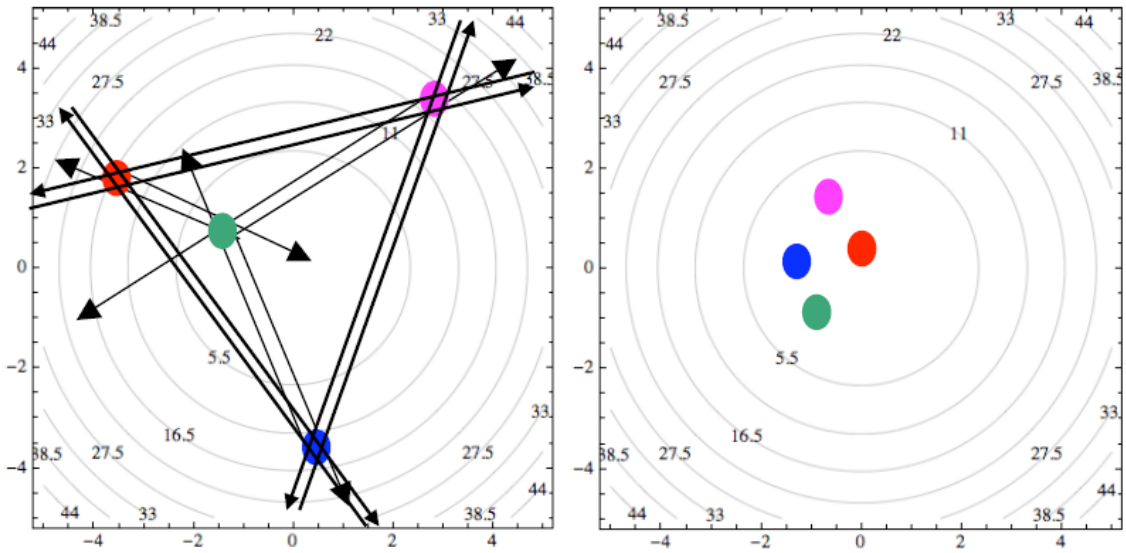
Obrázok 10: Princíp stratégie All To One, zelený bod predstavuje Leadra a ostatné body sú jedinci v populácii, stav pred migráciou vľavo a stav po migrácii vpravo [31]

2.2.2 Všetci k jednému náhodne (All To One Random)

V tejto stratégii je Leader zvolený náhodne a nie na základne hodnoty účelovej funkcie, ako to bolo pri predchádzajúcej stratégii. Okrem iného každý jedinec má svojho Leadra náhodne zvoleného zvlášť. V praxi by táto stratégia vyzerala nasledovne: je daný jedinec, náhodne sa zvolí Leadra zo zvyšnej populácie. Jedinec vykoná migráciu, vráti sa na štartovnú pozíciu a pamätá si svoju najlepšiu nájdenú pozíciu pri migrácii. Takto rovnako budú migrovať aj ostatní jedinci s tým, že pre každého jedinca sa volí náhodne nový Leader. Na záver, keď všetci jedinci vykonali migráciu, sa presunú na svoje najlepšie nájdené extrémy a končí jedno migračné kolo.

2.2.3 Všetci ku všetkým (All To All)

Jedna z menej sofistikovaných stratégií, kde neexistuje Leader. Oproti predchádzajúcim dvom stratégiám je výpočtovo náročnejšia, avšak dochádza k väčšej explorácii. Stratégia vyzerá nasledovne: jedinec migruje postupne ku každému jedincovi zo štartovacej pozície. Po tom, čo dokončí migráciu, vráti sa na štartovnú pozíciu a pamätá si svoju najlepšiu pozíciu, ktorú pri migráciách k jednotlivým jedincom našiel. Keď dokončia migráciu všetci jedinci, tak sa následne presunú na svoje najlepšie nájdené extrémy a tým končí kolo migrácie a môže začať ďalšie. Na obrázku 11 sa nachádza názorná ukážka spomínanej stratégie.



Obrázok 11: Princíp stratégie All To To All, zelený bod predstavuje Leadra a ostatné body sú jedinci v populácii, stav pred migráciou vľavo a stav po migrácii vpravo [31]

2.2.4 Adaptívne všetci ku všetkým (All To All Adaptive)

Táto stratégia je veľmi podobná stratégii All To All, ale s tým rozdielom, že jedinec pri svojich migráciách ku zvyšným jedincom sa nevracia na štartovaciu pozíciu, no pri migrácii k danému jedincovi sa vráti na pozíciu, kde našiel najlepšiu pozíciu. Z tejto pozície preňho najlepšieho nájdeného extrému štartuje migráciu k ďalšiemu jedincovi z populácie a algoritmus sa rovnako opakuje pre všetkých jedincov.

2.3 Aplikácie

V nasledujúcich podkapitolách sú stručne predstavené niektoré z aplikácii algoritmu SOMA. Algoritmus svoje využitie našiel ako v reálnom svete, tak aj vo virtuálnom svete a to napríklad v hrách či v grafike.

2.3.1 Inžinierske aplikácie

Langmuirova sonda

Jednou z aplikácii v reálnom svete je využitie algoritmu SOMA pre vyladenie Langmuirovej sondy. Algoritmus bol využitý pre automatický odpočet štrnástich Fourierových výrazov vo vysokofrekvenčnom tvare vlny. Langmuir sú diagnostické nástroje, ktoré sa používajú na stanovenie iónovej hustoty a distribúcie elektrónovej energie v plazmových procesoch. [32]

Na spomínaný problém bolo už v minulosti použité Simulované Žíhanie ako aj Diferenciálna Evolúcia. Z tohto dôvodu sa SOMA spolu so Simulovaným Žíhaním a Diferenciálnou Evolúciou výkonnostne porovnávali a SOMA jednoznačne prekonala oba spomínane algoritmy.

Kvantovanie farieb

Kvantovanie farieb je bežná technika spracovania obrazu na zníženie počtu zreteľných farieb v obraze. Výber farieb, ktoré tvoria farebnú paletu je náročné, pretože určujú výslednú kvalitu obrazu. Na GECCO 2020 (Genetic and Evolutionary Computation Conference) bol predstavený nový kvantizačný algoritmus založený na algoritme SOMA. [33]

2.3.2 Aplikácie v hrách

Tic Tac Toe

Hra založená na princípe tzv. Piškvoriek. Je dostupná ku stiahnutiu na Google Play. Bol to vedecký experiment, kde bola SOMA ako evolučný algoritmus využitá v tejto hre pre prehľadávanie stavového priestoru. Hráč má teda za spoluhráča algoritmus SOMA. Vyhrá ten hráč, ktorý má ako prvý 5 rovnakých symbolov (krížiky alebo krúžky) pod sebou, vedľa seba alebo v diagonálnom smere. [32, 34]

StarCraft: Brood War

Ďalším experimentom bola strategická hra, ktorá je názornou ukážkou ako dokáže program založený na evolučných algoritmoch nahradiť človeka v strategickej hre. V hre sa nachádza využitie nekonvenčných techník ako napríklad evolučný výpočet, ale aj klasické techniky umelej inteligencie. V spomínanej hre je umelo vytvorený hráč, ktorý je kombináciou rozhodovacieho stromu a algoritmu SOMA. Poskytuje tak efektívny a koordinovaný pohyb

jednotiek po bojovej krajine. Tento výskum ukázal potencionálny prínos evolučných výpočtov aj v oblasti strategických hier. [32]

3 MODERNÉ VARIANTY ALGORITMU SOMA

Rovnako ako iné rojové algoritmy, tak aj SOMA sa v priebehu rokov neustále vyvíja a snaží sa vylepšiť si výkonnosť. Vďaka tomuto trendu vznikli mnohé pokročilejšie verzie prispôsobujúce sa problémom, ktoré riešia.

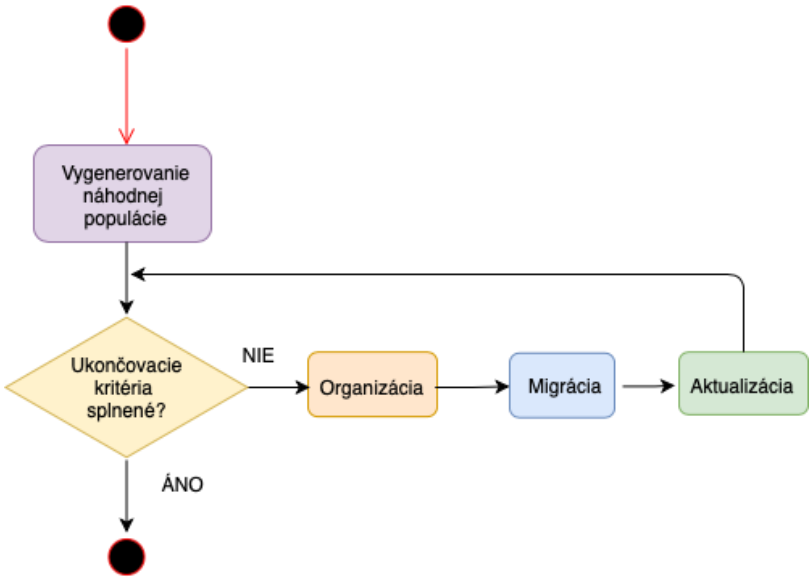
Vzniklo mnoho variant algoritmu, medzi ktoré patrí napríklad C-SOMGA, ktorá využíva výhody genetického algoritmu a funkcií SOMA na riešenie obmedzených nelineárnych problémov. Ďalšia hybridizácia, ktorá stojí za zmienku bola CSOMA, ktorá je kombináciou SOMA prístupu a Kultúrneho algoritmu. [35]

V nasledujúcich podkapitolách je zadefinovaných niekoľko moderných a pomerne nových variant algoritmu SOMA, ktoré vznikli v posledných rokoch. Popísaná je napríklad T3A SOMA (Team To Team Adaptive), ktorá bola predstavená v roku 2019, Pareto SOMA, rovnako publikovaná v roku 2019, ESP – SOMA (The Ensemble of Strategies and Perturbation Parameter) publikovaná v rovnakom roku, či SOMA – CL (with Clustering-aided Migration), ktorá bola predstavená v roku 2020.

3.1 T3A SOMA (Team To Team Adaptive)

T3A SOMA bola publikovaná v roku 2019 [35] a testovaná v rámci CEC 2019 100 digits competition, kde získala 4. miesto z celkového počtu 38 algoritmov.

Počiatočná populácia sa generuje rovnako ako v základnej verzii SOMA, a to na základe rovnice č. 2, ktorá bola popísaná v kapitole 2.1.1.1.



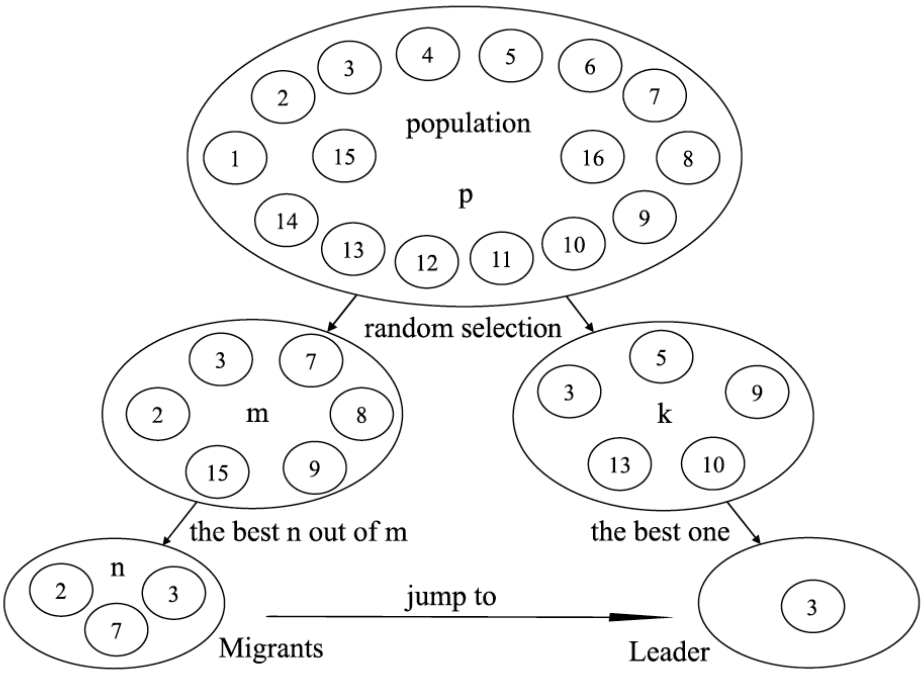
Obrázok 12: Stavový diagram T3A SOMA

Ďalej sa algoritmus T3A SOMA skladá z troch hlavných procesov, ktoré sa opakujú, a to organizácia, migrácia a aktualizácia, ako je znázornené v stavovom diagrame vyššie na Obrázku 12.

V nasledujúcej podkapitolách sú popísané spomínané tri hlavné procesy T3A SOMA algoritmu.

3.1.1 Organizácia

V procese organizácie dochádza ku dvom hlavným činnostiam a to ku výberu jedincov, ktorý sa budú pohybovať, tiež nazývaný ako *migranti* a ku výberu *Leadra*. Samotný proces organizácie prebieha veľmi jednoducho a tento proces je znázornený na Obrázku 13. Na začiatku sa náhodne vyberie *m* jedincov z populácie a z týchto *m* jedincov sa vyberie *n* najlepších jedincov, kedy platí, že $n \leq m$. Títo jedinci sa stávajú migrantmi. Pre výber Leadra funguje obdobný proces, kedy náhodne je vybraných *k* jedincov z populácie a najlepší jedinec z týchto *k* jedincov sa stáva Leadrom. Následne vybraný migranti sa pohybujú k vybranému Leadrovi. [35]



Obrázok 13: Proces organizácie [35]

Môže však nastať situácia, kedy zvolený Leader je aj jedným z migrantov, ako tomu je aj na Obrázku 13. Je samozrejmosťou, že ak nastane takáto situácia, takýto jedinec vynecháva proces migrácie.

Dôležitou úlohou procesu organizácie je preskúmanie sľubných pod-priestorov. Ak hodnoty *m*, *n* a *k* sú rovnaké ako veľkosť populácie, vtedy sa algoritmus správa ako klasický algoritmus SOMA All To One. Avšak ak hodnoty *m*, *n* a *k* sú rôzne od veľkosti populácie a teda menšie ako je veľkosť populácie, no sú to väčšie čísla, tak algoritmus prehľadáva v menších pod-priestoroch, a tým sa teda zameriava na najlepšiu skupinu jedincov v populácii. Takéto počínanie vážne vplýva na rozmanitosť populácie počas toho, ako algoritmus prechádza migračnými kolami. V opačnom prípade, kedy by hodnoty *m*, *n* a *k* boli nižšie čísla, dochádzalo by k tomu, že algoritmus hľadá vo veľkom priestore a to by bolo veľmi neefektívne. Takýmto nastavením algoritmus zistí rýchlo a pomerne presne polohu globálneho extrému ak sa jedná o jednoduché funkcie a malý počet dimenzií. Avšak pre komplexné funkcie a veľký počet dimenzií, algoritmus môže uviaznuť v lokálnom extréme. [35]

3.1.2 Migrácia

V pôvodnej verzii algoritmu SOMA migrácia prebieha tak, že jedinci sa pohybujú k Leadrovi podľa stanoveného pevného kroku *step* až kým nedosiahnu hodnotu predom danej dĺžky trasy *pathLength.* Rovnako ako aj tieto dva parametre, aj parameter *PRT* je pred začatím nastavený na pevnú hodnotu. Tieto parametre patria medzi riadiace parametre a sú nastavené na začiatku pred samotným spustením algoritmu.

V prípade T3A SOMA algoritmu, *PRT* je adaptívnym parametrom, voči ktorému algoritmus SOMA vykazuje veľkú citlivosť. Čím hodnota tohto parametru je bližšie k hodnote 1, tým má jedinec väčšiu tendenciu pohybovať sa práve priamo k Leadrovi. V opačnom prípade, kedy parameter sa blíži k hodnote 0, tak jedinec má tendenciu prieskumu okolia. [35] Z tohto dôvodu T3A SOMA začína na nízkej hodnote *PRT* parametra a následne sa zvyšuje o počet migračných kôl podľa nasledujúcej rovnice 5:

$$PRT = 0{,}05 \ + 0{,}90\frac{FEs}{MaxFEs} \tag{5}$$

- *FEs* – aktuálny počet ohodnotení účelovej funkcie
- *MaxFEs* – maximálny počet ohodnotení účelovej funkcie

Rovnako ako *PRT*, aj parameter *step* v tomto prípade je adaptívny. Veľkosť kroku je redukovaná podľa rovnice č. 6. Táto redukcia sa robí z dôvodu, že T3A sa snaží objaviť sľubné pod-priestory, a preto treba aby hodnota parametru *step* bola nižšie číslo. V prípade, že *step* nadobúda väčšiu číselnú hodnotu, rýchlosť algoritmu sa zrýchli a zníži sa počet ohodnotení účelovej funkcie.

$$step = 0{,}15 - \ 0{,}08\frac{FEs}{MaxFEs} \tag{6}$$

- *FEs* – aktuálny počet ohodnotení účelovej funkcie
- *MaxFEs* – maximálny počet ohodnotení účelovej funkcie

Ďalšou odlišnosťou je, že počet skokov je fixný, a teda neviaže sa na parameter *pathLength*, ako tomu je v klasickej verzii SOMA. To znamená, že každý jedinec sa pohybuje smerom k Leadrovi o *Njumps* (N skokov) a v týchto bodoch doskoku je jedinec ohodnotený fitness funkciou. [35]

3.1.3 Aktualizácia

Proces aktualizácie prebieha rovnako ako u klasického algoritmu. Pohybujúci sa jedinec si počas svojho putovania zapamätá svoju najlepšiu pozíciu, ktorú na konci putovania porovná s pôvodnou pozíciou. Ak nová pozícia je lepšia ako pôvodná, tak pôvodná je nahradená novou pozíciou.

3.1.4 Pseudokód

Pre ľahšie predstavenie fungovania algoritmu sa pozrite na nasledujúci Pseudokód 2:

Vstupy:
- Náhodne vygenerovaná populácia na základe *specimena*
- Ohodnotenie jedincov v počiatočnej populácii
- Nastavenie riadiacich a ukončovacích parametrov

pokiaľ ukončovacie podmienky nie sú splnené **rob:**
  aktualizuj *PRT* a *step*
  vyber migrantov
  pre $i = 1$ až počet migrantov **rob:**
    vyber Leadra
    ak migrant nie je Leader **rob:**
      migrant skáč k Leadrovi
      skontroluj neprekročenie hraníc
      prepočítaj funkciu fitness
      aktualizuj najlepšiu nájdenú pozíciu migranta

Výstup: Najlepšie nájdené riešenie.

Pseudokód 2: Algoritmus T3A SOMA

3.2 Pareto SOMA

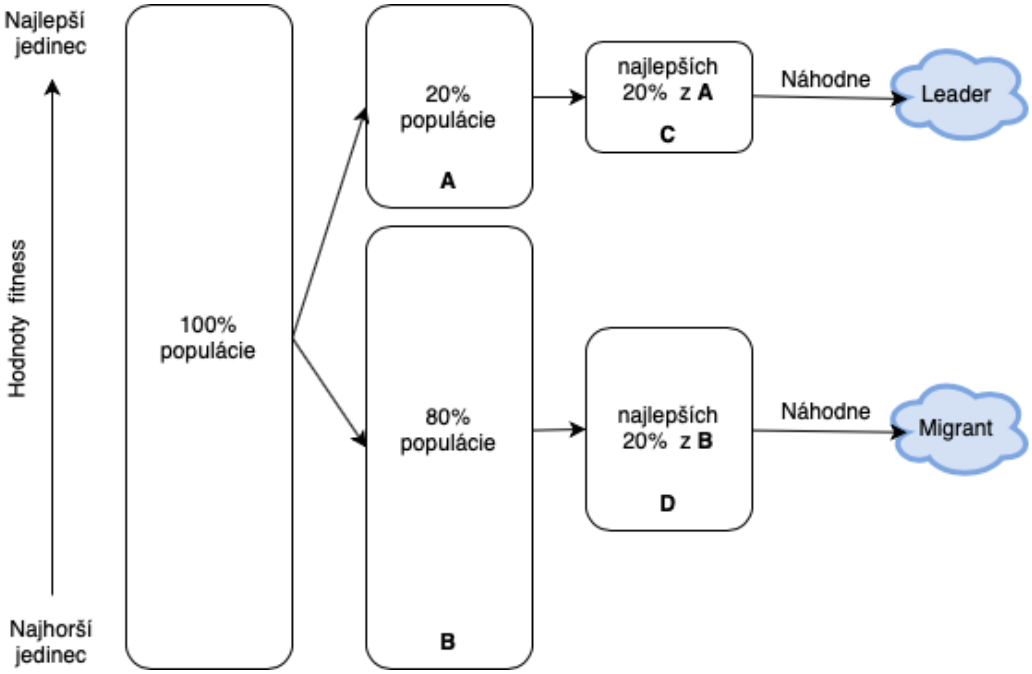
Pareto SOMA bola publikovaná v roku 2019 [36] a testovaná v rámci CEC 2019 100 digits competition, kde sa umiestnila na 6. mieste z celkového počtu 38.

Počiatočná populácia podobne ako v základnej verzii SOMA a T3A verzii sa generuje na základe rovnice č. 2, ktorá bola popísaná v kapitole 2.1.1.1. V ďalšom kroku dochádza k ohodnotení počiatočnej populácie na základe fitness funkcie a algoritmus vstúpi do migračných kôl. Ďalej v porovnaní s T3A verziou, aj Pareto obsahuje 3 hlavné časti a to organizácia, migrácia a aktualizácia, ako bolo znázornené na Obrázku 12.

3.2.1 Organizácia

V procese organizácie sa vyberá jeden jedinec inak nazvaný *Migrant*, ktorý sa bude pohybovať k *Leadrovi* so snahou nájsť lepšiu pozíciu. Tento proces zohráva zásadnú úlohu pri skúmaní sľubných pod-priestorov.

