Nástroj pro identifikaci slabých míst evolučních algoritmů

Bc. Martin Hazda

Diplomová práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin Hazda**
Osobní číslo:	**A20612**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Kybernetická bezpečnost**
Forma studia:	**Kombinovaná**
Téma práce:	**Nástroj pro identifikaci slabých míst evolučních algoritmů**
Téma práce anglicky:	**A Tool For Identifying Weaknesses In Evolutionary Algorithms**

Zásady pro vypracování

1. Vypracujte literární rešerši na dané téma.
2. Sestavte množinu vhodných testovacích problémů.
3. Navrhněte metodiku pro testování a vyhodnocení výsledků.
4. Zaměřte se na analýzu diverzity populace a vznik clusterů v populaci.
5. Otestujte vybrané moderní evoluční algoritmy a zhodnoťte získané poznatky.

Forma zpracování diplomové práce: **Tištěná**

Seznam doporučené literatury:

1. ZELINKA, Ivan. Evoluční výpočetní techniky: principy a aplikace. Praha: BEN – technická literatura, 2009. ISBN 978-80-7300-218-3.
2. SALA, Ramses a Ralf MULLER. Benchmarking for Metaheuristic Black-Box Optimization: Perspectives and Open Challenges. In: 2020 IEEE Congress on Evolutionary Computation (CEC) [online]. IEEE, 2020, 2020, s. 1-8 [cit. 2021-12-01]. ISBN 978-1-7281-6929-3. Dostupné z: doi:10.1109/CEC48606.2020.9185724
3. AHRARI, Ali, Mohammad R. SAADATMAND, Masoud SHARIAT-PANAHI a Ali A. ATAI. On the limitations of classical benchmark functions for evaluating robustness of evolutionary algorithms. Applied Mathematics and Computation [online]. 2010, 215(9), 3222-3229 [cit. 2021-12-01]. ISSN 00963003. Dostupné z: doi:10.1016/j.amc.2009.10.009
4. MORRISON, Ronald W. a Kenneth A. DE JONG. Measurement of Population Diversity. COLLET, Pierre, Cyril FONLUPT, Jin-Kao HAO, Evelyne LUTTON a Marc SCHOENAUER, ed. Artificial Evolution [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, 2002-4-10, s. 31-41 [cit. 2021-12-01]. Lecture Notes in Computer Science. ISBN 978-3-540-43544-0. Dostupné z: doi:10.1007/3-540-46033-0_3
5. VIKTORIN, Adam, Roman SENKERIK, Michal PLUHACEK a Ales ZAMUDA. Steady success clusters in Differential Evolution. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI) [online]. IEEE, 2016, 2016, s. 1-8 [cit. 2021-12-01]. ISBN 978-1-5090-4240-1. Dostupné z: doi:10.1109/SSCI.2016.7850252

Vedoucí diplomové práce: **Ing. Michal Pluháček, Ph.D.**
Ústav automatizace a řídicí techniky

Konzultant diplomové práce: **Ing. Adam Viktorin, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **3. prosince 2021**
Termín odevzdání diplomové práce: **23. května 2022**

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

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

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

Ve Zlíně dne 24. ledna 2022

Martin Hazda:
Nástroj pro identifikaci slabých míst evolučních algoritmů:
Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně, dne 18.5.2022 Martin Hazda v.r.
podpis diplomanta

ABSTRAKT

Cílem práce bylo vytvoření nástroje pro identifikaci slabých míst evolučních algoritmů s důrazem na popsání vnitřní dynamiky algoritmu v kontextu jeho výkonnosti. Teoretická část obsahuje uvedení do problematiky evolučních algoritmů a rešerši posledních poznatků z oblasti evolučních algoritmů. Byla sestavena sada testovacích problémů a následně analyzováno chování instancí evolučních algoritmů při jejich řešení pomocí vytvořeného softwarového nástroje. Bylo sledováno vytváření clusterů v populaci, změny její diverzity, konvergenční trajektorie, dynamika pohybu jedinců v prohledávaném prostoru a robustnost algoritmu. Výstupy analýzy upozorňující na odhalené potenciální slabiny jsou uvedeny v závěrečné části práce.

Klíčová slova: evoluční algoritmus, diverzita populace, clustery, konvergence, prohledávaný prostor, analýza

ABSTRACT

The aim of the thesis was to create a tool for identifying weaknesses in evolutionary algorithms with an emphasis on describing the internal dynamics of the algorithm in the context of its performance. The theoretical part contains an introduction to evolutionary algorithms and the research of the latest knowledge of evolutionary algorithms. A set of test problems were set up and the behavior of instances of evolutionary algorithms was analyzed by the created tool as the solution was being produced. The analysis was focused on creating clusters in the population, changes in its diversity, convergence trajectory, dynamics of individuals' movement in the search space and the robustness of the algorithm. Analysis outputs discuss potential weaknesses and are presented in the final part of the thesis.