Na výber Migranta a Leadra je využitý Pareto princíp a to takým spôsobom, že populáciu delíme na dve skupiny. Skupina „A“ obsahuje 20% počtu jedincov, ktorí majú 80% hodnoty populácie a teda predpokladá sa, že obsahuje väčšinu najkvalitnejších riešení. Druhá skupina „B“ obsahuje 80% počtu jedincov, ale len 20% hodnoty populácie a teda zahŕňa väčšinu menej kvalitných riešení. Jednoduchšie povedané, v každom migračnom kole sú jedinci v populácii zoradení podľa vzrastajúcej hodnoty fitness. Takáto zotriedená populácia je následne rozdelená na 2 časti, kde prvá časť „A“ obsahuje 20% z počtu jedincov s najlepšou hodnotou fitness a druhá časť označená ako „B“ obsahuje zvyšných 80% počiatočnej populácie. Následne z každej z týchto 2 skupín vyberieme 20% najlepších jedincov a vzniknú nám skupiny „C“ a „D“. Ako posledný krok je náhodný výber jedného jedinca zo skupiny „C“, ktorého označíme za Leadra a jedného jedinca zo skupiny „D“, ktorého označíme za migrujúceho jedinca. Pre ľahšie pochopenie a lepšiu vizualizáciu procesu organizácie sa pozrite na Obrázok 14, ktorý popisuje celý proces výberu Leadra a Migranta. [36]



Obrázok 14: Proces organizácie Pareto SOMA

Z popísaného procesu výberu Leadra, nie je vybraný striktne najlepší jedinec z populácie, ale náhodne je vybraný jeden jedinec s dobrou hodnotou fitness. Táto situácia nastáva z toho dôvodu, že ak by bol vždy vybraný najlepší jedinec, algoritmus by sa zameral na využite daného pod-priestoru a zanedbával by iné sľubné pod-priestory, ktoré môžu obsahovať globálne extrémy. V takomto prípade kedy je Leader striktne najlepší jedinec, algoritmus rýchlo konverguje do lokálnych extrémov.

3.2.2 Migrácia

Migrácia sa v tomto algoritme odráža od migrácie T3A verzie. Parametre *PRT* aj *step* sú adaptívne. Rovnako ako tomu je u T3A, aj tu prichádza na scénu nový parameter *Njumps*, ktorý hovorí o tom koľko skokov má Migrant vykonať, a teda Migrant sa neviaže na parameter *pathLength* ako tomu bolo v pôvodnej verzii SOMA. Parametre *PRT* a *step* sa počítajú na základe rovníc č. 7 a 8. Tieto dva parametre zabezpečia diverzitu populácie a napomáhajú jedincom od uviaznutia. [36]

$$PRT = 0{,}5 + 0{,}45cos\left(T_1\pi\frac{FEs}{MaxFEs} + \pi\right) \tag{7}$$

$$step = 0{,}35 + 0{,}15cos\left(T_2\pi\frac{FEs}{MaxFEs}\right) \tag{8}$$

- *FEs* – aktuálny počet ohodnotení účelovej funkcie,
- *MaxFEs* – maximálny počet ohodnotení účelovej funkcie,

- T_1, T_2 – kontrolné parametre, autor neuvádza význam vplyvu. Odporúčaná hodnota autorom je $T_1 = T_2 = 1$. [36]

Neskoršie úpravy algoritmu Pareto SOMA využili mechanizmy z T3A SOMA, a teda rovnice č. 7 a 8 boli mierne modifikované.

Ďalšou zmenou v tejto verzii je aj zmena výpočtu hodnoty *PRTVector*. V základnej verzii SOMA bolo ukázané, že *PRTVector* môže nadobúdať iba hodnoty 0 a 1. V Pareto verzii sa tento parameter adaptuje každým migračným kolom a to na základe rovnice č. 9. [36]

$$PRTVector_j = \begin{cases} 1, & ak\ rand_j < PRT \\ FES/MaxFES, & inak \end{cases} \qquad pre\ j = 1, 2, \dots, D \qquad (9)$$

3.2.3 Aktualizácia

Proces aktualizácie prebieha rovnako ako v klasickom algoritme SOMA. Jedná sa o rozhodovanie, či pohybujúci sa jedinec pri migrovaní ku Leadrovi našiel lepšiu pozíciu ako bola jeho pôvodná alebo nie. Vyhodnocovanie tohto stavu sa robí na základne fitness funkcie. V prípade, že bola nájdená lepšia pozícia, pohybujúci jedinec sa na túto pozíciu presunie. V prípade, že nie, tak v danom migračnom kole nedošlo k vylepšeniu.

3.2.4 Pseudokód

Pre lepšiu predstavu ako algoritmus funguje je nižšie vyobrazený Pseudokód 3, ktorý popisuje základné funkcionality algoritmu Pareto SOMA.

Vstupy:
- Náhodne vygenerovaná populácia na základe *specimena*
- Ohodnotenie jedincov v počiatočnej populácii
- Nastavenie vstupných parametrov

pokiaľ ukončovacie podmienky nie sú splnené **rob:**
aktualizuj *PRT* a *step*
zotrieď populáciu
vyber *Leadra* a *Migranta* na základe obrázku č.15
Migrant skáče k *Leadrovi*
kontrola hraníc prehľadávaného priestoru
prepočítanie fitness funkcie
aktualizácia lepšej pozície *Migranta*

Výstup: Najlepšie nájdené riešenie.

Pseudokód 3: Algoritmus Pareto SOMA

3.3 ESP – SOMA

ESP – SOMA bola rovnako ako predchádzajúce modifikácie publikovaná v roku 2019 [37] a testovaná v rámci CEC 2019 100 digits competition, kde získala 18. miesto z celkového počtu 38 algoritmov.

ESP – SOMA je inšpirovaná známym optimalizačným algoritmom, a to Diferenciálnou Evolúciou. Konkrétne sa inšpirovala jeho modifikáciou EPSDE (Ensemble of Mutation and Crossover Strategies and Parameters in Differential Evolution) [38], ktorá preukázala, že je robustnejšia ako jej predchodcovia.

3.3.1 Inicializácia

Znova ako v klasickej verzii algoritmu SOMA, aj tu sa náhodne vygeneruje populácia podľa specimena a rovnice č. 2 v podkapitole 2.1.1.1. Veľkosť populácie určená vstupným parametrom *popSize.* Parameter *D* je daný dimenziou problému a jednotlivé zložky vektoru jedinca ležia v hraniciach prehľadávaného priestoru.

Spolu s týmito parametrami musia byť nastavené aj ďalšie novo vyskytujúce sa vstupné parametre. Parametre, ktoré treba nastaviť sú nasledujúce:

- **Refreshing *gap*** (obnovujúca medzera) – maximálny počet neúspešných iterácii. Neúspešná iterácia je taká, kedy pri migrácii celej populácii nedôjde k zlepšeniu čiastočného riešenia. V prípade, že jedinec dosiahne túto hodnotu, je nutné nastaviť na novo parametre *PRT* a výber stratégie (*strategy*). Odporúčané hodnoty tohto parametra sú v rozmedzí celých čísel <7, 20>. [37]
- **Zastavujúce kritéria** – medzi zastavujúce kritéria patrí *MAX_ITERATION* maximálny počet iterácii alebo *maxFES* maximálny počet ohodnotení hodnotiacou funkciou.
- ***step* a *pathLength*** – predstavujú rovnaké parametre ako v základnej verzii SOMA a ich odporúčané nastavenie je na nasledujúce hodnoty: *step* = 0,31 a *pathLength* = 3,0.
- ***adaptivePrt*** – môže nadobúdať hodnotu 0 alebo 1. Ak parameter *adaptivePrt* = 0, tak všetci jedinci majú parameter *PRT* striktne pevný a nastavený na hodnotu 0,3. Avšak ak *adaptivePrt* = 1, tak potom každý jedinec svoju vlastnú hodnotu *PRT* môže adaptovať. [37]

- **Ak *adaptivePrt* = 1** – ak hodnota nového parametra *adaptivePrt* je rovná hodnote 1, potom každý jedinec si náhodne zvolí svoju vlastnú *PRT* hodnotu z množiny nasledujúcich hodnôt {0,1; 0,3; 0,5; 0,7; 0,9}. Rozloženie možných hodnôt by malo byť čo najjednotnejšie pre celú populáciu. [37]
- **Zvolenie stratégie jedincom (*strategy*)** - každý jedinec x si náhodne vyberá svoju vlastnú stratégiu a to zo stratégii: All To One, All To All alebo All To Random. [37] Tieto stratégie sú už popísané v kapitole 2.2 ako varianty pôvodného algoritmu SOMA.

3.3.2 Migrácia

Pohyb jednotlivých jedincov sa vykonáva rovnako ako v klasickej verzii SOMA, a to na základe rovnice č. 4, ktorá bola symbolicky zapísaná v podkapitole 2.1.1.3.

3.3.3 Adaptácia

Adaptácia parametrov spočíva v tom, že ak väčšina jedincov má konkrétne nastavenia (hovoríme o parametroch *PRT* a *strategy*), tak takéto nastavenie môže byť pravdepodobne optimálne. Z tohto dôvodu sa využíva pravidlo rulety v prípade, že jedinec si musí zvoliť nové nastavenia. [37] Je samozrejmosťou, že po nastavení nových parametrov je počítadlo neúspešných iterácii vynulované.

Môže však nastať situácia, kedy jedinec dosiahne maximálny počet neúspešných iterácii (parameter *gap*), takýto jedinec prechádza do adaptačnej fázy a parametre *PRT* a výber stratégie sa musí nanovo zvoliť.

Následne $PRTVector_{i,j}$ sa počíta podľa nasledujúcej rovnice 10:

$$PRTVector_{i,j} = \begin{cases} 1, & ak\ rand_j < PRT_i \\ 0, & inak \end{cases} \qquad pre\ i = 1, 2, \dots, N \tag{10}$$

3.3.4 Pseudokód

Nižšie uvedený Pseudokód 4 popisuje funkcionalitu algoritmu ESP – SOMA.

Vstupy:

- Nastav veľkosť populácie, D a MAX_ITERATION
- Nastav *gap* = 7, *pathLength* = 3.0, *step* = 0.31 a *adaptivePrt* = 1

pre každého jedinca v populácii **rob:**
ak *adaptivePrt* == 0 **potom:**
PRT = 0.3
inak
PRT = náhodne vyber z {0.1, 0.3, 0.5, 0.7, 0.9}
strategy = náhodne vyber z {AllToOne, AllToAll, AllToRandom}
counter = 0

pokiaľ *iterácia* < *max_iteration* **rob:**
pre každého jedinca v populácii **rob:**
ak *strategy* == AllToOne **potom:**
Leader = nastav najlepšie riešenie x
inak ak *strategy* == AllToRandom **potom:**
Leader = nastav náhodné riešenie x
inak ak *strategy* == AllToAll **potom:**
P = celá populácia x
Leader = {P} − x
pre *t* = *step* **až** *pathLength* **s** *t* += *step* **rob:**
vygeneruj *PRTVector*
jedinec migruje k Leadrovi
ulož najlepšie nájdené riešenie do novej populácie
ak jedinec sa nezlepšil **tak:**
counter += 1
ak *counter* > *gap* **tak:**
counter = 0
vyber *PRT* pomocou rulety
vyber *strategy* pomocou rulety
ulož najlepšie nájdené riešenie

Výstup: Najlepšie nájdené riešenie.

Pseudokód 4: Algoritmus ESP SOMA

3.4 SOMA – CL

SOMA – CL bola publikovaná v roku 2020 [39]. Ako jediná modifikácia algoritmu SOMA bola testovaná v rámci CEC 2020 100 digits competition, kde sa umiestnila na 8. mieste z celkového počtu 11.

Táto modifikácia algoritmu SOMA kombinuje dve rôzne migračné stratégie. Jedna je klasická All To Random, ktorá slúži hlavne na prieskum prehľadávaného priestoru a druhou stratégiou je nová All To Cluster Leaders, ktorá slúži predovšetkým na prehľadanie potencionálnych oblastí, kde by sa mohlo nachádzať globálne optimum.

3.4.1 Inicializácia

Inicializácia počiatočnej populácie sa vykonáva rovnako ako u klasickej verzie algoritmu SOMA. Nižšie je uvedený zoznam parametrov, ktoré musia byť nastavené vo fáze inicializácie:

- **Zastavujúce kritéria –** *maxFES* alebo *MAX_ITERATION*, čiže maximálny počet ohodnotení účelovej funkcie alebo maximálny počet iterácií algoritmu.
- ***step*** a ***pathLength*** sú nastavené na hodnoty 0,33 a 3,0 ako sa odporúča aj v klasickej verzii algoritmu SOMA [31]. Tieto parametre sa nastavujú pre prvú stratégiu a teda pre All To Random.
- ***PRT*** je nastavený na hodnotu 0,5
- $\boldsymbol{NP_L}$ je počet zhlukov a počíta sa ako 10% z celkovej veľkosti populácie ***NP***. Táto hodnota je nastavená na základe početných experimentov. [39] Každý zhluk má pri tom jedného Leadra.
- $\boldsymbol{step_L}$ a $\boldsymbol{pathLength_L}$ sú hodnoty nastavené na 0,11 a 2,0. Tieto parametre platia pre druhú stratégiu All To Cluster Leaders. V tomto prípade je treba spomenúť aj parameter $\boldsymbol{PRT_L}$ nastavený na hodnotu 0.3. [39]

3.4.2 Princípy algoritmu

All To Random stratégia, slúži hlavne na prieskum prehľadávaného priestoru. Leader sa volí náhodne pre každého jedného jedinca. Po zvolení Leadra pre jedinca, jedinec migruje k Leadrovi na základe rovnice z pôvodnej verzie SOMA, a to na základe rovnice č. 4. Rozdielom klasickej verzie algoritmu SOMA je ten, že všetky partikulárne riešenia nájdené počas migrácie jedinca sa ukladajú do pamäte *M.* [39] Táto pamäť je neskôr použitá, ako sa dozviete v nasledujúcom texte.

Identifikácia zhlukov Leadrov. V jednej iterácii cez celú populáciu sa ukladajú do pamäte *M* všetky nájdené riešenia jedincov počas ich prehľadávania priestoru. Cieľom prečo sa tieto riešenia ukladajú je, že sa algoritmus bude snažiť v tejto pamäti nájsť sľubné riešenia a použiť ich ako kandidátov na Leadrov do ďalšej stratégie. Celý tento proces sa odohráva v dvoch krokoch.

V prvom kroku sa riešenia rozdeľujú do zhlukov na základe polohy v prehľadávanom priestore pomocou metódy *k*-means. Riešenia z pamäte *M* sú tým pádom rozdelené do NP_L

zhlukov. [39] Algoritmus *k*-means je pomerne jednoduchý a prebieha v niekoľkých krokoch, ktoré sú nasledujúce:

1. Náhodne z populácie je zvolených *k* centroidov.
2. Opakuje sa pokiaľ nie sú splnené ukončujúce podmienky:
    a. Priradenie každého jedinca z populácie ku centroidu, ku ktorému je najbližšie.
    b. Prepočítanie centroidov na základe priemeru.
3. Algoritmus končí v momente, ak sa splní jedna z nasledujúcich podmienok:
    a. Počet iterácií resp. zmien centroidov.
    b. Zmena centroidov už je zanedbateľná.
    c. Ak už nenastáva zmena priradenia jedincov ku centroidom.

V druhom kroku, len najlepší jedinci v daných zhlukoch sú zvolení za Leadrov. Následne sa zoradia od najlepšieho nájdeného riešenia po najhoršie. [39] Zoradené riešenia sa následne používajú v nasledujúcej stratégii All To Cluster Leader.

All To Cluster Leader stratégia predstavuje hybridizáciu All To One a All To Random, avšak Leader x_L nie je vybraný náhodne, ale práve zo zhlukov pomocou techniky Rank Selection (riešenie s najlepšou hodnotou má najväčšiu pravdepodobnosť, že bude zvolené za Leadra a práve riešenie s najhoršou hodnotou má najmenšiu pravdepodobnosť). [39] Migrácia prebieha ako v klasickej stratégii algoritmu SOMA podľa rovnice č. 4 okrem výberu Leadra, ktorý sa vyberá pre každého jedinca. Následne riešenie x_i ďalej migruje a najlepšie nájdené riešenie na t-ej pozícii sa presunie do novej iterácie algoritmu. Parameter *t* sa generuje z rozsahu 0 až $pathLength_L$ a spolu s veľkosťou kroku $step_L$. $PRTVector_j$ sa následne generuje rovnako ako v základnej verzii SOMA podľa rovnice č. 3 ale s tým rozdielom, že namiesto *PRT* je využitý PRT_L parameter.

Po skončení jednej iterácie tejto stratégie sa zopakuje celý proces odznova. Znamená to, že sa prechádza do ďalšej iterácie stratégie All To Random, pričom pamäť *M* je vymazaná a nanovo naplnená. Následne sa vytvoria zhluky a prechádza sa do druhej stratégie All To Cluster Leader. Tento proces sa opakuje až kým nie sú splnené ukončovacie podmienky.

3.4.3 Pseudokód

Nižšie uvedený Pseudokód 5 zobrazuje funkcionalitu popisovaného algoritmu SOMA – CL.

Vstupy:

- Nastav NP, NP_L, D a $maxFES$
- Nastav $pathLength$, $step$ a PRT
- Nastav $pathLength_L$, $step_L$ a PRT_L

Pokiaľ nie sú splnené zastavujúce kritéria **rob:**
  $M = \{\}$
  pre $i = 1$ **až** NP **rob:**
    $Leader$ = nastav náhodné riešenie x
    pre $t = 0$ **až** $pathLength$ **s** $t \mathrel{+}= step$ **rob:**
      vygeneruj $PRTVector$
      migruj jedinca k Leadrovi
      ulož každé ohodnotené riešenie do M

  použi metódu k-means clustering pre riešenia uložené v M
  nechaj len najlepšie riešenie z každého zhluku
  zotrieď zvyšné riešenia v M
  pre $i = 1$ **až** NP **rob:**
    x_L = použi rank selection na zhluk leadrov
    pre $t = 0$ **až** $pathLength_L$ **s** $t \mathrel{+}= step_L$ **rob:**
      vygeneruj $PRTVector$
      migruj jedinca k Leadrovi
  ulož najlepšie riešenie

Výstup: Najlepšie nájdené riešenie.

Pseudokód 5: Algoritmus SOMA-CL

4 TESTOVANIE

EA sú typické svojou stochastičnosťou. Cieľom týchto algoritmov je čo najviac sa priblížiť ku globálnemu optimu danej účelovej funkcie. Pre vyhodnotenie sa používa práve odchýlka od tohto optima. K tomu, aby sa dal výkon algoritmu zhodnotiť je potrebné daný algoritmus spustiť viac krát. Následne vo vyhodnotení sa zameriava na štatistické veličiny ako je napríklad minimum, maximum, priemer či medián. Na porovnávanie výkonnosti či robustnosti algoritmov sa využívajú hlavne dva prístupy, a to porovnanie už s existujúcimi vyriešenými problémami alebo porovnanie na množine testovacích funkcií, ktoré tiež nazývame ako benchmarkové funkcie.

Prvým spôsobom vyhodnotenia výkonnosti je porovnanie výsledkov dvoch algoritmov spustených na jeden optimalizačný problém. Princíp spočíva v tom, že optimalizačný problém už bol vyriešený iným algoritmom a sú známe výsledky. Na optimalizačný problém sa pustí algoritmus, ktorý následne pôjde do porovnania. Na záver sú zhodnotené a porovnané získané výsledky už s existujúcimi.

Druhým spôsobom vyhodnotenia výkonnosti či robustnosti algoritmu sú testovacie funkcie. Vyznačujú sa tým, že majú známu polohu globálneho extrému a preto sú vhodné na určenie funkčnosti evolučných algoritmov. Princípom je, že algoritmus je pustený na určitý benchmark, a teda na množinu testovacích funkcií kde je hodnotená vzdialenosť od globálneho extrému pre jednotlivé funkcie. Výsledky sú vyhodnotené na základe štatistických charakteristík. V testovacích funkciách existujú dva spôsoby akými sa pristupuje k vyhodnocovaniu:

- **Prístup založený na cieli (target based) –** tento prístup spočíva vo vyhodnotení, koľkokrát EA našiel dielčí cieľ a prípadne s akou presnosťou.
- **Prístup založený na maximálnom prípustnom počte ohodnotení testovacej funkcie (budget based)** – v tomto prístupe sa objavuje maximálny limit počtu ohodnotení účelovej funkcie.

Výhodou testovacích funkcií je, že sú známe analytické vzťahy a teda u väčšiny týchto funkcií je možné jednoducho vypočítať pozíciu a hodnotu globálneho extrému pre ľubovoľnú dimenziu. Každá testovacia inštancia sa zvyčajne opakuje 30-51 krát. [9]

II. PRAKTICKÁ ČASŤ

5 ADAPTÍVNE MECHANIZMY V MODERNÝCH VARIANTÁCH SOMA

Pred samotným návrhom adaptívnych mechanizmov boli preskúmané už existujúce adaptívne techniky v moderných variantách algoritmu SOMA. V tejto časti práce boli skúmané vplyvy existujúcich adaptívnych mechanizmov, ktoré sú popísané v nasledujúcich podkapitolách.

5.1 Proces organizácie

Proces organizácie hrá dôležitú rolu v objavovaní a preskúmaní sľubných pod-priestorov. V základných verziách algoritmu SOMA sa všetci jedinci pohybujú k Leadrovi. V jednotlivých moderných verziách sa proces organizácie modifikoval nasledovne:

- V **T3A** – proces organizácie využíva kombináciu náhodného výberu a výberu najlepších jedincov zo subpopulácií. Detailný popis je uvedený v kapitole 3.1.1.
- V **Pareto** – proces organizácie kombinuje náhodný výber a pareto princíp pre organizáciu populácie. Celý proces je uvedený v kapitole 3.2.1.

5.2 Adaptívne riadiace parametre

- **Parameter *PRT*** – *PRT* parameter ovplyvňuje smer jedinca, ktorý migruje k Leadrovi. Ak *PRT* sa blíži k 1, tak jedinec má tendenciu sa pohybovať priamo k Leadrovi, avšak ak *PRT* sa blíži k hodnote 0, tak jedinec má tendenciu prieskumu okolia. Navyše v základných verziách je parameter *PRT* pevne daný na začiatku algoritmu a je nemenný. Dve modifikácie tohto parametra sú zadefinované v kapitolách 3.1.2 a 3.2.2, kedy sa parameter *PRT* počíta na základe *FEs* a *MaxFEs*, či periodickej funkcie kosínus podľa rovníc 5 a 7.
- **Parameter *step* –** Parameter *step* udáva dĺžku kroku, ktorým sa jedinec pohybuje k Leadrovi. Rovnako ako *PRT* aj parameter *step* je v klasických verziách algoritmu SOMA pevne daný parameter na začiatku algoritmu. Hodnota tohto parametra vplýva na počet krokov, ktoré vykoná jedinec kým príde k Leadrovi, resp. kým dosiahne dĺžku trasy *pathLength*. Každý jeden doskok znamená nové možné riešenie. Vo vyššie spomínaných modifikáciách (kapitola 3.1.2 a 3.2.2) sa parameter *step* prepočítava v každom migračnom kole na základe hodnôt *FEs*, *MaxFEs*, či periodickej funkcie kosínus podľa implementácie rovníc 6 a 8.