Keywords: evolutionary algorithm, population diversity, clusters, konvergence, search space, analysis

Rád bych poděkoval vedoucímu mé diplomové práce Ing. Michalu Pluháčkovi Ph.D., za odborné vedení, trpělivost, podporu, vstřícnost a užitečné rady. Děkuji své ženě Barboře za trpělivost, podporu a shovívavost při tvorbě této práce.

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

OBSAH

ÚVOD

Cílem této práce bylo vytvoření nástroje pro identifikaci slabých míst evolučních algoritmů, pomocí analýzy vybraných aspektů vnitřní dynamiky algoritmu. K naplnění tohoto cíle bylo nezbytné stanovit množinu jevů, jejichž analýza je předmětem činnosti nástroje, aplikace, a popsat tyto jevy spolu s jejich teoretickými základy. Úvodem do problematiky je v teoretické části popis základní terminologie zkoumané oblasti, a základních přístupů k řešení optimalizačních problémů pomocí evolučních algoritmů. Východiskem pro popis procesů spojených s hledáním globálního optima je popis genetických algoritmů, jejichž principy selekce, křížení a mutace rezonují současnými metodami řešení optimalizačních problémů. V návaznosti na genetické algoritmy jsou v teoretické části popsány další základní směry, jejichž představiteli jsou DE (Differential Evolution), PSO (Particle Swarm Optimizer) a SOMA (Self-Organizing Migrating Algorithm), a byly představeny jejich pokročilé varianty. Testování výkonnosti algoritmů představuje doplňkový aspekt při analýze vedoucí k identifikaci slabých míst algoritmů, a je mu věnována samostatná kapitola teoretické části. Hodnocení výkonnosti instancí algoritmů tvoří kriterium analýzy průběhů konvergencí pomocí aplikace pouze z hlediska úspěchu nebo neúspěchu konvergence, nebo jako indikátor její stagnace. Cílem analýzy je pokusit se odhalit příčiny neschopnosti algoritmů nalézat lepší řešení v souvislosti s dynamikou pohybu jedinců po hyperploše, tvorbou clusterů v populaci a vývojem její diverzity. Praktická část práce pojednává o funkcích aplikace, její struktuře, uživatelských odezvách, a definuje požadované parametry datových vstupů. Popisuje způsob, jakým jsou řešeny jednotlivé kroky analýzy, a definuje parametry výstupů aplikace. Záznamy konvergencí a populací, tvořících vstup aplikace, byly vytvořeny algoritmy DE, PSO, a SOMA T3A (SOMA Team To Team Adaptive), na sadě testovaných problémů, které jsou definovány v této části práce, přičemž výstupy pokročilé varianty SOMA jsou využity jako etalon k porovnání s výstupy DE a PSO. Zdrojový kód aplikace tvoří přílohu této práce, výstupy aplikace jsou záznamy tvorby clusterů v jednotlivých běžích algoritmů, diverzity populací a velikosti průměrných pohybů, skoků, jedinců v jednom opakování. Výstupy představují strukturovaný a vizualizovaný popis chování instancí evolučních algoritmů na zvolených optimalizačních problémech v kontextu nalézaných řešení. Závěrečná část praktické části je věnována interpretaci výsledků analýzy pro trojici algoritmů na definovaných testovacích funkcích, a ve vybraných příkladech k identifikaci slabých míst evolučních algoritmů.