- **Parameter *Njumps* –** Nový parameter, ktorý sa v klasických verziách algoritmu SOMA nenachádza. V modifikáciách nahrádza parameter *pathLength. Njumps* uvádza počet skokov, ktorý ma jedinec skákajúci k Leadrovi vykonať. [35, 36]
- **Parameter *PRTVector*** – Ďalším adaptívnym parametrom objavujúcim sa v moderných variantách algoritmu SOMA je výpočet parametra *PRTVector.* V klasických verziách môže nadobúdať len hodnotu 1 alebo 0, pričom tento vektor udáva smer jedinca, ktorý sa pohybuje k Leadrovi. V adaptívnej verzii môže nadobúdať hodnotu 1 alebo *FEs*/*MaxFEs*, ako je vyobrazené v rovnici 9. [36] Detailnejšie informácie sa nachádzajú v kapitole 3.2.2.
- **Parameter *adaptivePrt*** – Je taktiež novým parametrom, ktorý sa v základných verziách algoritmu SOMA neobjavuje. Môže nadobúdať hodnoty 0 alebo 1. Ak *adaptivePrt* = 0, tak jedinci v populácii majú hodnotu parametra *PRT,* takú aká bola nastavená na začiatku a je nemenná. Ak hodnota *adaptivePrt* = 1, v takom prípade každý jedinec z populácie si náhodne vyberie svoj vlastný *PRT* parameter a to z nasledujúcej množiny {0,1; 0,3; 0,5; 0,7; 0,9}. Hodnota tohto parametra je na začiatku nastavená na hodnotu 0. [37]
- **Parameter *strategy*** – Parameter *strategy* sa objavuje v ESP – SOMA. Jedná sa o to, že každý jedinec si náhodne zvolí svoju vlastnú stratégiu, akou bude migrovať. Volí si zo stratégii All To All, All To One alebo All To Random. [37]
- ***Clustering*** – Clustering sa objavuje vo verzii SOMA – CL, ktorej princípy boli detailne popísané v kapitole 3.4.2. SOMA – CL funguje akoby na 2 etapy. V prvej etape je využitá základná verzia algoritmu SOMA All To Random, s tým rozdielom, že sa ukladajú do pamäte *M* partikulárne riešenia nájdene počas migrácie jedincov k svojim Leadrom. [39] V druhej etape je využitá opäť verzia All To Random, pričom Leader je zvolený z určitej subpopulácie. SOMA – CL si osvojuje techniku identifikácie sľubných pod-priestorov, pričom zachováva diverzitu populácie.

6 VYBRANÉ ADAPTÍVNE STRATÉGIE V OSTATNÝCH METAHEURISIKÁCH

V tejto kapitole sú popísané vybrané moderné a zaujímavé techniky využité v iných metaheuristikách.

6.1 Adaptívnosť pomocou clusteringu

Jedným zo spôsobov využívania clusteringu v metaheuristikách je aplikácia clusteringu na počiatočnú populáciu ako tomu je u Diferenciálnej Evolúcie založenej na clusteringu (anglicky Cluster-Based Differential Evolution). Veľmi stručne povedané, počiatočná populácia sa na základe clusteringu rozdelí na k clusterov pomocou algoritmu k-means, ktorého funkčnosť bola spomenutá v kapitole 3.4.2. Takýmto spôsobom vznikne k clusterov. Potom optimalizačný proces DE prebieha zvlášť v každom clusteru. Tento návrh povolí detailné prehľadávanie v každom clusteru. Obdobným spôsobom bol clustering využitý napr. aj v algoritme PSO. [41]

6.2 Reštart populácie

Spôsobov reštartu populácie je niekoľko. Môže byť reštartovaná len časť populácie alebo celá. Tento adaptívny mechanizmus môže pridávať jedincov do populácie a následne odoberať.

Reštart populácie sa objavuje napríklad pri algoritme Diferenciálna Evolúcia s automatickým pridávaním nových jedincov (anglicky DE with automatic population injection). Jedná sa o to, že sa populácia počas behu zväčšuje a zmenšuje za určitých podmienok. Je daná počiatočná populácia NP, predefinovaná hodnota ζ, ktorá predstavuje toleranciu a hodnota Υ, ktorá predstavuje počet generácii, počas ktorých má nastať zlepšenie. Ak počas Υ generácii sa najlepší jedinec v populácii nezlepší s preddefinovanou toleranciou ζ, tak sa veľkosť populácie zväčší o NP_{add}, ktorá je náhodne vygenerovaná. Pre Ω generácií sa pustí algoritmus na novú populáciu o veľkosti NP_{add}. Následne dochádza k spojeniu pôvodnej populácie s novou, a teda vznikne nám nová populácia o veľkosti $NP_{new} = NP + NP_{add}$. Znova sa na s takto vytvorenou populáciou pustí algoritmus, ak dôjde k zlepšeniu, veľkosť populácie sa zmenší na pôvodnú veľkosť a to vybraním tých najlepších riešení, inak sa algoritmus opakuje a to tak, že ak počas Υ nenastane zlepšenie najlepšieho jedinca s preddefinovanou toleranciou ζ, tak sa populácia znova zväčšuje o novo náhodne vygenerovanú NP_{add}. [42]

Veľkosť populácie je redukovaná na pôvodnú veľkosť pri splnení jednej z týchto podmienok:

- Dôjde k zlepšeniu najlepšieho riešenia v populácii
- Populácia dosiahne preddefinovanú veľkosť populácie a najlepší jedinec v populácii sa počas Υ generácii nezlepšil
- Ak najlepšie nájdené riešenie dosiahne hodnotu globálneho optima. Môže sa uvažovať aj určitá tolerancia. Táto podmienka platí však len v prípade, ak je známe globálne optimum.

Je vhodné spomenúť, že veľkosť pridanej populácie NP_{add} je potrebné definovať pred začatím algoritmu. Rovnako medzi parametre, ktoré treba nastaviť na začiatku algoritmu patria aj parametre Υ, ζ a Ω.

7 POPIS NAVRHNUTÝCH MODIFIKÁCIÍ

Skôr než boli navrhnuté originálne modifikácie, bol skúmaný vplyv jednotlivých adaptívnych techník spomínaných v kapitole 5 a 6 na algoritmus SOMA verziu All To Random s rôznymi nastaveniami. All To Random ponúka vyváženejší stav medzi exploráciou (prehľadávanie celkového priestoru) a exploitáciou (lokálne prehľadávanie v okolí konkrétnych riešení). Na rozdiel od All To All, ktorá vykoná menej iterácií za rovnaký počet ohodnotení účelovej funkcie či All To One, ktorá môže mať tendenciu rýchlej konvergencie k lokálnemu optimu. Následne na základe štatistických hodnôt a Friedmanovho rankovacieho testu (anglicky Friedman Rank Test) bolo rozhodnuté o možných vhodných kombináciách adaptívnych stratégii. Viac informácií je uvedených v kapitole 10 o vykonaných experimentoch. Výsledkom kombinácií sú vybrané modifikácie popísane v nasledujúcich podkapitolách.

7.1 Modifikácia 1 – ESP / T3A *PRT*

Prvá modifikácia spočíva v implementovaní dvoch adaptívnych mechanizmov do základnej verzie algoritmu SOMA. Do základnej verzie bola zahrnutá možnosť pre každého jedinca výberu stratégie s akou bude migrovať a to z nasledujúcej množiny {*All To All*, *All To One*, *All To Random*}. Výber stratégie je pomocou ruletového pravidla, ktoré funguje na princípe proporcionálneho výberu. Na začiatku má každý jedinec priradenú stratégiu náhodne no najlepšie s rovnomerným rozdelením.

Okrem iného *PRT* parameter, ktorý je v základnej verzii nastavený napevno a to pred začatím samotného algoritmu, modifikácia implementuje výpočet parametra *PRT* podľa T3A verzie na základe rovnice č. 5 definovanej v kapitole 3.1.2.

Nasledujúci Pseudokód 6 popisuje funkcionality modifikácie 1.

Vstupy:
- Nastavenie vstupných parametrov pop_size, step, path_length, D, Max_FEs
- Náhodne vygeneruj počiatočnú populáciu na základe *specimena*
- Na základe rovnomerného rozdelenia priraď každému jedincovi jeho stratégiu z množiny strategy
- Ohodnotenie jedincov v počiatočnej populácii

pokiaľ ukončovacie podmienky nie sú splnené **rob:**
pre i = 1 **až** pop_size **rob:**
strategy$_i$ = ruletový výber stratégie z množiny strategy
aktualizuj *PRT* na základe rovnice **č. 17**
ak strategy$_i$ == AllToOne **rob:**
Leader = vyber najlepšieho jedinca z populácie x
inak ak strategy$_i$ == AllToAll **rob:**
P = cela populácia x
Leader = {P} − x$_i$
inak ak strategy$_i$ == AllToRandom **rob:**
Leader = vyber náhodne jedinca z populácie

pre *t* = *step* **až** *path_length* **s** *t* += *step* **rob:**
vygeneruj PRTVector$_i$ podľa rovnice 11
migruj x_i ku *Leadrovi* podľa rovnice 10
ulož najlepšie x_i do novej populácie

Výstup: Najlepšie nájdené riešenie.

Pseudokód 6: Modifikácia algoritmu SOMA č. 1

7.2 Modifikácia 2 - Reštart populácie / T3A *PRT*

Modifikácia č. 2 spočíva v tom, že do základnej verzie SOMA All To Random sa implementoval adaptívny parameter *PRT* vypočítaný na základe rovnice č. 5. Okrem toho sa tu implementovala ďalšia adaptívna technika a to *reštart populácie*.

Reštart populácie je pomerne jednoduchý, na začiatok však treba nastaviť určité parametre. Okrem tých, ktoré sa nachádzajú v základnej verzii SOMA, treba nastaviť nasledujúce parametre:

- NP_{add} – veľkosť pridávanej populácie v prípade, že nedôjde k zlepšeniu s určitou error_tolerance.
- *genToImprove* – počet generácií, počas ktorých by malo prísť k zlepšeniu s určitou error_tolerance.
- *pop_add_generations* – počet generácii algoritmu, ktoré majú byť aplikované na pridanú populáciu avšak ešte pred jej pridaním k pôvodnej.
- *error_tolerance* – tolerancia chyby s akou je akceptované zlepšenie populácie.

- *max_pop_size* – maximálna veľkosť populácie, ktorú môže populácia dosiahnuť.

Celý proces reštartu je jednoduchý a spočíva v tom, že ak po počte *genToImprove* nedôjde k zlepšeniu s danou *error_tolerance*, k pôvodnej populácii sa pridáva nová populácia náhodne vygenerovaná NP_{add}, ktorá ešte pred samotným pridaním prejde počtom *pop_add_generations* generáciami a až následne sa pridá k pôvodnej populácii. Tento proces sa opakuje až kým nie sú splnené ukončovacie podmienky. K redukcii populácie na pôvodnú veľkosť dochádza za daných podmienok a to tak, že sa zo zväčšenej populácie vyberú len tí najlepší jedinci, tak aby nová populácia bola o veľkosti tej pôvodnej:

- počas počtu *genToImprove* došlo k zlepšeniu najlepšieho riešenia v aktuálnej populácii.
- Bol dosiahnutý *max_pop_size* a najlepší jedinec sa v populácii počas *genToImprove* nezlepšil.
- V prípade, že najlepšie nájdené riešenie dosiahne hodnotu globálneho optima. Môže byť braná do úvahy aj určitá tolerancia. Táto podmienka platí však len v prípade, ak je známe globálne optimum.

Nasledujúci Pseudokód 7 reprezentuje funkcionality novovzniknutej modifikácie.

Vstupy:

- Nastavenie vstupných parametrov pop_size, path_length, D, Max_FEs
- Náhodne vygeneruj počiatočnú populáciu na základe *specimena*
- Na základe rovnomerného rozdelenia priraď každému jedincovi jeho stratégiu z množiny strategy
- Ohodnotenie jedincov v počiatočnej populácii

pokiaľ ukončovacie podmienky nie sú splnené **rob:**
  pre i = 1 **až** pop_size **rob:**
    aktualizuj *PRT* na základe rovnice č. 17
    Leader = vyber náhodne jedinca z populácie
    pohyb jedinca k *Leadrovi*
    pre t = *step* **až** *path_length* **s** t += *step* **rob:**
      vygeneruj $PRTVector_i$ podľa rovnice 11
      migruj x_i ku *Leadrovi* podľa rovnice 10
    ulož najlepšie x_i do novej populácie
    zaznamenaj či došlo k zlepšeniu s danou toleranciou

  ak počas *genToImprove* nedošlo k zlepšeniu **rob**:
    vygeneruj náhodne populáciu o veľkosti NP_{add}
    aplikuj *pop_add_generations* základnú verziu SOMA AllToRandom na NP_{add}
    spoj pôvodnú populáciu $NP + NP_{add}$

  inak ak počas *genToImprove* došlo k zlepšeniu **alebo** *pop_size* == *max_pop_size* **rob**:
    redukuj veľkosť populácie na pôvodnú – výber najlepších riešení

Výstup: Najlepšie nájdené riešenie.

Pseudokód 7: Modifikácia algoritmu SOMA č. 2

7.3 Modifikácia 3 – ESP / T3A *PRT* + *step*

Tretia modifikácia je skoro rovnaká ako prvá, ale s tým rozdielom, že v tomto návrhu sa aj parameter *step* počíta podľa implementácie T3A SOMA, definovanej v rovnici č. 6 a nie je napevno daný ako tomu je v základnej verzii SOMA.

Funkcionalita navrhnutej modifikácie je popísaná v nasledujúcom Pseudokóde 8:

Vstupy:

- Nastavenie vstupných parametrov pop_size, path_length, D, Max_FEs
- Náhodne vygeneruj počiatočnú populáciu na základe *specimena*
- Na základe rovnomerného rozdelenia priraď každému jedincovi jeho stratégiu z množiny strategy
- Ohodnotenie jedincov v počiatočnej populácii

pokiaľ ukončovacie podmienky nie sú splnené **rob:**
  pre i = 1 **až** pop_size **rob:**
    $strategy_i$ = ruletový výber stratégie z množiny strategy
    aktualizuj *PRT* na základe rovnice č. 17
    aktualizuj *step* na základe rovnice č. 18
    ak $strategy_i$ == AllToOne **rob:**
      Leader = vyber najlepšieho jedinca z populácie x
    inak ak $strategy_i$ == AllToAll **rob:**
      P = cela populácia x
      Leader = {P} − x_i
    inak ak $strategy_i$ == AllToRandom **rob:**
      Leader = vyber náhodne jedinca z populácie

    pre *t* = *step* **až** *path_length* **s** *t* += *step* **rob:**
      vygeneruj $PRTVector_i$ podľa rovnice 11
      migruj x_i ku *Leadrovi* podľa rovnice 10
    ulož najlepšie x_i do novej populácie

Výstup: Najlepšie nájdené riešenie.

Pseudokód 8: Modifikácia algoritmu SOMA č. 3

7.4 Modifikácia 4 – Reštart populácie / T3A *PRT* + *step*

Odráža sa od modifikácie č. 2. Rozdielom je, že okrem *reštartu populácie* a výpočtu *PRT* podľa implementácie T3A SOMA, tak aj parameter *step* je počítaný podľa implementácie T3A SOMA na základe rovnice 6.

Nasledujúci Pseudokód 9 približuje funkcionality navrhnutej modifikácie.

Vstupy:

- Nastavenie vstupných parametrov pop_size, path_length, D, Max_FEs
- Náhodne vygeneruj počiatočnú populáciu na základe *specimena*
- Na základe rovnomerného rozdelenia priraď každému jedincovi jeho stratégiu z množiny strategy
- Ohodnotenie jedincov v počiatočnej populácii

pokiaľ ukončovacie podmienky nie sú splnené **rob:**
pre i = 1 **až** pop_size **rob:**
aktualizuj *PRT* na základe rovnice č. 17
aktualizuj *step* na základe rovnice č. 18
Leader = vyber náhodne jedinca z populácie
pohyb jedinca k *Leadrovi*
pre t = *step* **až** *path_length* **s** t += *step* **rob:**
vygeneruj $PRTVector_i$ podľa rovnice 11
migruj x_i ku *Leadrovi* podľa rovnice 10
ulož najlepšie x_i do novej populácie
zaznamenaj či došlo k zlepšeniu s danou toleranciou

ak počas *genToImprove* nedošlo k zlepšeniu **rob**:
vygeneruj náhodne populáciu o veľkosti NP_{add}
aplikuj *pop_add_generations* základnú verziu SOMA AllToRandom na NP_{add}
spoj pôvodnú populáciu $NP + NP_{add}$

inak ak počas *genToImprove* došlo k zlepšeniu **alebo** *pop_size* == *max_pop_size* **rob**:
redukuj veľkosť populácie na pôvodnú – výber najlepších riešení

Výstup: Najlepšie nájdené riešenie.

Pseudokód 9: Modifikácia algoritmu SOMA č. 4

8 IMPLEMENTÁCIA

Na implementáciu jednotlivých algoritmov bolo zvolené vývojové prostredie PyCharm 2020.2.4 a programovací jazyk Python 3.7. Python patrí medzi interpretované jazyky, ktoré sú ľahko použiteľné a zrozumiteľné. Niekedy je radený aj medzi tzv. skriptovacie jazyky. Avšak možnosti sú oveľa väčšie, keďže ponúka možnosť tvorby rozsiahlych a plnohodnotných aplikácii, aj napr. s grafickým užívateľským rozhraním. Pýši sa jednoduchou syntaxou a produktívnosťou z hľadiska rýchlosti písania programov. Využitie je veľmi široké ako napríklad v oblasti webových aplikácii, dátovej analýzy, administrácie na serveroch či v oblasti testovania aplikácii. Jednou z výhod je silný arzenál zbraní na spracovávanie dát a ich vizualizáciu. Z tohto dôvodu je vhodný na spracovanie veľkého množstva dát, strojové učenie či prediktívnu analytiku.

Pri implementácii boli použité knižnice ako:

- **Math –** knižnica, ktorá obsahuje rôzne matematické funkcie a konštanty. Funkcie tejto knižnice nie je možné použiť s komplexnými číslami, avšak túto možnosť ponúka knižnica cmath.
- **Numpy –** knižnica, ktorá sa používa na prácu s poľami. Je vhodná aj na prácu v doméne lineárnej algebry, Fourierovej transformácie a matíc. Numpy ponúka rýchlu prácu s vektormi.
- **Plotly –** knižnica grafov, ktorá je schopná vytvoriť interaktívne grafy. Ponúka veľké množstvo grafov ako napríklad čiarové, bodové, plošné, stĺpcové, histogramy, bublinové, viacosové a mnoho iných grafov.
- **Csv** – knižnica, ktorá slúži na čítanie a zapisovanie do csv súborov.

Ďalej je v programe implementovaná knižnica pre benchmark CEC2020. Vytvorenú knižnicu testovacích funkcií v Pythone poskytlo laboratórium umelej inteligencie na FAI UTB[1]. Detailnejšie informácie o tomto benchmarku nájdete v kapitole 9.

[1] https://github.com/TBU-AILab/Bison-Algorithm

Na analýzu, vyhodnotenie dát a Friedmanov rankový test bolo využité prostredie Wolfram Mathematica 12.2, ktorá poskytuje základné výpočtové prostredie pre pedagógov, študentov či mnoho ďalších ľudí. Použitie tohto prostredia je veľmi jednoduché. Výhodou je, že patrí medzi neustále sa rozširujúce systémy.

8.1 Štruktúra implementácie

V tejto podkapitole je stručne popísaná štruktúra implementácie. V hlavnej zložke SOMA sa nachádzajú podzložky a súbory, ako je vyobrazené na Obrázku 15 pod nasledujúcim popisom:

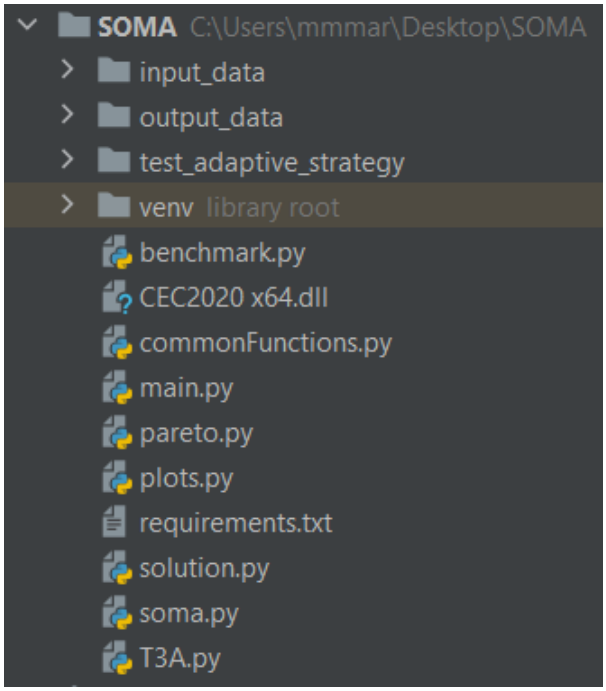
1. **Zložky:**

   - **input_data** – zložka obsahuje vstupné dáta vo forme textových súborov, ktoré sú potrebné pre správne fungovanie benchmarku. Zložka obsahuje tri druhy vstupných súborov a to matice rotácii v súboroch s názvami „*M_f_Dx.txt*“, kde f predstavuje číslo funkcie a x dimenziu, ďalej obsahuje vektory posunu v súboroch s názvami *shift_data.txt*“ a vektory potrebné pre hybridné funkcie v súboroch s názvami *shuffle_data.txt*. Detailné informácii o benchmarku CEC2020 sú zosumarizované v ďalšej kapitole 9.

   - **output_data** – obsahuje výstupné dáta s koncovkou *csv* pre jednotlivé algoritmy v hlavnej zložke SOMA, a to pre tri základné verzie algoritmu SOMA, pre algoritmus Pareto SOMA a T3A SOMA. Detailnejší popis hierarchie tejto zložky a obsahu jednotlivých súborov sa nachádza v ďalšej podkapitole 8.1.1

   - **test_adaptive_strategy** – obsahuje dva druhy súborov. Jedným druhom sú súbory s koncovkou *py*, pričom každý takýto súbor predstavuje jednu alebo kombináciu adaptívnych stratégií aplikovaných na algoritmus SOMA All To Random. Takto vzniknuté algoritmy slúžili na testovanie vplyvu jednotlivých stratégii na algoritmus a tiež pre testovanie rôznych kombinácii týchto stratégii a ich vplyvu. Rovnako sa tu nachádzajú aj výsledné modifikácie, ktoré boli popísané v kapitole 7. Detailnejšie informácie sa nachádzajú v neskôršej kapitole o experimentoch. Druhým druhom súborov sú súbory s koncovkou *csv*, ktoré sú zoskupené vo vnorenej zložke *test_data*. Jedná sa o výstupné dáta jednotlivých stratégií a kombinácii. Výstupné dáta majú rovnaký formát ako dáta popísane v kapitole 8.1.1

- **venv** – zložka, ktorá obsahuje inštaláciu Pythonu pre konkrétnu verziu Pythonu a veľké množstvo balíčkov a knižníc, ktoré sa v diplomovej práci využívajú.



Obrázok 15: Súborová štruktúra diplomovej práce

2. **Súbory:**

- **benchmark.py** – súbor, ktorý obsahuje metódy na prácu s knižnicou CEC2020 x64.dll. Hlavnou metódou je *cec20* so vstupnými parametrami vektor (jedinec z populácie), dimenzia (dimenzia problému) a čísla testovacej funkcie. Spomínaná metóda vracia hodnotu účelovej funkcie daného vektora na spomínanej dimenzii a testovacej funkcii.
- **CEC2020 x64.dll** – knižnica testovacích funkcií z CEC2020 poskytnutá laboratóriom umelej inteligencie na FAI UTB.
- **commonFunctions.py** – súbor obsahuje metódy, ktoré sú spoločné pre všetky algoritmy. Medzi tieto metódy patrí napríklad *generate_individual* (metóda, ktorá vygeneruje jedinca v rámci prehľadávaného priestoru preddefinovanej veľkosti a vráti jeho vektor), *create_and_evaluate_population* (metóda, ktorá vytvorí list preddefinovanej veľkosti a naplní ho riešeniami typu *Solution*, tento objekt v sebe obsahuje vektor jedinca a hodnotu účelovej funkcie) alebo *check_boundary* (metóda, ktorá zabezpečí neprekročenie hraníc prehľadávaného

priestoru novo vzniknutými jedincami) a mnoho ďalších potrebných metód pre všetky algoritmy.

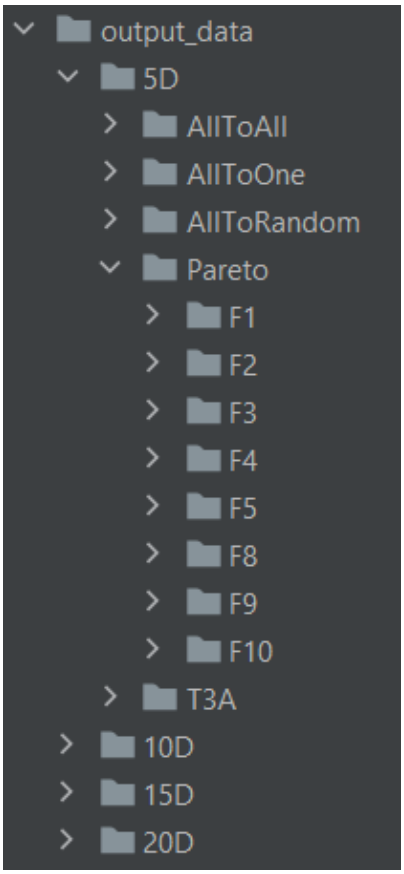
- **main.py** – obsahuje metódy na spustenie a ukladanie dát do súborov pre každý jeden vytvorený algoritmus, adaptívnu stratégiu alebo ich kombinácie. Taktiež obsahuje triedu *Main(),* ktorá tieto metódy volá už s preddefinovanými parametrami, s ktorými sa testy púšťali. Spustenie jednotlivých testov je veľmi jednoduché a to príkazom *python3 cesta_k_súboru_main/main.py.*
- **pareto.py** – súbor obsahuje implementáciu algoritmu Pareto SOMA.
- **plots.py** – súbor, ktorý obsahuje metódu pre načítanie dát zo súborov, ich spracovanie a následne vykreslenie do konvergenčných offline grafov.
- **requirements.txt** – textový súbor, ktorý obsahuje požiadavky na knižnice, ktoré sú potrebné pre správne fungovanie algoritmov.
- **solution.py** – obsahuje triedu *Solution()*, ktorá jedno riešenie daného algoritmu. Jej atribútmi sú *vector* (vektor predstavujúci polohu jedinca v priestore) a *cost_function* (hodnota účelovej funkcie daného jedinca). Okrem iného obsahuje get metódy.
- **soma.py** – súbor, ktorý obsahuje implementáciu troch základných verzií algoritmu SOMA a to All To All, All To One a All To Random.
- **T3A.py** – súbor obsahuje implementáciu algoritmu T3A SOMA.

8.1.1 Štruktúra zložky output_data

Zložka output_data obsahuje v sebe štyri podzložky, ktoré reprezentujú dimenziu. Následne v každej jednej dimenzii sa nachádzajú zložky reprezentujúce jednotlivé algoritmy. Do nich sú vnorené zložky jednotlivých testovacích funkcií očíslovaných od 1 po 10. V každej jednej takejto zložke sa nachádza 31 csv súborov. Tieto súbory obsahujú tzv. error-value, v slovenčine veľkosť chyby, a je to hodnota, ktorá reprezentuje rozdiel globálneho optima danej funkcie (globálne optimá jednotlivých testovacích sú známe a sú popísané v kapitole 9) a dosiahnutej hodnoty účelovej funkcie daného riešenia. V týchto zložkách sa nachádzajú 2 druhy csv súborov. Prvý druh obsahuje dva stĺpce. V prvom je hodnota *FES* a v druhom error-value, ktorá bola získaná pri danej hodnote *FES*. Takýto súbor teda obsahuje zoznam hodnôt error-value počas evolučného procesu algoritmu, pri čom je uchovávaná aj hodnota

FES, pri ktorej došlo k zmene (zníženiu) hodnoty error-value. Takýchto súborov je 30, pretože každý algoritmus sa pri testovaní opakoval 30 krát. Druhým druhom súborov je súbor, ktorý rovnako obsahuje dva stĺpce pričom prvý stĺpec vyjadruje číslo opakovania <1; 30> a druhý stĺpec obsahuje najnižšiu hodnotu error-value získanú v konkrétnom opakovaní daného algoritmu. Prvý druh súborov bol využívaný na tvorbu priemerných konvergenčných grafov pre dané funkcie, algoritmy a dimenzie. Druhý druh súborov bol využívaný pre štatistické účely a Friedmanov rankový test. Z dát z tohto súboru sa vykonávala štatistika, ktorú bola spracovávaná v prostredí Wolfram Mathematica. Štatistické hodnoty obsahujú minimum, maximum, priemer, medián a smerodajnú odchýlku na základe hodnôt z týchto súborov.

Obdobným spôsobom je robená aj hierarchia súborov a dát, ktoré sa nachádzajú v zložke *test_adaptive_strategy/test_data.* Na nasledujúcom Obrázku 16 môžete vidieť štruktúru zložky output_data.



Obrázok 16: Súborová štruktúra zložky output_data

9 TESTOVANIE NA BENCHMARKU CEC 2020

V nasledujúcich častiach diplomovej práce je popísaný vybraný benchmark, na ktorom prebiehalo testovanie. Jednou z výhodou použitia benchmarku je, že všetky funkcie sú upravené tak, aby mali spoločné hranice prehľadávaného priestoru. Tieto benchmarkové sady každý rok publikuje IEEE, celým názvom Institute of Electrical and Electronics Engineers (slovensky Inštitút pre elektrotechnické a elektronické inžinierstvo). Benchmarkové sady predstavené IEEE CEC, celým názvom Congress on Evolutionary Computation (slovensky Kongres o evolučných výpočtoch) sú celosvetovo uznávané a používané pre porovnávanie algoritmov. Z tohto plynie ďalšia výhoda použitia benchmarku a to tá, že výskumníci, študenti alebo vo všeobecnosti ľudia si môžu medzi sebou porovnávať algoritmy na základe dosiahnutých výsledkov v konkrétnom benchmarku. Pre testovanie bol zvolený benchmark IEEE CEC 2020, ktorý bol najnovší v období zadania diplomovej práce a realizácie praktickej časti.

9.1 Benchmark CEC 2020

V nasledujúcej Tabuľke 4 sa nachádza prehľad jednotlivých funkcií a ich globálne minimum a pod tabuľkou stručný popis prevzatý z dokumentácie ohľadne definícií problému a vyhodnocovacích kritérií pre CEC 2020 [40].

Tabuľka 4: Prehľad benchmarku CEC 2020

	Číslo	Funkcia	Minimum
Unimodálna	1	Bent Cigar funkcia s posunom a rotáciou	100
Základné	2	Schwefel funkcia s posunom a rotáciou	1100
	3	Lunacek bi-Rastrigin funkcia s posunom a rotáciou	700
	4	Rozšírená Rosenbrock plus Griewangk funkcia	1900
Hybridné	5	Hybridná funkcia 1 (N=3)	1700
	6	Hybridná funkcia 2 (N=4)	1600
	7	Hybridná funkcia 3 (N=5)	2100
Kompozitné	8	Kompozitná funkcia 1 (N=3)	2200
	9	Kompozitná funkcia 2 (N=4)	2400
	10	Kompozitná funkcia 3 (N=5)	2500

Všetky testovacie funkcie sú definované ako minimalizačný problém nasledovne:

$$Min\ f(x), x = [x_1, x_2, \dots, x_D]^T \text{ , kde} \tag{11}$$

D je počet dimenzií problému.

Funkcie sú navrhnuté tak, aby mali spoločný rozsah prehľadávaného priestoru, a teda v našom prípade to je v rozmedzí $[-100, 100]^D$.

$\boldsymbol{O_{i1} = [O_{i1}, O_{i2}, ..., O_{iD}]^T}$ je vektor posunu globálnych optím, ktoré sú definované v súbore „*shift_data_x.txt*“ a hodnoty sú náhodne vygenerované z $[-80, 80]^D$.

$\boldsymbol{M_i}$ je matica rotácie. Každá testovacia funkcia má svoju vlastnú maticu rotácie. Rovnako ako vektor posunu aj matice rotácie sú definované v súbore „*M_f_Dx.txt*“, kde f predstavuje číslo funkcie a x dimenziu.

Matematický zápis jednotlivých základných funkcií, ktoré sa využívajú v tomto benchmarku, buď samostatne alebo na skladanie hybridných či kompozitných funkcií, nájdete v prílohe PI.

9.1.1 Hybridné funkcie

Vzhľadom na to, že v reálnych optimalizačných problémoch, rôzne subkomponenty premenných môžu mať rôzne vlastnosti, tak z tohto dôvodu komponenty jedincov sa delia podľa pravdepodobnosti *p* na *n* segmentov. V každom jednom segmente sa potom volá iná základná funkcia. [40] Skladanie hybridnej funkcie funguje na základe nasledujúcej rovnice:

$$F(x) = g_1(M_1 z_1) + g_2(M_2 z_2) + \cdots + g_N(M_N z_N) + F^*(x) \text{ , kde} \quad (12)$$

- **F(x)**: výsledná hybridná funkcia,
- **g_i(x)**: i-ta základná funkcia využitá na konštrukciu hybridnej funkcie,
- **N**: počet základných funkcií,
- **z_i**: polohový vektor jedinca,
- **p_i**: pravdepodobnosť rozdelenia pôsobnosti – rozdelenie základných funkcií jednotlivým dimenziám. Na základe tejto pravdepodobnosti je každej funkcii potom pridelené n_i dimenzií v náhodnom poradí podľa vektoru v súboroch *shuffle_data.txt*. [40]

1. **Hybridná funkcia 1:**
   - **a.** N = 3
   - **b.** p = [0.3, 0.3, 0.4]
   - **c.** g_1: Modifikovaná funkcia Schwefel
   - **d.** g_2: Funkcia Rastrigin
   - **e.** g_3: Funkcia High Conditioned Elliptic

2. **Hybridná funkcia 2:**
   - **a.** N = 4
   - **b.** p = [0.2, 0.2, 0.3, 0.3]
   - **c.** g_1: Rozšírená funkcia Schaffer
   - **d.** g_2: Funkcia HGBat
   - **e.** g_3: Funkcia Rosenbrock
   - **f.** g_4: Modifikovaná funkcia Schwefel
3. **Hybridná funkcia 3:**
   - **a.** N = 5
   - **b.** p = [0.1, 0.2, 0.2, 0.2, 0.3]
   - **c.** g_1: Rozšírená funkcia Schaffer
   - **d.** g_2: Funkcia HGBat
   - **e.** g_3: Funkcia Rosenbrock
   - **f.** g_4: Modifikovaná funkcia Schwefel
   - **g.** g_5: Funkcia High Conditioned Elliptic

9.1.2 Kompozitné funkcie

Rovnako ako hybridné funkcie, tak aj kompozitné sa skladajú z viacerých funkcií. Rozdielom od hybridných je, že tieto funkcie sa prekrývajú vo všetkých dimenziách. [40] Kompozitné funkcie vznikajú na základe nasledujúcej rovnice:

$$F(x) = \sum_{i=1}^{N} \{\omega_i^* [\lambda_i g_i(x) + bias_i]\} + F^* \text{ , kde} \quad (13)$$

- **F(x)**: výsledná kompozitná funkcia,
- **g_i(x)**: i-ta základná funkcia využitá na konštrukciu kompozitnej funkcie,
- **N**: počet základných funkcií,
- **$bias_i$**: určuje, ktoré optimum je globálnym,
- **λ_i**: využíva sa kontrolu výšky každej základnej funkcie,
- **ω_i**: normalizovaná váha každej základnej funkcie, ktorá sa počíta podľa nasledujúcich rovníc:

$$\omega_i = \frac{w_i}{\sum_{i=1}^{n} w_i}, \quad (14)$$

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{D}(x_j - o_{ij})^2}} \exp\left(-\frac{\sum_{j=1}^{D}(x_j - o_{ij})^2}{2D\sigma_i^2}\right), \text{ kde} \quad (15)$$

- $\mathbf{o_i}$: nové posunuté optimum každej základnej funkcie,
- $\boldsymbol{\sigma_i}$: využíva sa kontrolu rozsahu pokrytia každej základnej funkcie.

1. **Kompozitná funkcia 1:**
   - **a.** N = 3
   - **b.** σ = [10, 20, 30]
   - **c.** λ = [10, 10, 1]
   - **d.** bias = [0, 100, 200]
   - **e.** g_1: Funkcia Rastrigin
   - **f.** g_2: Funkcia Griewank
   - **g.** g_3: Modifikovaná funkcia Schwefel
2. **Kompozitná funkcia 2:**
   - **a.** N = 4
   - **b.** σ = [10, 20, 30, 40]
   - **c.** λ = [10, 1e – 6, 10, 1]
   - **d.** bias = [0, 100, 200, 300]
   - **e.** g_1: Funkcia Ackley
   - **f.** g_2: Funkcia High Conditioned Elliptic
   - **g.** g_3: Funkcia Griewank
   - **h.** g_4: Funkcia Rastrigin
3. **Kompozitná funkcia 3:**
   - **a.** N = 5
   - **b.** σ = [10, 20, 30, 40, 50]
   - **c.** λ = [10, 1, 10, 1e – 6, 1]
   - **d.** bias = [0, 100, 200, 300, 400]
   - **e.** g_1: Funkcia Rastrigin
   - **f.** g_2: Funkcia Happycat
   - **g.** g_3: Funkcia Ackley
   - **h.** g_4: Funkcia Discus
   - **i.** g_5: Funkcia Rosenbrock

10 EXPERIMENTY

Hlavným cieľom praktickej časti diplomovej práce bolo navrhnúť vlastné modifikácie algoritmu SOMA, konkrétne návrh originálnych adaptívnych stratégií pre riadenie populačnej dynamiky a parametrov algoritmu. Prvým krokom bolo preskúmanie už existujúcich adaptívnych mechanizmov a zistiť ich v vplyv na algoritmus. Detailnejší popis jednotlivých moderných adaptívnych mechanizmov sa nachádza v kapitole 5. Následne v kapitole 6 sa nachádzajú moderné adaptívne mechanizmy využité v iných metaheuristikách, ktoré boli tiež aplikované na algoritmus SOMA. Nasledovalo mnoho testovaní, o ktorých sa dozviete neskôr. Výsledkom týchto testov bol návrh originálnych modifikácii, ktoré sú popísané v kapitole 7. Tieto modifikácie vznikli kombináciou adaptívnych mechanizmov, či návrhom čisto nových adaptívnych mechanizmov pre algoritmus SOMA na základe štatistických vyhodnotení. Posledným a dôležitým krokom bolo tieto novo vytvorené modifikácie implementovať a následne porovnať s vybranými verziami algoritmu SOMA. K porovnaniu som zvolila algoritmus SOMA s použitím variant All To One, All To One Random a All To All, ktoré sú popísane v kapitolách 2.2.1, 2.2.2 a 2.2.3. Okrem toho modifikácie sú porovnávané s Pareto SOMA a T3A SOMA, ktoré sú jedny z moderných modifikácií využívajúcich adaptívne stratégie. Tieto dve spomínané verzie sú popísané v kapitolách 3.1 a 3.2.

10.1 Experimenty s jednotlivými adaptívnymi stratégiami

Testovanie jednotlivých adaptívnych stratégii prebiehalo na nasledujúcich nastaveniach:

- ***Dimenzia***: 10D,
- ***MaxFES*****:** 10^6,
- ***pop_size***: 30.

a hodnoty parametrov algoritmu SOMA odporúčané ich autorom [31], v prípade ak boli potrebné:

- ***PRT*****:** 0,3,
- ***step***: 0,11,
- ***path_length***: 3,0.

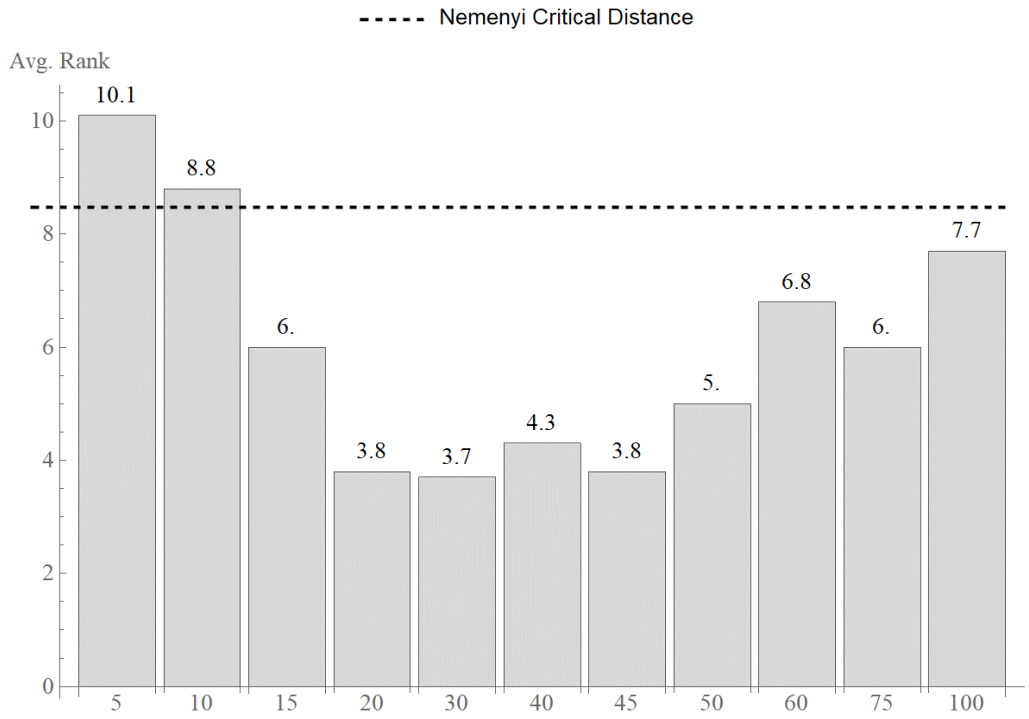
Hlavným motívom testovania bolo zistiť ako jednotlivé mechanizmy a ich nastavenia parametrov vplývajú na algoritmus SOMA. Pre testovacie účely bola zvolená varianta All

To Random. Získané dáta z testov boli následne štatisticky spracované a bol vykonaný Friedmanov rankový test, pokiaľ to bolo možné.

Freidmanov rankový test je neparametrický štatistický test, ktorý ako inak vyvinul Milton Friedman. Tento test sa používa na jednosmernú analýzu rozptylu podľa blokov opakovaných meraní. Blok s najnižším priemerným hodnotením dosahuje najlepšie výsledky, tieto výsledky sú však významné iba ak hodnota p (p-value) je pod hodnotou hladiny významnosti, v našom prípade 0,05. P-value vyjadruje najnižšiu hladinu významnosti testu, pri ktorej na daných dátach ešte zamietneme nulovú hypotézu. Platí teda, že čím je nižšia hodnota p-value testu, tým menšia je pravdepodobnosť, že platí nulová hypotéza. Nulová hypotéza v našom prípade znie: „Nulová hypotéza, že priemer je rovnaký je zamietnutá na hladine významnosti 5 percent na základe Friedmanovho rankového testu " v prípade ak hodnota p-value je nižšia ako 0,05 a teda dáta sú relevantné.

10.1.1 Experimentálna štúdia s parametrom *Njumps*

Rozdielom od klasického algoritmu SOMA je, že sa nepoužíva parameter *pathLength*, ale namiesto toho *Njumps*, ktorý ohovorí o tom koľko skokov má jedinec pri migrácii vykonať. Odporúčaná hodnota autora verzie SOMY, ktorá túto techniku obsahuje je hodnota 45. [35] Testy boli vykonávané s nasledujúcimi hodnotami parametra *Njumps* = {5, 10, 15, 20, 30, 40, 45, 50, 60, 75, 100}. Na Obrázku 17 je možné vidieť výsledky Friedmanovho rankového testu. Hodnota p-value bola 9,94E-08, teda nižšia ako hladina významnosti 5% a to znamená, že výsledky sú relevantné.



Obrázok 17: Výsledky Friedmanovho rankového testu pre parameter Njumps

Z predchádzajúceho výsledku je zrejmé, že nastavenie *Njumps* = 30 vedie výrazne lepšie ako nastavenie na hodnotu 5 a 10, ktoré sa už nachádzajú nad krivkou kritickej vzdialenosti. Odporúčaná hodnota autorom, ako už bolo spomínané, bola hodnota 45. Ako je z výsledku vidieť boli dosiahnuté podobné výsledky v blízkom okolí tejto hodnoty. Pre ďalšie testovacie účely bol volený parameter *Njumps* = 30 na základe výsledku Friedmanovho rankového testu, pretože dosiahlo toto nastavenie najnižšiu hodnotu a tiež na základe štatistického vyhodnotenia jednotlivých nastavení parametra *Njumps*, ktoré sa nachádza v prílohe PII: Štatistické vyhodnotenie adaptívnych stratégií.

10.1.2 Experimentálna štúdia s parametrom *step*

Parameter *step* sa dal počítať dvomi spôsobmi a to podľa algoritmu T3A SOMA alebo podľa Pareto SOMA. Popis výpočtu parametru step sa nachádza v kapitole 5.3. Keďže hodnota p-value bola vyššia ako hladina významnosti 5%, výsledky z Friedmanovho rankového testu nie sú relevantné a z toho dôvodu boli porovnávané výkonnosti týchto dvoch spôsobov výpočtu parametra *step* len na základe štatistických výsledkov.