TEORETICKÁ ČÁST

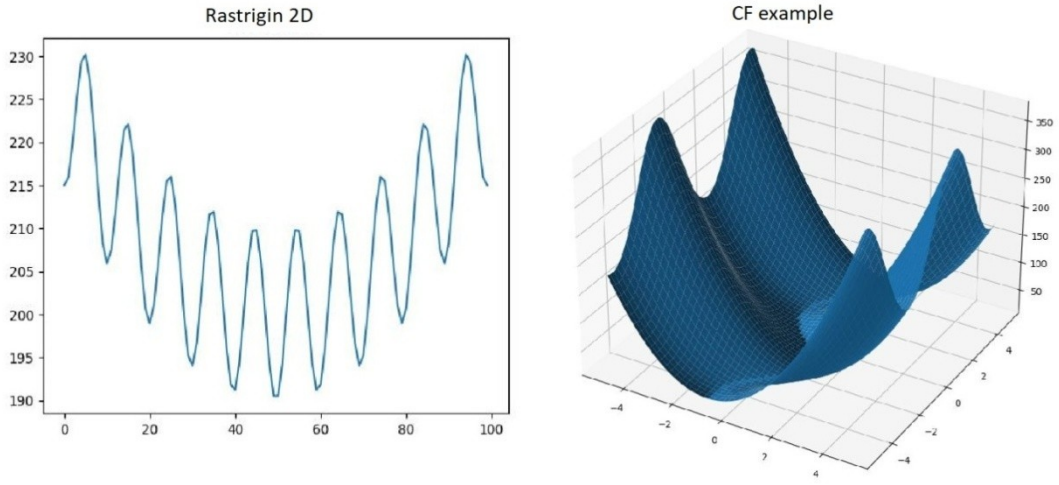
1 EVOLUČNÍ ALGORITMY

Evoluční algoritmy (EA) jsou iterativní soubory procesů, ve kterých dochází ke změnám v uspořádání skupiny možných řešení optimalizačního problému za účelem postupného nalezení jeho globálního extrému. Změny v uspořádání nastávají působením definovaných deterministických a stochastických vlivů, a jsou inspirovány přírodními evolučními procesy přirozeného výběru, selekcí, křížením a schopností přes-generační adaptace potomků na změny prostředí, mutací [1]. Proces hledání optimálního řešení problému je hledáním nejvýhodnější kombinace jeho vstupů, při použití atributů daného jedince. Optimum reálného problému lze jen obtížně nalézt analytickými nebo numerickými metodami s ohledem na výpočetní náročnost nebo proveditelnost, nalezení globálního extrému prostřednictvím EA je proto aproximací nejlepších řešení reálných optimalizačních problémů [2]. Životní cyklus EA je zahájen inicializací, tvorbou skupiny možných řešení, jejich evolucí a průběžným vyhodnocováním vhodnosti řešení do okamžiku splnění kritéria pro zastavení evolučního procesu. Metodiky těchto kroků jsou závislé na typu EA. Evoluční procesy jsou ovlivňovány nastavením řídících parametrů, jejichž míra uplatnění závisí na komparaci pseudonáhodně generovaných čísel s konstantami, jedná se tedy o pravděpodobnostní jevy související s Bernouliovskými experimenty s výstupem typu úspěch nebo neúspěch [3]. Mezi klasické EA patří genetické algoritmy (Genetic Algorithms, GA), evoluční strategie (Evolution strategies, ES) a diferenciální evoluce (Differential Evolution, DE) [4]. Z pohledu moderních optimalizačních přístupů lze mezi EA zařadit také metody inspirované chováním skupin živočichů, zejména v hejnech. Významnými zástupci hejnových metod jsou samo-organizující se migrační algoritus (Self-Organizing Migrating Algorithm, SOMA) a algoritmy rojení částic (Particle Swarm Optimizer, PSO) [4]. EA najdou uplatnění například při trénování neuronových sítí za účelem aproximace odhadu parametrů materiálních modelů z experimentálních výsledků, topologických optimalizacích, plánování logistiky, robotizované výrobě, stanovení mechanických parametrů stavebních a montážních prvků, optimalizaci toku tepla a redukci ztrát tepelných soustav, nebo provádění regresních analýz [5].

1.1 Účelová funkce

Optimalizační problém je matematicky popsán a reprezentován účelovou funkcí (Cost Function), a D rozměrným prostorem možných řešení, ve kterém se nachází globální

extrém, optimum. Kvalita každého řešení je matematicky vyjádřena jako funkční hodnota účelové funkce na *D* rozměrné ploše ležící v *D+1* rozměrném prostoru [4]. V souvislosti s typem problému jsou ekvivalentními pojmenováními cílová, kriteriální, ztrátová nebo nákladová funkce. Dimenzemi *D* jsou vyjádřeny argumenty účelové funkce a odpovídají množství parametrů reálných problémů. Optimalizační problémy mohou být jednokriteriální (Single-objective) ve kterých je hledáno nejlepší řešení vzhledem k jednomu kritériu, veličině, nebo vícekriteriální (Multi-objective), ve kterých jsou hledaná řešení kompromisem mezi jednotlivými kritérii a mohou tvořit pareto množiny, ve kterých platí, že změnou kteréhokoliv argumentu jednoho kriteria ve směru vhodnějších řešení dojde vždy ke snížení vhodnosti u argumentů konkurenčního kritéria [6]. Prostor, hyperplocha, má obvykla stanoveny hranice (Boundaries) a může také obsahovat penalizované oblasti, ve kterých není přípustné nebo účelné hledat optima (Constraints) [1][4], představující omezení argumentů účelové funkce. Překročením hranic prostoru nebo penalizovaných oblastí nastávají krizové stavy. Pokud mechanizmy EA umožní pohyb jedince ve směru za hranice povoleného prostoru, je využita některá z hraničních strategií (Boundary Strategies). Jedinec je zastaven na hranici a jeho poloha bude použita jako výchozí při další iteraci (Clipping), náhodně opětovně vygenerován jako nový jedinec uvnitř povolené oblasti (Random), velikost části skoku uvnitř zakázané oblasti je aplikována opačným směrem (Reflection), nebo algoritmus jedince po dosažení hranice umístí na protilehlý počátek prohledávaného prostoru a velikost části skoku uvnitř zakázané oblasti je aplikována v původním směru (Periodic) [4]. Jedinci vyskytující se uvnitř prostoru omezeného argumenty účelové funkce nemusí být přemísťováni v souladu s hraničními strategiemi, mohou setrvat ve svých pozicích, avšak jejich účast na evolučním procesu je penalizována přičítáním vysokých hodnot k nalezeným řešením [4]. Toto znevýhodnění často znamená jejich nahrazení novými jedinci prostřednictvím mechanizmů evolučních procesů.