Nasledujúca Tabuľka 5 predstavuje štatistické vyhodnotenie T3A a Pareto implementácie pre step. V tabuľke sa nachádza štatistické vyhodnotenie pre všetky testovacie funkcie. Tabuľka obsahuje štatistické hodnoty minimum, maximum, priemer, medián a štandardná odchýlka z najlepších nájdených riešení (najnižšie hodnoty error-value) počas 30tich

opakovaní algoritmu s daným nastavením. V každej tabuľke je zvýraznený najnižší získaný priemer. Na prvej funkcii dosiahlo lepšie výsledky parameter *step* podľa vzorca z algoritmu Pareto SOMA. Avšak celkovo T3A implementácia mala úspech 8/10, pričom Pareto implementácia len 2/10. Z globálneho hľadiska je pre rôzne problémy lepší T3A vzorec, ktorý bol neskôr používaný aj v kombináciách adaptívnych stratégii pri vzniku nových modifikácii algoritmu SOMA.

Tabuľka 5: Štatistické vyhodnotenie adaptívneho parametra step

Implementácia step	Min	Max	Priemer	Medián	Štandardná odchýlka
F1					
T3A	0.00E+00	5.32E+02	4.88E+01	1.00E-08	1.49E+02
Pareto	0.00E+00	1.00E-08	**1.00E-09**	0.00E+00	3.05E-09
F2					
T3A	1.81E-01	1.36E+01	**5.30E+00**	3.91E+00	3.84E+00
Pareto	1.78E-01	5.02E+01	1.53E+01	1.54E+01	1.07E+01
F3					
T3A	1.04E+01	1.21E+01	**1.10E+01**	1.08E+01	4.57E-01
Pareto	1.04E+01	1.29E+01	1.14E+01	1.14E+01	6.71E-01
F4					
T3A	3.68E-01	3.19E+00	1.21E+00	1.13E+00	7.27E-01
Pareto	5.01E-02	1.57E+00	**6.49E-01**	5.47E-01	4.11E-01
F5					
T3A	1.00E-02	6.70E+01	**1.49E+01**	4.27E+00	2.00E+01
Pareto	5.01E-04	1.59E+02	2.14E+01	2.46E+00	4.04E+01
F6					
T3A	3.04E-02	1.06E+00	**2.53E-01**	2.51E-01	2.34E-01
Pareto	2.88E-02	1.20E+02	2.18E+01	5.23E-01	4.49E+01
F7					
T3A	6.50E-06	6.24E-01	**1.38E-01**	2.80E-03	1.86E-01
Pareto	1.71E-03	5.85E+01	9.18E+00	4.74E-01	1.54E+01
F8					
T3A	3.47E+01	1.02E+02	**9.55E+01**	1.01E+02	1.43E+01
Pareto	6.59E+01	1.01E+02	9.84E+01	1.00E+02	7.95E+00
F9					
T3A	1.00E+02	3.42E+02	**2.85E+02**	3.36E+02	7.86E+01
Pareto	1.00E+02	3.42E+02	3.02E+02	3.37E+02	8.00E+01
F10					
T3A	3.98E+02	4.45E+02	**4.21E+02**	4.26E+02	2.13E+01
Pareto	3.98E+02	4.47E+02	4.32E+02	4.44E+02	2.03E+01

10.1.3 Experimentálna štúdia s parametrom *PRT*

Rovnako ako tomu bolo pri parametru *step*, aj tento parameter sa objavoval v moderných modifikáciách ako adaptívny a počítal sa dvomi spôsobmi podľa T3A a Pareto. Tieto

spôsoby už boli popísané vyššie v kapitole kde sa preberali moderné adaptívne stratégie. Keďže hodnota p-value bola opäť vyššia ako hladina významnosti 5%, výsledky z Friedmanovho rankového testu nie sú relevantné a z toho dôvodu boli porovnávanie výkonnosti týchto dvoch spôsobov výpočtu parametra *PRT* len na základe štatistických výsledkov.

Tabuľka 6: Štatistické vyhodnotenie adaptívneho parametra PRT

Implementácia PRT	Min	Max	Priemer	Medián	Štandardná odchýlka
F1					
T3A	0.00E+00	2.00E-08	**7.66E-09**	1.00E-08	5.04E-09
Pareto	0.00E+00	1.00E-08	8.66E-09	1.00E-08	3.45E-09
F2					
T3A	1.91E-01	2.08E+01	**7.16E+00**	6.76E+00	5.43E+00
Pareto	1.87E-01	4.52E+01	1.16E+01	8.72E+00	1.02E+01
F3					
T3A	1.05E+01	1.31E+01	**1.15E+01**	1.14E+01	6.79E-01
Pareto	1.04E+01	1.32E+01	1.16E+01	1.14E+01	7.12E-01
F4					
T3A	4.19E-01	2.10E+00	1.13E+00	9.73E-01	4.91E-01
Pareto	1.56E-01	2.07E+00	**8.19E-01**	6.79E-01	4.99E-01
F5					
T3A	1.97E-01	2.35E+01	**4.99E+00**	1.44E+00	6.14E+00
Pareto	1.99E-01	3.90E+01	7.77E+00	2.67E+00	8.70E+00
F6					
T3A	2.93E-02	9.01E-01	**1.93E-01**	2.32E-01	1.83E-01
Pareto	2.42E-02	6.03E-01	2.71E-01	2.74E-01	1.64E-01
F7					
T3A	8.30E-06	6.23E-01	**1.01E-01**	2.83E-03	1.79E-01
Pareto	1.17E-05	6.25E-01	1.69E-01	1.64E-01	1.88E-01
F8					
T3A	3.00E-08	1.02E+02	6.76E+01	8.70E+01	3.61E+01
Pareto	1.10E-07	1.02E+02	**5.91E+01**	4.95E+01	3.71E+01
F9					
T3A	1.00E+02	3.38E+02	**1.26E+02**	1.00E+02	6.47E+01
Pareto	1.30E-07	3.40E+02	1.34E+02	1.00E+02	8.03E+01
F10					
T3A	3.98E+02	4.44E+02	**4.02E+02**	4.00E+02	9.95E+00
Pareto	3.98E+02	4.44E+02	4.02E+02	3.99E+02	1.14E+01

Analýza štatistických hodnôt sa vykonávala rovnako ako tomu bolo u parametru *step*. Zvýraznená je priemerná hodnota pre dané testovacie funkcie. Z Tabuľky 6 je zrejmé, že opäť lepším spôsobom pre výpočet parametra *PRT* je spôsob použitý v algoritme T3A SOMA s úspešnosťou 8/10.

10.1.4 Experimentálna štúdia s rozdelením populácie podľa T3A

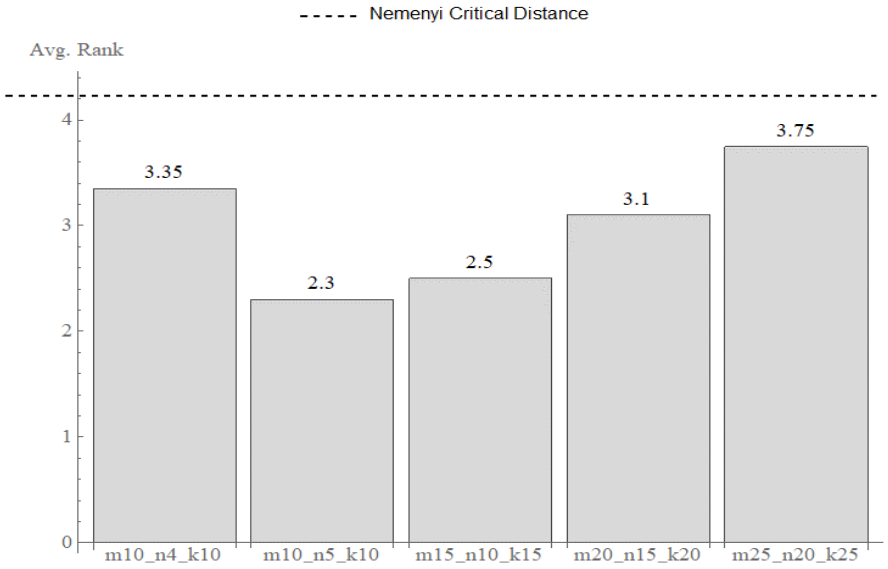
Pri tejto adaptívnej stratégii boli použité rôzne nastavenia vstupných parametrov *m*, *n* a *k*. Detailný popis techniky sa nachádza v kapitole 5.1. Odporúčaná hodnota autorom nastavenia parametrov pri veľkosti populácii 30 bola nasledovná [35]:

- $m = 10$, $n = 4$ a $k = 10$,

ďalšie nastavenia, ktoré boli testované sú nasledujúce:

- $m = 10$, $n = 5$ a $k = 10$,
- $m = 15$, $n = 10$ a $k = 15$,
- $m = 20$, $n = 15$ a $k = 20$,
- $m = 25$, $n = 20$ a $k = 25$.

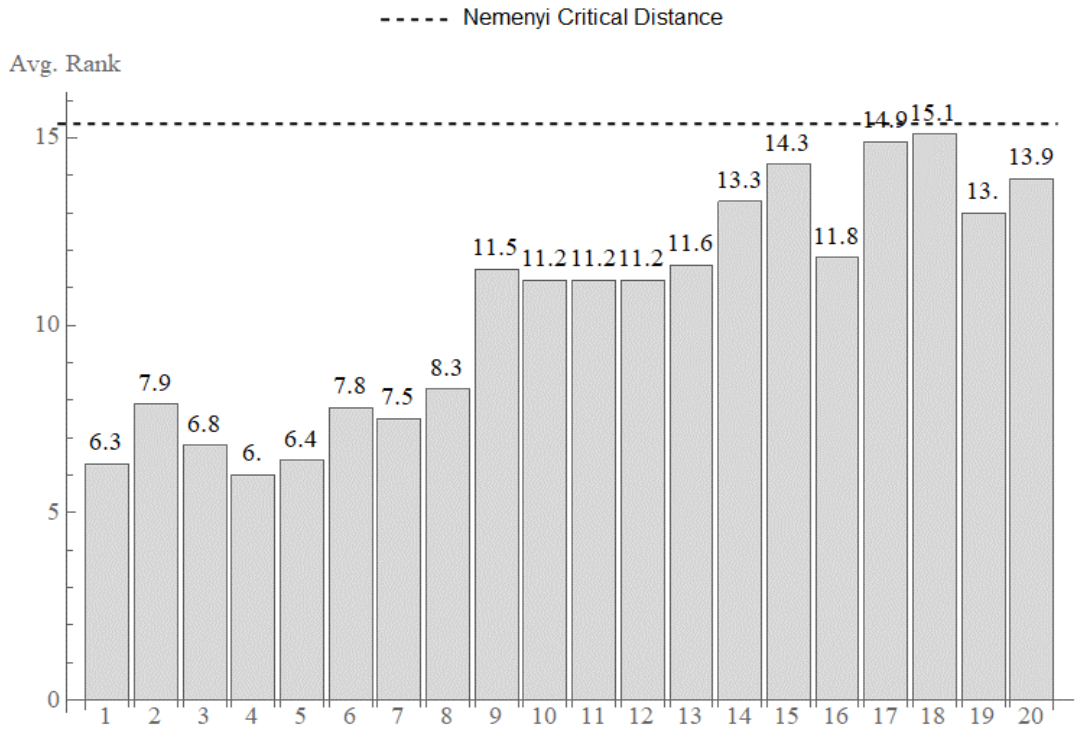
Podľa výsledkov Friedmanovho rankového testu na Obrázku 18, druhé nastavenie dosiahlo najlepšie hodnoty a z toho dôvodu toto nastavenie bude používane v ďalších testoch. Hodnota p-value bola 7,18E-14, nižšia ako hladina významnosti 5% a preto sú výsledky relevantné. Štatistického vyhodnotenia jednotlivých nastavení adaptívnej stratégie sa nachádzajú v prílohe PII.



Obrázok 18: Výsledky Friedmanovho rankového testu pre proces organizácie populácie podľa T3A

10.1.5 Experimentálna štúdia s clusteringom populácie

Táto adaptívna technika sa vyskytuje v iných moderných metaheuristikách, kde dosahovala prijateľné výsledky a z toho dôvodu bola aplikovaná na algoritmus SOMA. Keďže hodnota p-value bola 9,92E-06 a teda nižšia ako hladina významnosti 5%, tak výsledky sú považované za relevantné. Na nasledujúcom Obrázku 19 je znázornený výsledok Friedmanovho rankového testu. Najlepšie hodnoty dosahovalo štvrté nastavenie, a teda počet clusterov je nastavený na 6 až 7 percent veľkosti populácie (v našom prípade *počet clusterov* = 2) a *maximálny počet iterácii* pre algoritmus *k*-means, ktorý prerozdeľuje populáciu do clusterov je 1000. Takéto nastavenie bude používané aj v ďalších testoch. Štatistické vyhodnotenie jednotlivých nastavení adaptívnej stratégie sa nachádzajú v prílohe PII.



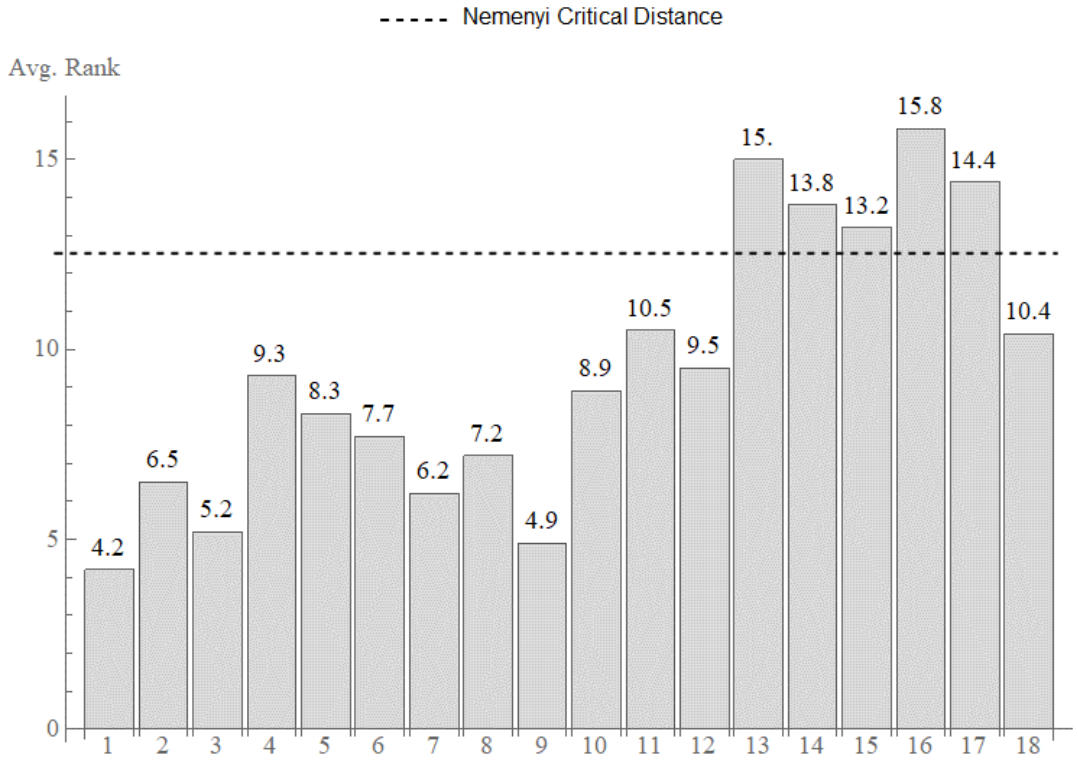
Obrázok 19: Výsledky Friedmanovho rankového testu pre proces organizácie populácie na clustery

Na výsledku z Friedmanovho rankového testu každý jeden blok predstavuje jedno konkrétne nastavenie vstupných parametrov pre túto stratégiu. Jednotlivé nastavenia sú *počet clusterov* zľava 2, 3, 4, 5 a 6, vždy pre *počet iterácii* 10, 100, 500 a 1000. Napríklad nastavenie č.1 je *počet clusterov* = 2 a *maximálny počet iterácii* = 10, nastavenie č. 4 je *počet clusterov* = 2 a *maximálny počet iterácii* = 1000 a nastavenie č. 5 je *počet clusterov* = 3 a *maximálny počet iterácii* = 10.

10.1.6 Experimentálna štúdia s reštartom populácie

Pre túto adaptívnu metódu bolo potrebné nastaviť niekoľko parametrov. Niektoré z nich sú NP_{add}, ktorý predstavuje veľkosť pridávanej populácie, *genToImprove*, ktorý predstavuje počet generácii, počas ktorých by malo dôjsť k zlepšeniu s určitou toleranciou a v neposlednom rade parameter *maxPopSize*, ktorý hovorí o tom aká môže byť maximálna veľkosť populácie. Na porovnanie rôznych kombinácii týchto parametrov boli znovu vyhotovené štatistické vyhodnotenia a vykoný Friedmanov rankový test. Keďže hodnota p-value bola 7,19E-14, výsledky tohto testu sú na hladine významnosti 5% relevantné.

Na nasledujúcom Obrázku 20 sa nachádza výsledok Friedmanovho rankového testu, kde je vidieť a s určitosťou sa dá povedať, že prvé nastavenie parametrov (prvý blok grafu) vedie výrazne lepšie ako nastavenia č. 13, 14, 15, 16 či 17 viď Obrázok 20, ktoré sa nachádzajú nad krivkou kritickej vzdialenosti. Do ďalších testov bolo vybrané práve nastavenie č. 1, ktoré dosahovalo najlepšie výsledkov.



Obrázok 20:Výsledky Friedmanovho rankového testu pre reštart populácie

Kombinácie jednotlivých nastavení parametrov boli zľava NP_{add} rovné 15, 30 a 45 vždy pre *genToImprove* 2, 5, 10 a *maxPopSize* 90 a 150. Prvé nastavenie je teda NP_{add} = 15, *genToImprove* = 2 a *maxPopSize* = 90. Druhé nastavenie je NP_{add} = 15,

genToImprove = 5, *maxPopSize* = 90 a nastavenie č. 4 je NP_{add} = 15, *genToImprove* = 2, *maxPopSize* = 150.

Štatistické vyhodnotenia jednotlivých nastavení adaptívnej stratégie sa nachádzajú v prílohe PII.

10.2 Experimenty s kombináciami adaptívnych techník

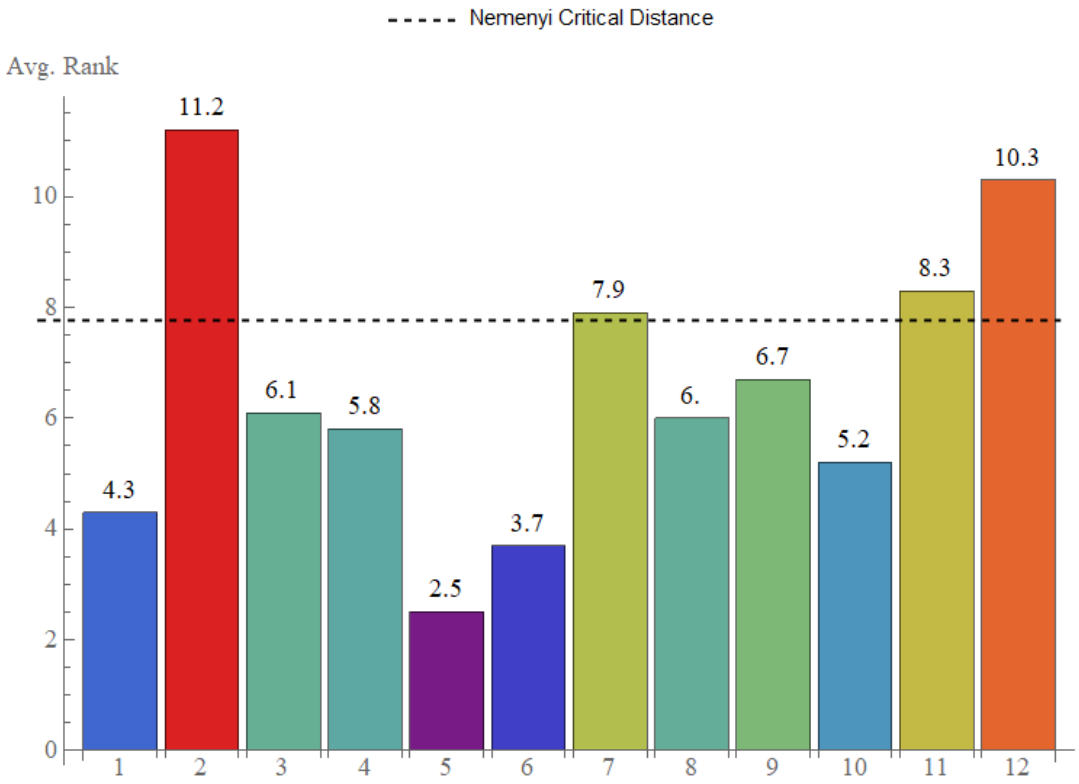
Ďalším krokom experimentov a testovaní bolo vytvoriť niekoľko kombinácii jednotlivých adaptívnych techník a porovnať ich. Jednotlivých adaptívnych mechanizmov bolo dvanásť, niektoré z nich už boli testované v predchádzajúcom kroku a to z dôvodu nájdenia najlepšieho nastavenia parametrov. V tomto porovnaní adaptívnych mechanizmov sa však objavia aj také, ktoré nemajú žiadne vstupné parametre iné ako tie, ktoré sú aj v základnej verzii algoritmu SOMA. Porovnávané boli nasledujúce techniky vyobrazené v Tabuľke 7, ktoré sú očíslované od 1 po 12 s tým, že toto číslo je následne aj zobrazené na výsledku z Friedmanovho rankového testu:

Tabuľka 7: Zoznam porovnávaných adaptívnych techník

	Názov	Popis
1.	Strategy	každý jedinec si volí svoju vlastnú stratégiu
2.	Choose PRT	každý jedinec si volí svoju vlastnú hodnotu PRT
3.	Cluster principe	aplikovanie clusteringu na počiatočnú populáciu
4.	Njumps	výmena parametra path_length za počet skokov
5.	PRT T3A	výpočet PRT parametra podľa T3A SOMA
6.	PRT Pareto	výpočet PRT parametra podľa Pareto SOMA
7.	PRTVector	výpočet parametra PRTVector na podľa Pareto SOMA
8.	Restart pop	reštart populácie o tzv. pridávanie jedincov do populácie
9.	Step Pareto	výpočet parametra step na základe implementácie Pareto SOMA
10.	Step T3A	výpočet parametra step na základe implementácie T3A SOMA
11.	T3A pop	proces organizácie populácie podľa T3A SOMA
12.	Pareto pop	proces organizácie populácie podľa Pareto SOMA

Síce sa na výsledku z Friedmanovho rankového testu objavujú adaptívne techniky pre výpočet parametru *step* a *PRT* podľa Pareto SOMA, už v kapitole 10.1.2 a 10.1.3 bolo štatisticky dokázané, že vzorce pre výpočet týchto parametrov podľa implementácie T3A boli účinnejšie a to potvrdzuje aj Friedmanov rankový test. Z nasledujúceho výsledku Friedmanovho rankového testu vyobrazeného na Obrázku 21 je vidieť, že adaptívna stratégia s č. 5,

a teda adaptívny parameter *PRT* podľa vzorca T3A rovnica č. 5, dosahuje s určitosťou lepšie výsledky ako adaptívne parametre č. 2, 7, 11 a 12. Výsledky z Friedmanovho rankového testu môžeme považovať za relevantné, keďže hodnota p-value bola 3,72E-26 a teda nižšia ako hodnota hladiny významnosti 5%.

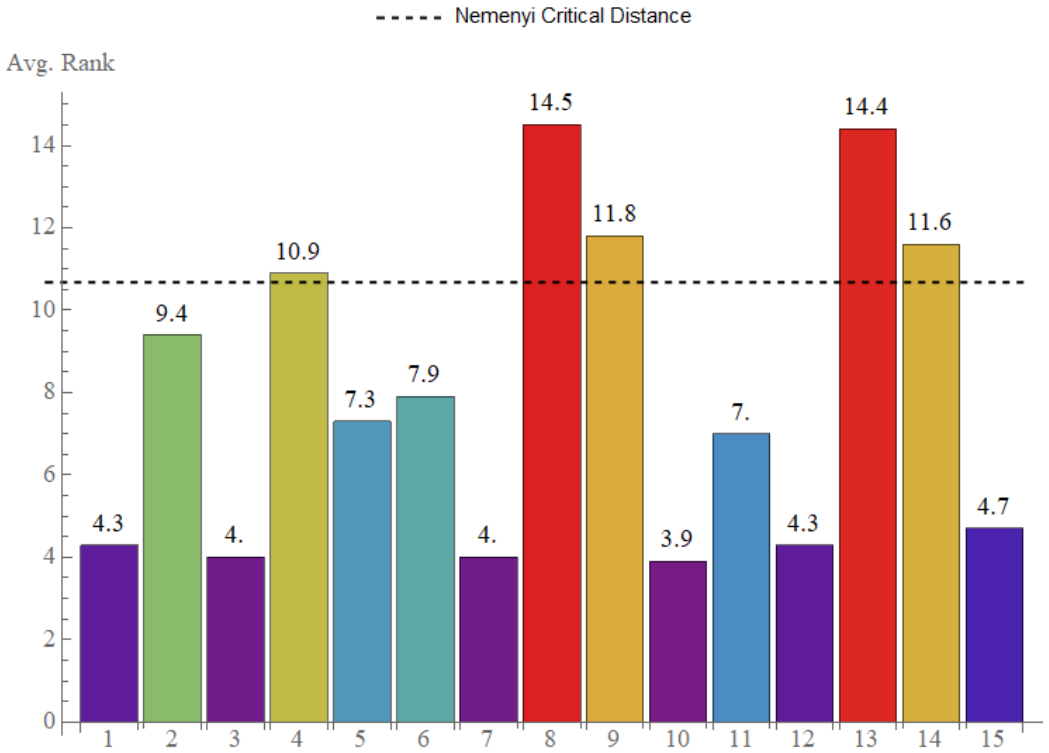


Obrázok 21: Výsledky Friedmanovho rankového testu pre jednotlivé adaptívne stratégie

Vzhľadom na výsledky Friedmanovho rankového testu bolo navrhnuté niekoľko kombinácií adaptívnych mechanizmov, ktoré by mohli dosahovať sľubné výsledky. Kombinácie týchto adaptívnych parametrov boli implementované rovnako na SOMA All To Random. Nastavenia vstupných parametrov základnej verzie SOMA a nastavenie dimenzie a *MaxFES* boli rovnaké ako pri testovaní jednotlivých stratégii. Navrhované, implementované a testované kombinácie sa nachádzajú v Tabuľke 8, kde krížik označuje využitie danej adaptívnej techniky.

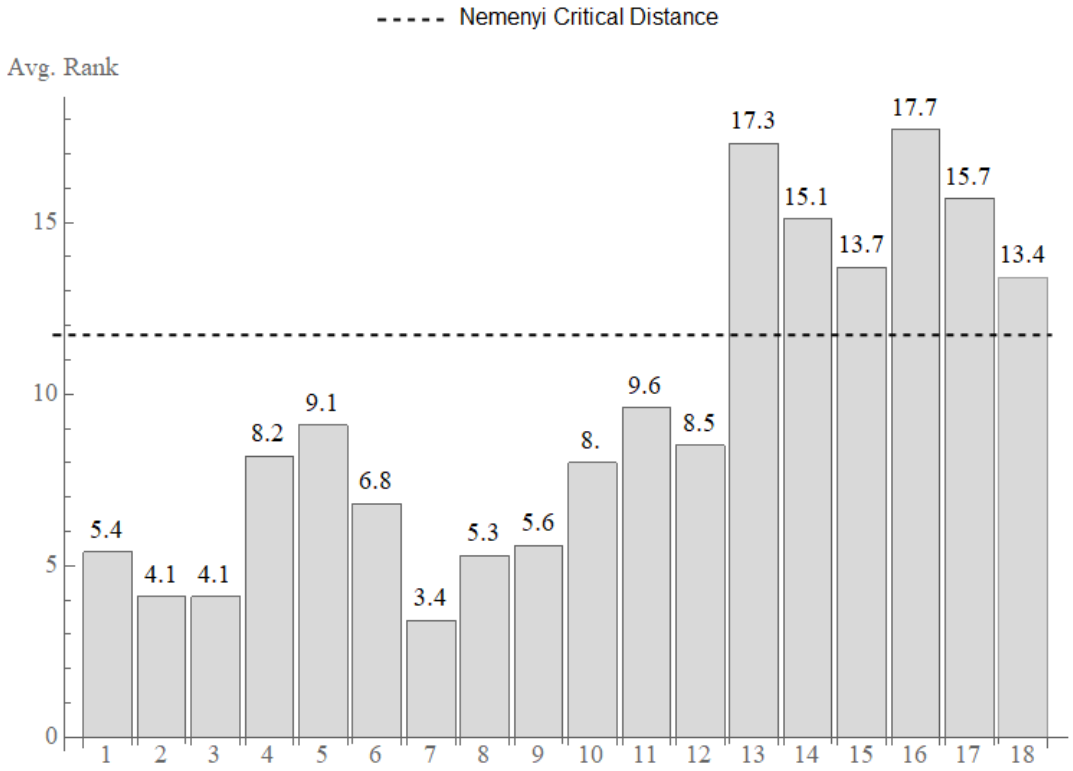
Tabuľka 8: Tabuľka testovaných kombinácii adaptívnych techník

Č. straté-gie	Strategy	Cluster principe	Restart pop	PRT T3A	step T3A	Njumps
1.	X			X		
2.		X		X		
3.			X	X		
4.	X	X				
5.	X	X		X		
6.	X	X		X	X	
7.	X			X	X	
8.		X			X	
9.			X		X	
10.			X	X	X	
11.	X	X		X		X
12.	X			X	X	X
13.		X			X	X
14.			X		X	X
15.			X	X	X	X



Obrázok 22: Výsledky Friedmanovho rankového testu pre jednotlivé kombinácie adaptívnych stratégií

Na základe výsledkov z Friedmanovho rankového testu vyobrazeného na Obrázku 22 bolo rozhodnuté venovať pozornosť kombináciám stratégií č. 1, 3, 7 a 10 z dôvodu dosiahnutia najnižších hodnôt, ktoré zároveň predstavujú vlastné modifikácie popisované v kapitole 7. Pre kombinácie stratégií č. 3 a 10 bolo potrebné ešte nájsť vhodné nastavenia vstupných parametrov a z toho dôvodu sa znova testovali tieto dva algoritmy s rôznymi nastaveniami. Kombinácie jednotlivých nastavení parametrov boli zľava NP_{add} rovné 15, 30 a 45 vždy pre *genToImprove* 2, 5, 10 a *maxPopSize* 90 a 150. Prvé nastavenie je teda $NP_{add} = 15$, *genToImprove* = 2 a *maxPopSize* = 90. Druhé nastavenie je $NP_{add} = 15$, *genToImprove* = 5, *maxPopSize* = 90 a nastavenie č. 4 je $NP_{add} = 15$, *genToImprove* = 2, *maxPopSize* = 150.

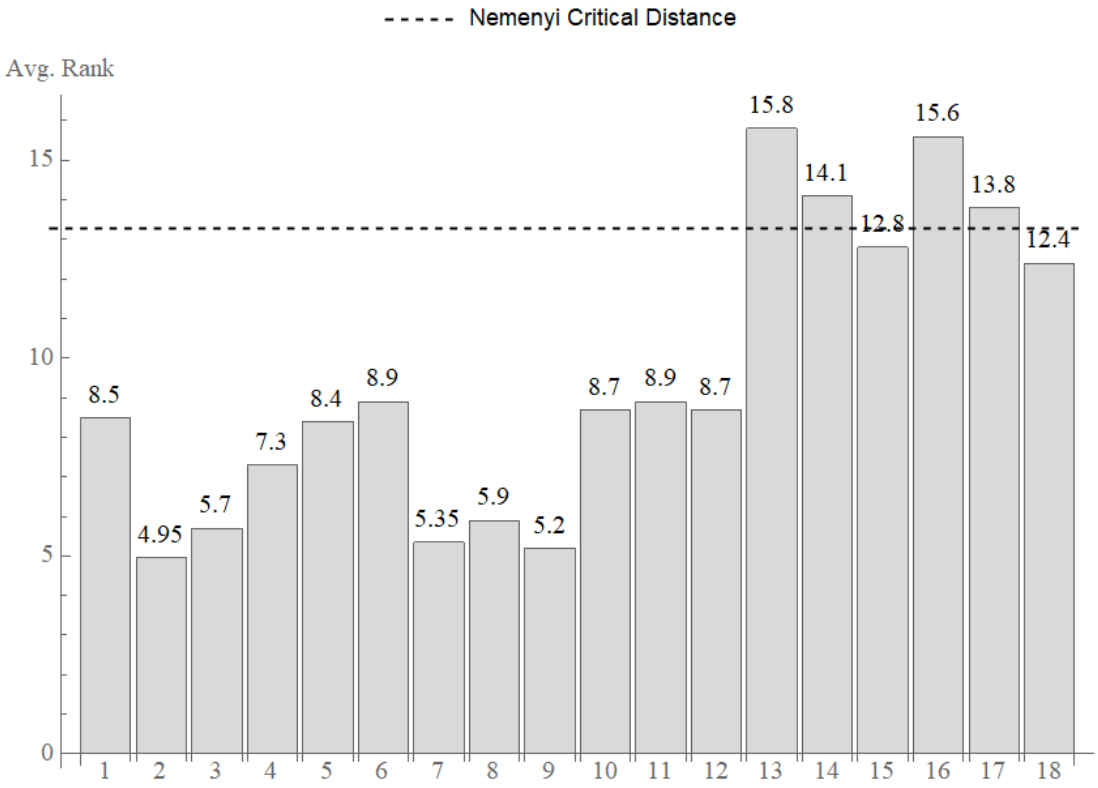


Obrázok 23: Výsledky Friedmanovho rankového testu pre reštart populácie u kombinácie stratégií č. 3

Na základe výsledku Friedmanovho rankového testu na Obrázku 23 a štatistického vyhodnotenia, ktoré sa nachádza v prílohe PII, bolo uznané vhodným nastavením pre kombináciu stratégií č. 3 nastavenie parametrov č.7 a teda $NP_{add} = 30$, *genToImprove* = 2 a *maxPopSize* = 90.

Z výsledku Friedmanovho rankového testu je možné tvrdiť, že nastavenie č. 7 si vedie výrazne lepšie ako nastavenia č. 13 až 18, ktoré sa nachádzajú nad krivkou kritickej vzdialenosti. Hodnota p-value bola 5,32E-45, výsledky tohto testu sú na hladine významnosti 5% relevantné.

Na nasledujúcom Obrázku 24 sa nachádzajú výsledky Friedmanovho rankového testu pre test nastavení parametrov kombinácie stratégií č. 10. Hodnota p-value bola 3,01E-20 a teda výsledky tohto testu sú relevantné na hladine významnosti 5%. S určitosťou je možné tvrdiť, že druhé nastavenie, kde NP_{add} = 15, *genToImprove* = 5 a *maxPopSize* = 90 si vedie výrazne lepšie ako nastavenia č. 13, 14, 16 a 17, ktoré sa nachádzajú nad krivkou kritickej vzdialenosti. Na základe Friedmanovho rankového testu a štatistického vyhodnotenia, ktoré sa nachádza v prílohe PII, nastavenie č. 2 bolo zvolené za vhodné nastavenie tejto modifikácie.



Obrázok 24: Výsledky Friedmanovho rankového testu pre reštart populácie u kombinácie stratégií č. 10

Vybrané 4 kombinácie adaptívnych stratégii a teda originálne navrhnuté modifikácie, ktoré sú popísané aj v kapitole 7, postúpili do posledného kola testovania a porovnávania už s existujúcimi verziami algoritmu SOMA. Grafické a štatistické vyhodnotenie finálnych testov sa nachádza v nasledujúcej kapitole.

11 VÝSLEDKY TESTOVANIA

Vzhľadom na odporúčania z dokumentácie pre benchmark CEC20 pre porovnávanie jednotlivých spomínaných algoritmov s novými modifikáciami boli zvolené nasledujúce nastavenia experimentov [40]:

1. **Dimenzie**:
    a. Pre funkcie 1 až 5 a 8 až 10 je dimenzia volená z množiny {5, 10, 15, 20}.
    b. Pre funkcie 6 a 7 je dimenzia volená z množiny {10, 15, 20}.
2. **Počet opakovaní jednotlivých experimentov**: 30.
3. **MaxFES**:

	MaxFES
***D* = 5**	5×10^4
***D* = 10**	1×10^6
***D* = 15**	3×10^6
***D* = 20**	1×10^7

4. **Hranice prehľadávaného priestoru**: $[-100, 100]^D$.
5. **Ukončovacia podmienka**: dosiahnutie *MaxFES* pre danú dimenziu.

Nastavenia jednotlivých porovnávaných algoritmov boli nastavené podľa odporúčania autorov a v prípade originálnych modifikácií nastavenia boli zvolené na základe množstva vykonaných predchádzajúcich testov. Nastavenia jednotlivých algoritmov sú prehľadne vyobrazené v nasledujúcej Tabuľke 9.

Tabuľka 9: Nastavenie vstupných parametrov pre jednotlivé algoritmy

	ATO	**ATR**	**ATA**	**T3A**	**Pareto**	**Mod. 1**	**Mod. 2**	**Mod. 3**	**Mod. 4**
popSize	30	30	30	30	30	30	30	30	30
pathLength	3,0	3,0	3,0	-	-	3,0	3,0	3,0	3,0
step	0,11	0,11	0,11	-	-	0,11	0,11	-	-
PRT	0,3	0,3	0,3	-	-	-	-	-	-
Njumps	-	-	-	45	45	-	-	-	-
m	-	-	-	10	-	-	-	-	-
n	-	-	-	4	-	-	-	-	-
k	-	-	-	10	-	-	-	-	-
T1	-	-	-	-	1	-	-	-	-
T2	-	-	-	-	1	-	-	-	-
NP_{add}	-	-	-	-	-	-	30	-	15
error_tolerance	-	-	-	-	-	-	0,001	-	0,001
genToImprove	-	-	-	-	-	-	2	-	5
maxPopSize	-	-	-	-	-	-	90	-	90
pop_add_generations	-	-	-	-	-	-	30	-	15

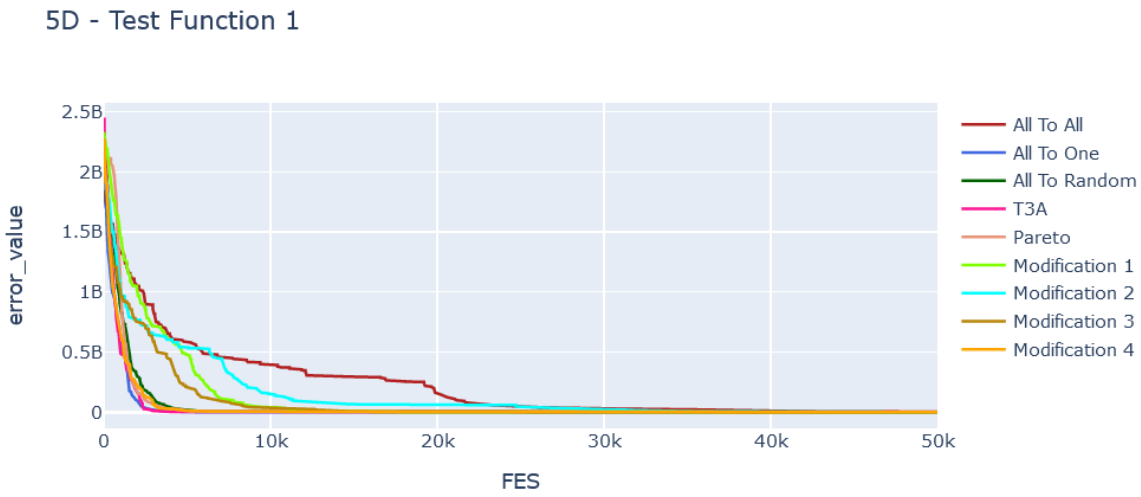
Testovanie prebiehalo na 10 funkciách a dimenziách 5*D*, 10*D*, 15*D* a 20*D*, zatiaľ čo každý test pre dané nastavenie sa opakoval 30 krát. Vzhľadom na početnosť vykonaných testov a početnosť výsledkov, je predstavené grafické vyhodnotenie a štatistické vyhodnotenie len na vybraných funkciách a dimenziách, pri čom ostatné štatistické vyhodnotenie sa nachádza v prílohe PII. Je ukázané vyhodnotenie na 5*D* a 10*D* z dôvodu prehľadnosti grafov a to na jednej funkcii z každého druhu (unimodálna, základná, hybridná a kompozitná).

Grafy zobrazujú spriemerované hodnoty 30 behov error-value, dosahované počas evolučného procesu každého jedného algoritmu. Dĺžka evolučného procesu je vyjadrená v jednotkách *FES* a teda počet ohodnotení účelových funkcií. Pod každým grafom sa nachádza príslušné štatistické vyhodnotenie ku funkcii a dimenzii, ktoré sú zobrazené na grafe. Štatistické vyhodnotenie sa vykonávalo z najlepších dosiahnutých hodnôt error-value počas každého opakovania behu pre jednotlivé algoritmy. Inak povedané, je to štatistika 30 najlepších hodnôt error-value, získaných počas opakovania behu každého jedného algoritmu. V každej tabuľke je zvýraznená najlepšia priemerná hodnota najlepších získaných hodnôt error-value počas všetkých behov jednotlivých verzií algoritmu a teda táto hodnota určuje, ktorý

algoritmus si viedol najlepšie na danej funkcii a v danom *D* – dimenzionálnom priestore.

11.1 Unimodálna funkcia: F1

Na základe nižšie vyobrazeného grafu na Obrázku 25 je možné vidieť správanie jednotlivých algoritmov v 5*D* priestore pre riešenie unimodálneho problému, kde účelová funkcia je reprezentovaná testovacou funkciou č. 1. Najrýchlejšiu konvergenciu dosahovali algoritmy SOMA All To One, T3A SOMA a Pareto SOMA. Na druhej strane najpomalšiu konvergenciu na tejto testovacej funkcii vykazovala SOMA All To All. Bližšie informácie o úspešnosti jednotlivých algoritmov na tejto testovacej funkcii v 5*D* priestore sú popísané pod obrázkom a dokázané na základe štatistického vyhodnotenia v Tabuľke 10.



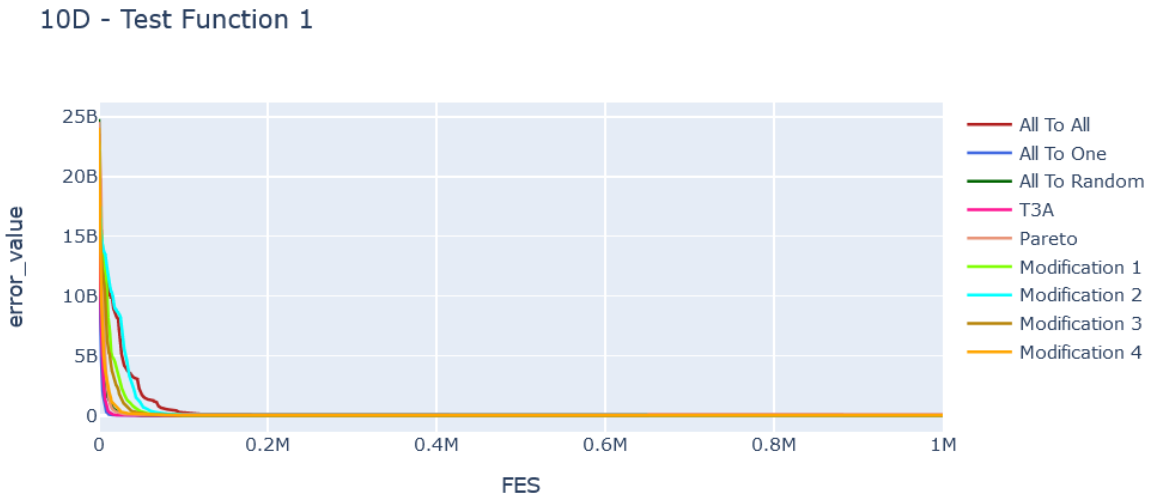
Obrázok 25: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 5*D* priestore pre testovaciu funkciu F1

Zo štatistického vyhodnotenia, ktoré sa nachádza nižšie v Tabuľke 10 vyplýva, že v 5*D* priestore niektoré verzie algoritmu dosiahli globálne minimum a teda hodnota error-value sa rovná 0.00E+00. Vzhľadom na priemernú hodnotu najlepších získaných hodnôt error-value počas všetkých behov si najlepšie viedla verzia All To One a najhoršie All To All.

Tabuľka 10: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 5*D* priestore pre funkciu F1

F1	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	0.00E+00	1.40E-07	**6.67E-09**	0.00E+00	2.59E-08
AllToRandom	2.59E-04	1.71E+01	9.39E-01	4.27E-02	3.25E+00
AllToAll	3.18E+02	5.55E+06	2.64E+06	2.29E+06	1.48E+06
T3A	0.00E+00	3.91E-07	3.31E-08	1.00E-08	7.53E-08
Pareto	2.00E-08	2.11E+01	1.29E+00	3.05E-04	4.55E+00
Modifikácia 1	1.00E-07	2.71E+03	5.90E+02	4.48E+02	7.02E+02
Modifikácia 2	1.31E+02	5.16E+05	1.12E+05	5.18E+04	1.34E+05
Modifikácia 3	0.00E+00	2.26E+05	2.76E+04	4.57E+03	5.64E+04
Modifikácia 4	1.12E-02	4.18E+03	4.09E+02	1.86E+01	9.12E+02

V 10*D* priestore na testovacej funkcii č. 1 malo taktiež najpomalšiu konvergenciu verzia All To All a modifikácia č. 2, ako je to možno vidieť na nasledujúcom Obrázku 26. Najrýchlejšiu konvergenciu dosahovali verzie All To One a T3A, podobne ako tomu bolo aj v 5*D* priestore. Bližšie informácie o úspešnosti jednotlivých algoritmov na tejto testovacej funkcii v 10*D* priestore sú zhrnuté pod obrázkom a ukázané na základe štatistického vyhodnotenia v Tabuľke 11.



Obrázok 26: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 10*D* priestore pre testovaciu funkciu F1

Zo štatistického vyhodnotenia v Tabuľke 11 je zrejmé, že v 10*D* priestore niektoré verzie algoritmu dokázali dosiahnuť globálneho minima ako tomu bolo aj v 5*D* priestore.

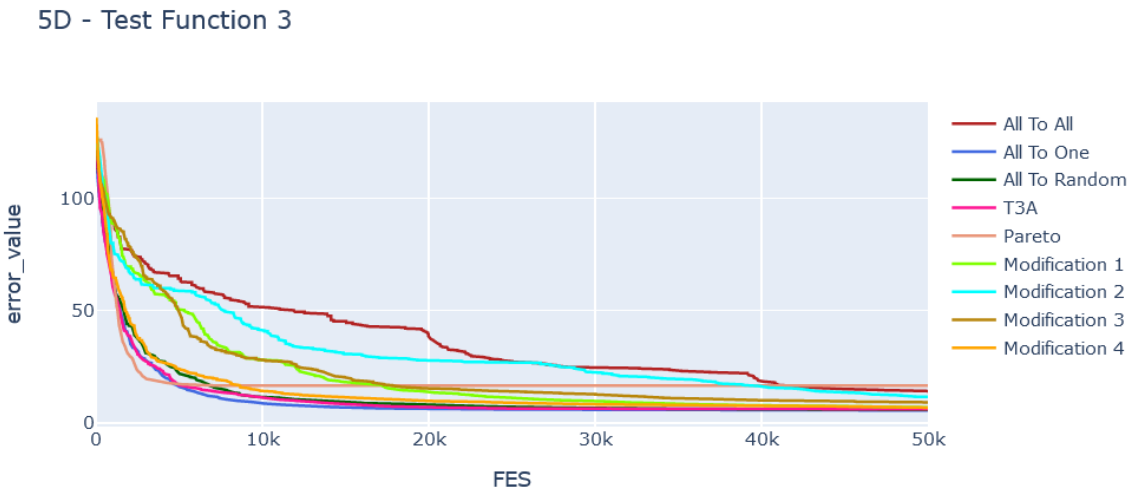
Vzhľadom na priemernú hodnotu najlepších získaných hodnôt error-value počas všetkých behov si najlepšie viedla opäť verzia All To One a najhoršie Pareto.

Tabuľka 11: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 10*D* priestore pre funkciu F1

F1	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	0.00E+00	1.00E-08	**2.00E-09**	0.00E+00	4.07E-09
AllToRandom	0.00E+00	2.54E-01	9.29E-03	1.00E-08	4.58E-02
AllToAll	8.26E-06	1.81E-01	1.81E-02	1.31E-03	4.34E-02
T3A	0.00E+00	2.00E-08	6.33E-09	1.00E-08	6.15E-09
Pareto	6.23E+04	7.47E+08	7.61E+07	1.72E+07	1.51E+08
Modifikácia 1	0.00E+00	1.48E-02	5.58E-04	3.33E-06	2.70E-03
Modifikácia 2	1.23E-05	7.47E-03	6.39E-04	2.27E-04	1.41E-03
Modifikácia 3	0.00E+00	1.16E-01	6.79E-03	2.53E-05	2.44E-02
Modifikácia 4	3.14E-06	5.13E-03	3.85E-04	7.82E-05	9.65E-04

11.2 Základná funkcia: F3

Zástupcom základných funkcií bola vybraná funkciu č. 3, na ktorej sa demonštrujú výsledky v 5 a 10*D* priestore. Ako z nasledujúceho Obrázka 27 vyplýva v 5*D* priestore, najrýchlejšie konvergovala verzia Pareto avšak je vidieť, že následne uviazla pravdepodobne v nejakom lokálnom minime a v konečnom dôsledku dosahovala najhoršie výsledky. Najpomalšie znova konvergovala verzia All To All.



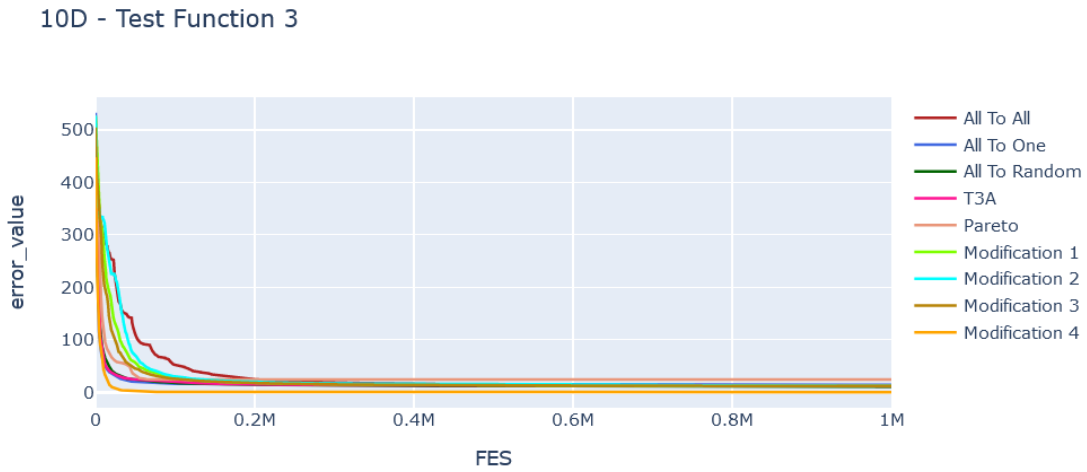
Obrázok 27: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 5*D* priestore pre testovaciu funkciu F3

Aj z grafu a aj z nasledujúcich štatistických výsledkov zhrnutých v Tabuľke 12 je zrejmé, že najlepšie výsledky dosahovala verzia All To One, ktorá tiež ako jediná dokázala dosiahnuť globálneho minima. Naopak najhoršie výsledky dosiahla verzia Pareto.

Tabuľka 12: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 5*D* priestore pre funkciu F3

F3	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	0.00E+00	7.52E+00	**5.59E+00**	5.82E+00	1.29E+00
AllToRandom	5.16E+00	7.93E+00	5.81E+00	5.69E+00	5.55E-01
AllToAll	5.92E+00	2.15E+01	1.52E+01	1.39E+01	2.95E+00
T3A	5.14E+00	8.19E+00	6.12E+00	5.96E+00	8.44E-01
Pareto	6.49E+00	3.06E+01	1.66E+01	1.53E+01	6.96E+00
Modifikácia 1	4.81E+00	9.05E+00	6.90E+00	6.72E+00	1.02E+00
Modifikácia 2	8.23E+00	1.82E+01	1.17E+01	1.09E+01	2.59E+00
Modifikácia 3	3.82E+00	1.46E+01	9.19E+00	9.03E+00	2.22E+00
Modifikácia 4	5.27E+00	9.32E+00	6.80E+00	6.42E+00	9.95E-01

V 10*D* priestore najrýchlejšiu konvergenciu preukazovala modifikácia č. 4, ktorá aj ako jediná dosiahla globálne minimum ako je zrejmé z nasledujúceho grafu vyobrazeného na Obrázku 28 a aj zo štatistických údajov uvedených v Tabuľke 13. Rovnako rýchlo konvergovali aj verzie All To One, All To Random či T3A. Najpomalšiu konvergenciu opäť vykazovala verzia All To All.



Obrázok 28: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 10*D* priestore pre testovaciu funkciu F3

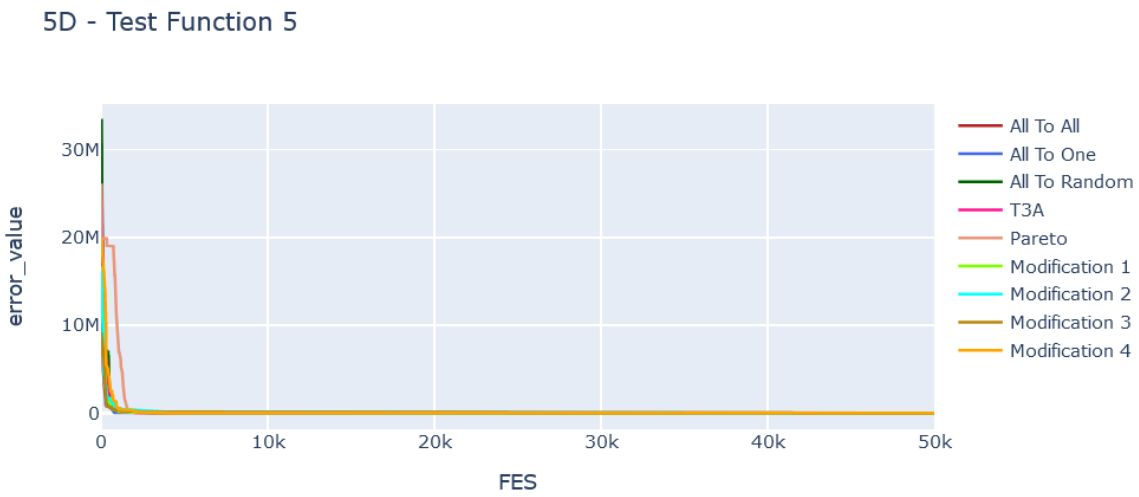
Z nasledujúcich štatistických výsledkov zobrazených v Tabuľke 13 vyplýva, že najlepšie výsledky v 10*D* priestore pre testovaciu funkciu č. 3 dosahovala verzia modifikácia 4 a najhoršie opäť verzia Pareto, ako bolo ukázané už aj na predchádzajúcom grafe.

Tabuľka 13: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 10*D* priestore pre funkciu F3

F3	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	1.09E+01	2.07E+01	1.50E+01	1.40E+01	2.02E+00
AllToRandom	1.04E+01	1.21E+01	1.10E+01	1.10E+01	3.90E-01
AllToAll	1.05E+01	1.26E+01	1.15E+01	1.15E+01	6.12E-01
T3A	1.04E+01	1.47E+01	1.25E+01	1.24E+01	1.04E+00
Pareto	1.46E+01	3.65E+01	2.43E+01	2.28E+01	5.81E+00
Modifikácia 1	1.09E+01	1.31E+01	1.17E+01	1.16E+01	5.48E-01
Modifikácia 2	1.07E+01	1.36E+01	1.19E+01	1.19E+01	7.18E-01
Modifikácia 3	1,05E+03	1.32E+01	1.16E+01	1.14E+01	6.90E-01
Modifikácia 4	0.00E+00	2.00E-08	**4.00E-09**	0.00E+00	6.21E-09

11.3 Hybridná funkcia: F5

Zástupcom hybridných funkcií je funkcia č. 5 a to z toho dôvodu, že pre 5*D* priestor funkcie 6 a 7, ktoré patria tiež medzi hybridné neboli vykonávané kvôli odporúčaným nastaveniam benchmarku. Hoci nasledujúci graf zobrazený na Obrázku 29 nie je moc prehľadný, no je možné spozorovať, že najpomalšiu konvergenciu vykazovalo Pareto a zvyšné verzie konvergovali približne rovnakým tempom.



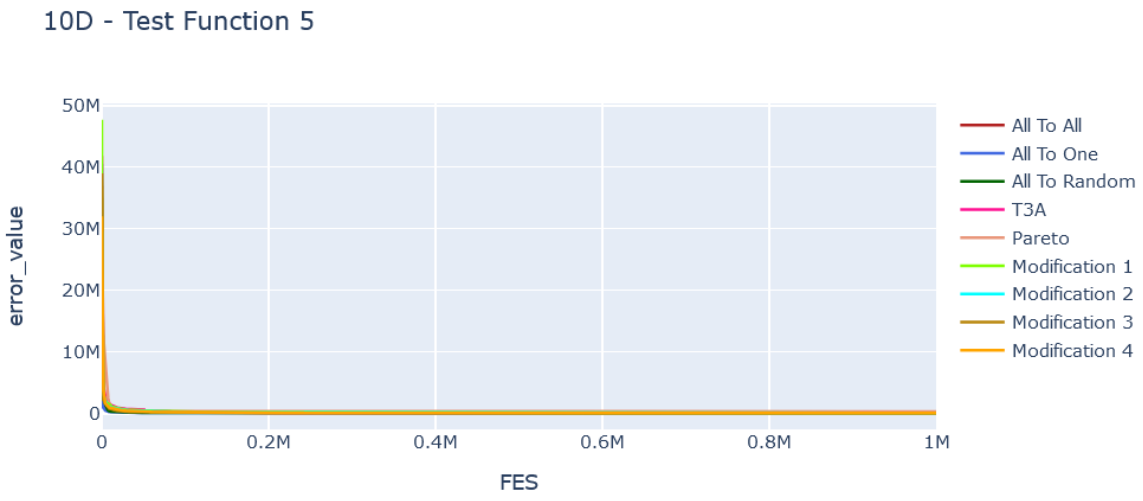
Obrázok 29: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 5*D* priestore pre testovaciu funkciu F5

V tomto prípade, kedy graf nie je moc prehľadný, viac o správaní jednotlivých algoritmov napovie nasledujúce štatistické vyhodnotenie zhrnuté v Tabuľke 14, kde je možné spozorovať, že na základe priemeru najlepšie si viedla verzia All To Random hoci nedosiahla globálne minimum ako tomu je u All To One či T3A. Taktiež je zrejmé, že opäť najhoršie si viedlo Pareto.

Tabuľka 14: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 5*D* priestore pre funkciu F5

F5	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	0.00E+00	3.98E+00	4.71E-01	3.12E-01	7.86E-01
AllToRandom	5.42E-06	6.39E-01	**9.00E-02**	1.39E-03	2.01E-01
AllToAll	1.70E+01	8.39E+01	4.83E+01	4.67E+01	1.98E+01
T3A	0.00E+00	3.97E+00	3.78E-01	1.00E-08	7.99E-01
Pareto	1.98E+00	2.75E+02	6.56E+01	4.71E+01	6.25E+01
Modifikácia 1	1.70E-04	5.68E+00	2.25E+00	1.90E+00	1.34E+00
Modifikácia 2	5.52E-01	2.57E+01	1.01E+01	9.13E+00	6.75E+00
Modifikácia 3	2.36E-01	2.54E+01	7.97E+00	5.41E+00	7.24E+00
Modifikácia 4	7.56E-04	6.13E+00	9.86E-01	6.28E-01	1.44E+00

V prípade 10*D* priestore, graf zobrazený na Obrázku 30 moc neprezradí z dôvodu neprehľadnosti, ale je možné konštatovať, že všetky verzie algoritmu SOMA konvergovali podobným tempom. Viac informácii preukáže štatistické vyhodnotenie, ktoré sa nachádza v Tabuľke 15.



Obrázok 30: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 10*D* priestore pre testovaciu funkciu F5

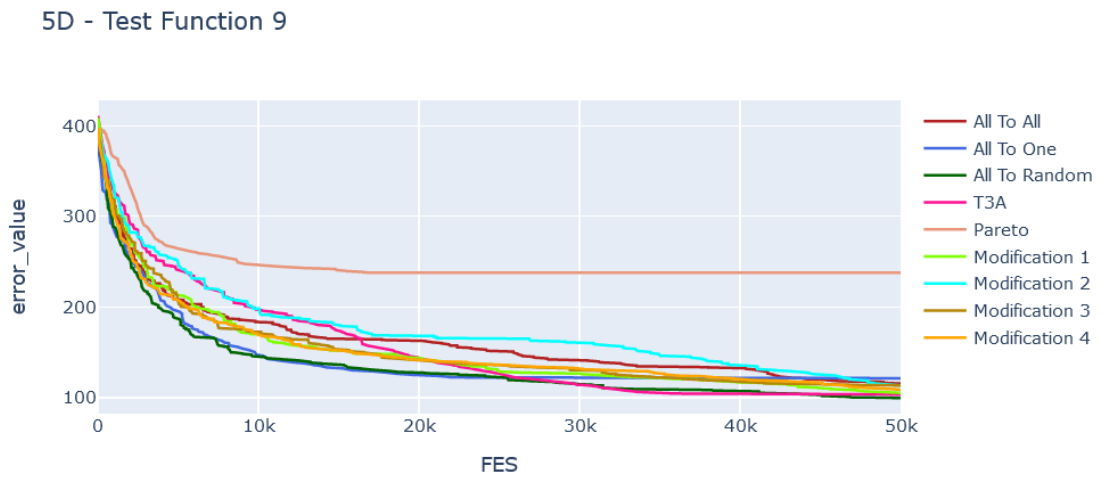
Na základe štatistického vyhodnotenia nižšie v Tabuľke 15 vyplýva, že najlepšie si viedla verzia All To Random, ktorá sa aj najbližšie priblížila ku globálnemu minimu. Opäť najhoršie výsledky na základe priemeru dosahovalo Pareto.

Tabuľka 15: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 10*D* priestore pre funkciu F5

F5	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	2.19E+00	2.68E+02	7.10E+01	4.10E+01	7.80E+01
AllToRandom	4.69E-03	1.86E+01	**4.46E+00**	1.57E+00	5.36E+00
AllToAll	4.69E-01	2.65E+02	2.38E+01	8.03E+00	5.61E+01
T3A	2.08E-01	1.33E+02	3.10E+01	1.31E+01	4.02E+01
Pareto	2.63E+01	1.61E+06	2.62E+05	1.59E+04	4.63E+05
Modifikácia 1	3.87E-01	1.27E+02	1.88E+01	1.16E+01	3.05E+01
Modifikácia 2	2.32E-01	3.65E+01	7.39E+00	4.85E+00	7.82E+00
Modifikácia 3	2.08E-01	4.43E+01	1.25E+01	1.20E+01	1.24E+01
Modifikácia 4	2.10E-01	6.49E-01	9.83E+00	5.12E+00	2.10E+01

11.4 Kompozitná funkcia: F9

Ako v poslednom zastúpení kompozitných funkcií je zvolená funkciu F9. Z nasledujúceho grafu na Obrázku 31 vyplýva, že v 5*D* priestore najrýchlejšie konvergovali verzie All To One a All To Random. Práve naopak najpomalšie konvergovalo Pareto, ktoré ako je vidieť, malo na tejto funkcií tendenciu uviaznuť v lokálnom minime. Tiež je zrejmé, že najlepšie výsledky dosiahla verzia All To Random, pričom v tesnej blízkosti za ním nasledovala T3A, modifikácia č.1 a 4.



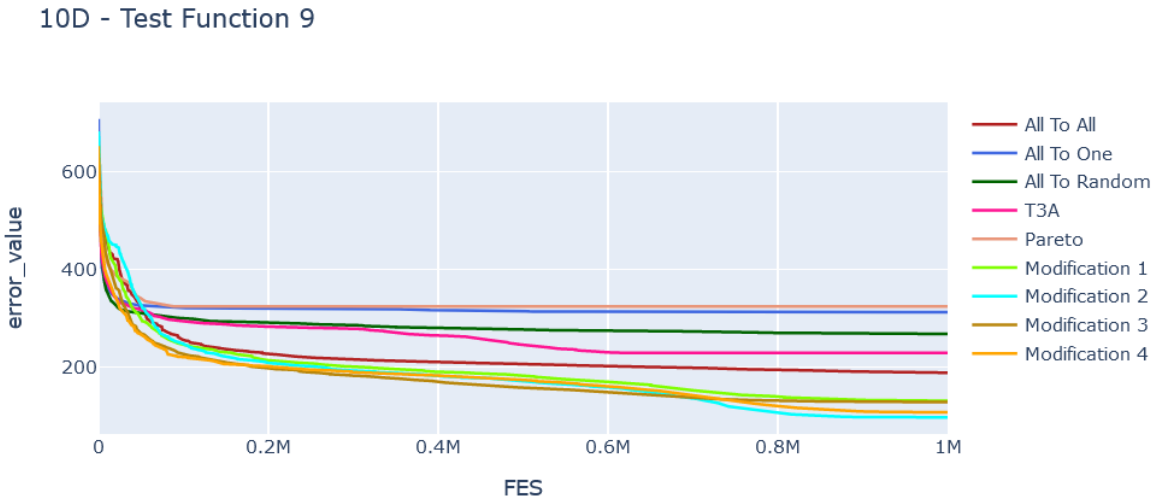
Obrázok 31: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 5*D* priestore pre testovaciu funkciu F9

Nižšie uvedené štatistické vyhodnotenie v Tabuľke 16 potvrdzuje to, čo bolo možné vidieť na predchádzajúcom grafe. Najlepšie výsledky dosiahla verzia All To Random. V tesnej blízkosti za ním nasledovali T3A, modifikácia č. 1 a 4. Taktiež zo štatistík vyplýva, že žiadna verzia nebola schopná na tejto testovacej funkcii v 5*D* priestore dosiahnuť globálne minimum.

Tabuľka 16: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 5*D* priestore pre funkciu F9

F9	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	7.40E-07	3.21E+02	1.21E+02	1.00E+02	7.95E+01
AllToRandom	1.74E+01	1.94E+02	**9.95E+01**	1.04E+02	4.00E+01
AllToAll	6.28E+01	1.48E+02	1.15E+02	1.17E+02	1.97E+01
T3A	2.00E-08	3.00E+02	1.03E+02	1.00E+02	7.20E+01
Pareto	1.98E+01	3.51E+02	2.38E+02	3.11E+02	1.13E+02
Modifikácia 1	1.91E+01	1.59E+02	1.05E+02	1.09E+02	2.78E+01
Modifikácia 2	5.89E+01	1.83E+02	1.13E+02	1.14E+02	2.10E+01
Modifikácia 3	3.45E+01	2.05E+02	1.13E+02	1.12E+02	2.99E+01
Modifikácia 4	7.92E+01	1.36E+02	1.09E+02	1.08E+02	1.27E+01

V 10*D* priestore na tejto testovacej funkcii dosiahla najlepších výsledkov modifikácia č. 2. Taktiež z nasledujúceho Obrázka 32 je zrejmé, že najhorších výsledkov dosiahlo Pareto a v tesnej blízkosti verzia All To One, hoci táto verzia pomerne rýchlo konvergovala no vyzerá, že mala tendenciu uviaznuť v nejakom lokálnom minime.



Obrázok 32: Priemerné hodnoty error-value získané počas 30 behov algoritmov v 10*D* priestore pre testovaciu funkciu F9

Z grafu a na základe priemeru vyplýva, že modifikácia č. 2 dosiahla najlepších výsledkov, hoci najbližšie ku globálnemu minimu sa priblížila verzia T3A, ako je možné vidieť v nasledujúcom štatistickom vyhodnotení v Tabuľke 17. Taktiež z grafu je zrejmé, že najhoršie výsledky dosahovala verzia Pareto, a to potvrdzuje aj štatistické vyhodnotenie.

Tabuľka 17: Štatistické vyhodnotenie najlepších dosiahnutých hodnôt error-value pre jednotlivé algoritmy v 10*D* priestore pre funkciu F9

F9	Min	Max	Priemer	Medián	Štandardná odchýlka
AllToOne	1.00E+02	3.59E+02	3.12E+02	3.45E+02	8.50E+01
AllToRandom	1.00E+02	3.42E+02	2.68E+02	2.98E+02	8.50E+01
AllToAll	1.00E+02	3.45E+02	1.88E+02	1.42E+02	9.43E+01
T3A	2.00E-08	3.50E+02	2.29E+02	3.35E+02	1.25E+02
Pareto	1.00E+02	3.66E+02	3.24E+02	3.48E+02	7.20E+01
Modifikácia 1	4.05E-01	3.43E+02	1.31E+02	1.00E+02	7.95E+01
Modifikácia 2	3.62E-02	2.01E+02	**9.67E+01**	1.00E+02	3.20E+01
Modifikácia 3	1.00E+02	3.43E+02	1.28E+02	1.00E+02	7.40E-01
Modifikácia 4	1.00E+02	2.02E+02	1.07E+02	1.00E+02	2.57E+01

11.5 Celková úspešnosť algoritmov naprieč funkciami a dimenziami

V predchádzajúcich kapitolách boli predstavené výsledky aké dosahovali algoritmy na vybraných testovacích funkciách v 5*D* a 10*D* priestore. Zvyšné štatistické vyhodnotenia pre všetky testovacie funkcie a všetky testované dimenzie sa nachádzajú v prílohe PII. Keďže spomínané štatistické údaje sa nachádzajú v 38 tabuľkách, nižšie v Tabuľke 18 je vyhodnotená a prehľadne zobrazená celková úspešnosť jednotlivých algoritmov pre testované dimenzie.

Tabuľka 18: Celková úspešnosť algoritmov

	5*D*	**10*D***	**15*D***	**20*D***
All To One	**5**	1	-	-
All To Random	2	2	-	-
AllToAll	-	-	-	-
T3A	-	-	-	-
Pareto	-	-	-	-
Modifikácia 1	1	1	-	1
Modifikácia 2	-	2	**4**	-
Modifikácia 3	-	1	1	1
Modifikácia 4	-	**3**	**4**	**7**

Ako je zrejmé z Tabuľky 18, v nižšej dimenzii *5D* dosahovali najlepšie výsledky prevažne základné verzie algoritmu SOMA. V 5*D* priestore si na všetkých testovaných funkciách najlepšie viedla verzia All To One s úspešnosťou 5/8. V 10*D* priestore už je možné vidieť, že sa prejavuje úspešnosť aj u navrhnutých modifikáciách. Najväčšiu úspešnosť mala modifikácia č. 3 a to 3/10. V 15*D* priestore sa o prvenstvo najvyššej úspešnosti bijú modifikácia č. 2 a 4, ktoré mali obe úspešnosť 4/9. Úspešnosť je rátaná z 9 prípadov, keďže v jednom prípade rozdiel medzi testovanými algoritmami nebol signifikantný. V 20*D* priestore so silným náskokom viedla modifikácia č. 4 a to s úspešnosťou 7/9. Opäť bola úspešnosť rátaná z 9 prípadov, keďže v jednom prípade rozdiel medzi testovanými algoritmami nebol signifikantný.

12 ZÁVER

Hlavným cieľom diplomovej práce bolo preskúmať existujúce moderné adaptívne stratégie algoritmu SOMA a navrhnúť vlastné modifikácie, či adaptívne stratégie pre riadenie populačnej dynamiky a následne porovnať ich s existujúcimi verziami algoritmu SOMA.

V teoretickej časti boli popísane EVT a základné princípy algoritmu SOMA, či iných verzií tohto algoritmu. Praktická časť bola zameraná na moderné adaptívne mechanizmy v algoritme SOMA alebo v iných metaheuristikách. Zisťoval sa ich vplyv na daný algoritmus, boli navrhnuté originálne adaptívne techniky a pomocou kombinácií existujúcich a navrhnutých variant sa vytvorilo niekoľko nových modifikácií algoritmu SOMA. Navrhovanými novými adaptívnymi mechanizmami bol clustering, ktorý sa používal na počiatočnú populáciu a druhým bol reštart populácie. Na základe množstva testov sa preukázalo, že clustering v takomto použití pre rozdelenie počiatočnej populácie nie je moc vhodný, i keď v iných metaheuristikách dosahoval vcelku priaznivé výsledky. Práve naopak testy preukázali, že adaptívna technika reštart populácie môže mať potenciál pri algoritme SOMA. Do konečného testovania sa dostali 4 najlepšie modifikácie.

Vzhľadom na spomínané fakty v praktickej časti diplomovej práce, ktoré sú podložené stovkami testov a štatistických vyhodnotení bolo zistené, že hoci vlastné navrhnuté modifikácie nedosiahli lepších výsledkov v nižších dimenziách oproti existujúcim, či ba základným verziám algoritmu SOMA, no práve naopak vo vyšších dimenziách dosahovali lepšie výsledky, a to hlavne modifikácie, ktoré obsahovali adaptívnu stratégiu reštart populácie. Je to z dôvodu toho, že ostatné verzie algoritmu SOMA vo vysokých dimenziách mali tendenciu uviaznuť v lokálnom minime, pri čom práve vlastná navrhnutá adaptívna stratégia má za úlohu takto uviaznutému algoritmu napomôcť uniknúť odtiaľ a to práve pridaním nových jedincov k pôvodnej populácii.

Hoci pri čítaní tejto diplomovej práce čitateľ môže nadobudnúť dojem, že porovnávame výkonnosť algoritmov a úspešnosť, kedy tvrdíme, že jeden algoritmus je lepší ako druhý, no stále treba mať na pamäti tzv. „No Free Lunch“ teorém, ktorý hovorí o tom, že neexistuje univerzálny algoritmus, ktorý dokáže riešiť všetky problémy lepšie ako iné algoritmy a teda každý algoritmus je vhodný pre riešenie rôznych typov problémov.

Je však možné si položiť otázku: „Čo sa teda dosiahlo touto prácou?“ Odpoveď je pomerne jednoduchá a zhrnieme si ju v niekoľkých nasledujúcich bodoch:

- Vzhľadom na veľké množstvo vykonaných testov, pri niektorých moderných adaptívnych technikách sme získali podstatné informácie ohľadne závislosti výkonu algoritmu SOMA na nastavení vstupných parametroch daných adaptívnych mechanizmov.
- Ďalej vzhľadom na množstvo vykonaných testov sme získali cenné informácie ohľadne vplyvu jednotlivých moderných adaptívnych techník, či už z existujúcich verzií SOMA alebo z iných metaheuristík.
- Ďalej vzhľadom na množstvo vykonaných testov sme získali cenné informácie ohľadne vhodnosti kombinácii jednotlivých moderných adaptívnych mechanizmov.
- Na základe všetkých simulačných pokusoch, ktoré boli vykonané sme získali nové poznatky, ktoré môžu byť nápomocné pre ďalších ľudí, ktorí sa o túto oblasť zaujímajú, či pre nás ak by sme chceli ďalej v tomto výskume pokračovať.

ZOZNAM POUŽITEJ LITERATÚRY

[1] ŠENKEŘÍK, Roman. *Evoluční výpočetní techniky: Úvod.* 1. prednáška z predmetu Evoluční výpočetní techniky. Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, 2020. [cit. 2020-12-25].

[2] THAN, Ker. *What is Darwin's Theory of Evolution?* [online]. 2018-02-27 [cit. 2020-12-25]. Dostupné z: https://www.livescience.com/474-controversy-evolution-works.html.

[3] HO, Y.C and D.L.Pepyne. *Simple Explanation of the No-Free-Lunch Theorem and Its Implications.* 2002. Journal of Optimization Theory and Applications 115, 549-570 [cit. 2020-12-31].

[4] ZELINKA Ivan Zuzana OPLATKOVÁ Miloš ŠEDA Pavel OŠMERA a Františe ČELAŘ. Evoluční výpočetní techniky: principy a aplikace. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3. [cit. 2020-12-31].

[5] WOLPERT, David and W.G. Macready. 2005. *Coevolutionary free lunches.* 2005 IEEE Transactions on Evolutionary Computation, IEEE, 2005, 721-735. [cit. 2020-12-31].

[6] ŠENKEŘÍK, Roman. *Evoluční výpočetní techniky: Vybrané algoritmy PSO, SOMA, ACO, SS.* 5.2. prednáška z predmetu Evoluční výpočetní techniky. Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, 2020. [cit. 2020-12-31].

[7] BORTEL, Martin. Evoluční algoritmy. 2011. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedouci práce Ing. Petra Lambertová. [cit. 2021-01-02].

[8] PRICE, Kenneth V, Rainer M STORN and Jouni A LAMPINEN. *Differentional evolution: a practical approach to global optimization.* 1. edition Berlin: Springer, 2005, 538s. ISBN 35-402-0950-6. [cit. 2021-01-02].

[9] ŠENKEŘÍK, Roman. *Evoluční výpočetní techniky: Základní pojmy, krízové stavy, benchmarking.* 2. prednáška z predmetu Evoluční výpočetní techniky. Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, 2020. [cit. 2021-01-02].

[10] HAUPT, Daniel. *Evoluční algoritmy.* 2009. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Vedouci práce Ing. Petr Honzík, PhD. [cit. 2021-01-02].

[11] SURJANOVIC Sonja and Darek BINGHAM. *Test functions and Datasets.* [online]. 2013. Simon Frases University. [cit. 2021-01-12]. Dostupné z: https://www.sfu.ca/~ssurjano/optimization.html.

[12] ARORA, Jasbir Singh. *Introduction to Optimum Design.* Fourth Edition 2017, 945 pages. ISBN 9780128008065. . [cit. 2021-01-12].

[13] CHALUPNÍK italij. Biologické algoritmy (1) - Evoluční algoritmy [online]. 4. 4. 2012 [cit. 2021-01-12]. Dostupné z: http://www.root.cz/clanky/biologicke- algoritmy-1-evolucni-algoritmy.

[14] MALLAWAARACHCHI, Vijini. *Introduction to Genetic Algorithms – Includin Example Code.* 2017-07-08 [cit. 2021-01-21]. Dostupné z: https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[15] GAD, Ahmed. *Introduction to Optimization with Genetic Algorithm*. 2018-07-03 [cit. 2021-01-21]. Dostupné z: https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b

[16] COHOON, James P. *Punctuated equilibria: a parallel genetic algorithm.* Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms. July 1987 at Massachusetts Institute of Technology, Cambridge, MA.

[17] GOLDBERG, David E., Bradley KORB and Kalyanmoy DEB. *Messy genetic algorithms: Motivation, analisys and frist results.* 1989 Complex systems 3.5: 493 – 530, 1989.

[18] KELLY, Jr., James D. And Lawrence DAVIS. *A Hybrid Genetic Algorithm for Classification.* 1991 IJCAI. Vol. 91. 1991.

[19] KRISHNAKUMAR, Kalmanje. *Micro – genetic algorithms for stationary and non – stationary function optimization.* Intellogent control and adaptive systems. Vol. 1196. International Society for Optics and Photonics. 1990.

[20] SYSWERDA, Gilbert. *A Study of Reproduction in Generational and Steady State Genetic Algorithms.* Foundations of Genetic Algorithms. San Mateo, Morgan Kaufman, San Francisco, 1991, pp.94 – 101.

[21] MAYER, D.G, B.P. KINGHORN and A.A. ARCHER. *Differencial evolution – an easy and efficient evolutionary algorithm for model optimisatio, Agricultural Systems.* 2005, pages 315-328. ISSN 0308-521X. [cit. 2021-01-27].

[22] ŠENKEŘÍK, Roman. *Diferenciální evoluce – princip, strategie a moderní varianty.* 4. prednáška z predmetu Evoluční výpočetní techniky. Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, 2020. [cit. 2021-01-27].

[23] KENNEDY, J., R. EBERHART. *Particle swarm optimization: Neural Networks.* 1995 IEEE Internation Conference, IEEE, Nov/Dec 1995, doi: 10.1109/ICNN.1995.488968. [cit. 2021-01-27].

[24] LIM, Chee Peng, LAKHMI C. Jain, SATCHIDANANDA Dehuri. *Innovations in Swarm Intelligence*. 1. edition Berlin: Springer, 253 pages. ISBN 978-3-642-04224-9. [cit. 2021-01-27].

[25] DORIGO, Marco. *Optimization, Learning and Natural Algorithms.* PhD thesis, Politecnico di Milano, Italy, 1992. [cit. 2021-01-29].

[26] DORIGO, Marco and Thomas STÜTZLE. *Ant Colony Optimization.* 2004. MIT Press, 2004. ISBN 0-262-04219-3. [cit. 2021-01-29].

[27] KARABOGA, Dervis. *A comparative study of Artificial Bee Colony algorithm.* 2009. Applied Mathematics and Computatuon 214, 2009, 108 – 132. [cit. 2021-01-29].

[28] MATUŠÍKOVÁ, Marcela. *Metaheristika Artificial Bee Colony (ABC) na riešenie úlohy obchodného cestujúceho (TSP).* 2017. Bakalárska práca. Žilinská Univerzita v Žiline, Fakulta riadenia a informatiky. Vedúci práce doc. Ing. Michal Kohání, PhD. [cit. 2021-01-29].

[29] VOLNÁ, Eva. *Evoluční algoritmy a neurónové sítě* [online]. Ostrava. Ostravská Univerzita, 2012. [cit. 2021-01-31].

[30] ŠENKEŘÍK, Roman. *Evoluční výpočetní techniky – vybrané algoritmy PSO, SOMA, ACO, SS.* 5.1. prednáška z predmetu Evoluční výpočetní techniky. Fakulta aplikované informatiky, Univerzita Tomáše Bati ve Zlíně, 2020. [cit. 2021-01-31].

[31] ZELINKA, Ivan and Donald DAVENDRA. *SOMA – Self-organizing Migrating Algorithm.* 2016. 289 pages. ISBN 978-3-319-28161-2. [cit. 2021-01-31].

[32] ZELINKA, Ivan. *SOMA Algorithm: Self-Organizing Migrating Algorithm* [online]. [cit. 2021-01-31]. Dostupné z: https://ivanzelinka.eu/somaalgorithm.

[33] MOUSAVIRAD, Seyed Jalaleddin, Gerald SCHAEFER and Iakov KOROVIN. 2020. *Color quantisation using self-organizing migrating algorithm.* 2020 GECCO Companion. Dostupné z: https://dl.acm.org/doi/10.1145/3377929.3398124

[34] BUKÁČEK, Michal. *SOMA TicTacToe* [online]. [cit. 2021-01-31]. Dostupné z: https://play.google.com/store/apps/details?id=cz.bukacek.soma_tictactoe

[35] DIEP, Quoc Bao. Self-Organizing Migration Algorithm *Team To Team Adaptive-SOMA T3A.* 2019 IEEE Congress on Evolutionary Computation, IEEE, 2019, 1182-1187. [cit. 2021-02-05].

[36] DIEP, Quoc Bao, Ivan ZELINKA a Swagatam DAS. Self-Organizing Migrating Algorithm Pareto. *Mendel* [online]. Institute of Automation and Computer Science, Brno University of Technology, 2019, **25**(1), 111-120. ISSN 2571-3701. [cit. 2021-02-08]. Dostupné z: doi:10.13164/mendel.2019.1.111

[37] KADAVÝ Tomáš, Michal PLUHÁČEK, Roman ŠENKEŘÍK a Adam VIKTORÍN. 2019. *The Ensemble of Strategies and Perturbation Parameter in Self-organizing Migrating Algorithm Solving CEC 2019 100-Digit Challenge*. 2019 IEEE Congress on Evolutionary Computation, IEEE, 2019, 372-375. [cit. 2021-02-08].

[38] KADAVÝ Tomáš, Michal PLUHÁČEK, Roman ŠENKEŘÍK a Adam VIKTORÍN. 2020. *Self-organizing Migrating Algorithm with Clustering-aided Migration*. 2020 GECCO Companion. [cit. 2021-02-13].

[39] MALLIPEDDI Rammohan and Ponnuthurai Nagaratnam SUGANTHAN. 2010. *Differential Evolution Algorithm with Ensemble of Parameters and Mutation and Crossover Strategies.* Swarm, Evolutionary and Memetic Computing, 2010, Volume 6466 ISBN 978-3-642-17562-6. [cit. 2021-02-13]

[40] YUE C.T., K.V.PRICE, P.N.SUGANTHAN, J.J.LIANG, M.Z.ALI, B.Y.QU, N.H.AWAD and Partha P BISWAS. *Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization.* [online]. [cit. 2021-03-16]. Dostupné z: https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark

[41] HALDER UDIT, Das SWAGATAM a Maity DIPANKAR. *A Cluster-Based Differential Evolution Algorithm With External Archive For Optimization in Dynamic Environments.* 2013 IEEE Transactions on Cybernetics, IEEE, 2013, vol. 43, no. 3, pp. 881-897, doi: 10.1109/TSMCB.2012.2217491. [cit. 2021-03-30].

[42] ELSAYED S. M. and R. A. SARKER. *Differential Evolution with automatic population injection scheme for constrained problems.* 2013 IEEE Syposium on Differential Evolution (SDE), Singapore, 2013, pp. 112-118, doi: 10.1109/SDE.2013.6601450. [cit. 2021-03-30].

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

EVT	Evolučné výpočtové techniky
EA	Evolučný algoritmus
SOMA	Self-Organizing Migrating Algorithm (Samo-organizujúci sa migračný algoritmus)
GA	Genetický algoritmus
DE	Diferenciálna evolúcia
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
T3A SOMA	Team To Team Adaptive Self-Organizing Migrating Algorithm
SOMA CL	Self-Organizing Migrating Algorithm with Clustering-aided Migration
ESP SOMA	The Ensemble Of Strategies and Peturbation Parameter in Self-Organizing Migrating Algorithm

ZOZNAM OBRÁZKOV

ZOZNAM PSEUDOKÓDOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

PI: Použité základné funkcie

PII: Štatistické vyhodnotenie adaptívnych stratégií a porovnanie rôznych verzií algoritmu SOMA. Dostupné v elektronické verzii diplomovej práce a na nasledujúcom odkaze:

https://utbcz-my.sharepoint.com/:u:/g/personal/ailab_fai_utb_cz/Eb8ke2NG5vNGm0mTLonSih4BgleGffPypT8GwIUjqUgZg?e=OBk5SE

PIII: CD so zdrojovými kódmi, dátovými súbormi a spracovanou štatistikou

PRÍLOHA P I: POUŽITÉ ZÁKLADNÉ FUNKCIE

V nasledujúcom texte sa nachádza matematický zápis jednotlivých základných funkcií, ktoré sú využité v benchmarku CEC20 buď samostatne alebo na skladanie hybridných, či kompozitných funkcií.

1. Funkcia Bent Cigar

$$f_1(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$$

2. Funkcia Rastrigin

$$f_2(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

3. Funkcia High Conditioned Elliptic

$$f_3(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2$$

4. Funkcia HGBat

$$f_4(x) = \left| \left(\sum_{i=1}^{D} x_i^2 \right)^2 - \left(\sum_{i=1}^{D} x_i \right)^2 \right|^{1/2} + \frac{\left(0.5 \sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right)}{D} + 0.5$$

5. Funkcia Rosenbrock

$$f_5(x) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right)$$

6. Funkcia Griewank

$$f_6(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

7. Funkcia Ackley

$$f_7(x) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} - exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)\right) + 20 + e$$

8. Funkcia Happycat

$$f_8(x) = \left|\sum_{i=1}^{D} x_i^2 - D\right|^{1/4} + \frac{\left(0.5\sum_{i=2}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right)}{D} + 0.5$$

9. Funkcia Discus

$$f_9(x) = 10^6 x_i^2 + \sum_{i=1}^{D} x_i^2$$

10. Funkcia Lunacek bi – Rastrigin

$$f_{10}(x) = min\left(\sum_{i=1}^{D}(\hat{x_i} - \mu_0)^2, dD + s\sum_{i=1}^{D}(\hat{x_i} - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D} cos(2\pi\hat{z_i})\right)$$

$$\mu_0 = 2.5,\ \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, d = 1$$

$$y = \frac{10(x-0)}{100}, \frac{x_i}{x_i} = 2sign(x_i^*)y_i + \mu_0, for\ i = 1, 2, \dots, D$$

$$z = \Lambda^{100}(\hat{x} - \mu_0)$$

11. Modifikovaná funkcia Schwefel

$$f_{11}(x) = 418.9829 * D - \sum_{i=1}^{D} g(z_i)$$

$$z_i = x_i + 4.209687462275036e + 002$$

$$g(z_i) = \begin{cases} z_i sin\left(|z_i|^{\frac{1}{2}}\right) & ak\ |z_i| \le 500 \\ \left(500 - mod(z_i, 500)\right)sin\left(\sqrt{|500 - mod(z_i, 500)|}\right) - \frac{(z_i - 500)^2}{10000D} & ak\ z_i > 500 \\ (mod(|z_i|, 500) - 500)sin\left(\sqrt{|mod(|z_i|, I, 500) - 500|}\right) - \frac{(z_i + 500)^2}{10000D} & ak\ z_i < -500 \end{cases}$$

12. Rozšírená funkcia Schaffer

$$g(x,y) = 0.5 + \frac{\left(sin^2\left(\sqrt{x^2+y^2}\right) - 0.5\right)}{\left(1+0.001(x^2+y^2)\right)^2}$$

$$f_{12}(x) = g(x_1,x_2) + g(x_2,x_3) + \cdots + g(x_{D-1},x_D) + g(x_D,x_1)$$

13. Rozšírená funkcia Rosenbrock plus Griewangk

$$f_{13}(x) = f_6(f_5(x_1,x_2)) + f_6(f_5(x_2,x_3)) + \cdots + f_6(f_5(x_{D-1},x_D)) + f_6(f_5(x_D,x_1))$$

14. Funkcia Weierstrass

$$f_{14}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{kmax}\left[a^k cos\left(2\pi b^k(x_i+0.5)\right)\right]\right) - D\sum_{k=0}^{kmax}\left[a^k cos\left(2\pi b^k(x_i+0.5)\right)\right]$$

$$a = 0.5, b = 3, kmax = 20$$