Obrázek 1: Příklady účelových funkcí

Optimalizační problémy jsou definovány jako spojité nebo diskrétní funkce, nebo jejich kombinace, jejichž optimum je nalézáno v globálních extrémech jako minimum nebo maximum [4]. Účelová funkce může být multimodální, může obsahovat více lokálních extrémů. Tvar hyperplochy je ovlivněn separabilitou problému, asymetrií, výskytem šumu nebo plochami v okolí extrémů [1]. Globální optimum může být také reprezentováno celou plochou. Úkolem EA je nalezení globálního optima na účelové funkci její minimalizací nebo maximalizací, v závislosti na typu optimalizačního problému. Obecné matematické vyjádření minimalizace / maximalizace účelové funkce s argumenty proměnných X [1] [3] [4]:

$$min/max\ f_{cost}(X) \tag{1}$$

1.2 Populace

Skupina N možných řešení vyskytující se ve stejný okamžik v prostoru možných řešení představuje populaci, generaci jedinců, vektorů, jejichž atributy počtem odpovídají dimenzi problému D [1][4]. Velikost populace a dimenzionalita problému jsou řídícími parametry EA. Obecný zápis N jedinců v podobě vektorů s atributy [1]:

$$X_1 = (x_1, x_2, x_3 \dots x_D)$$

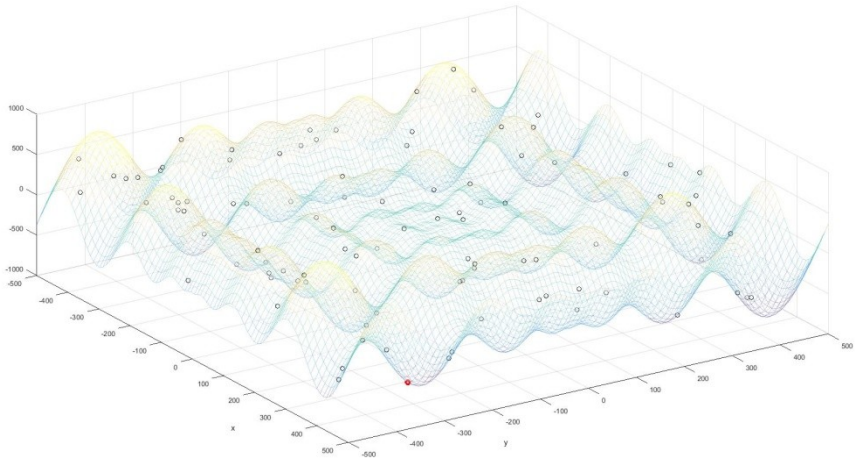
$$X_2 = (x_1, x_2, x_3 \dots x_D)$$

$$X_3 = (x_1, x_2, x_3 \dots x_D)$$

....

$$X_N = (x_1, x_2, x_3 \dots x_D) \quad (2)$$

Počáteční populace je ve většině případů generována pseudonáhodně s rovnoměrným rozložením, za dodržení podmínek existence uvnitř povoleného, ohraničeného prostoru [1]. Po každé iteraci algoritmu jsou jedinci vstupní generace nahrazováni úspěšnějšími nebo stejně kvalitními jedinci výstupní generace na základě zvolené populační strategie. Rodiče μ jsou nahrazeni potomky λ, nebo v případě uplatnění principu elitismu [4] jsou do nové generace vybíráni nejvhodnější jedinci ze skupiny rodičů a zároveň ze skupiny potomků [1].

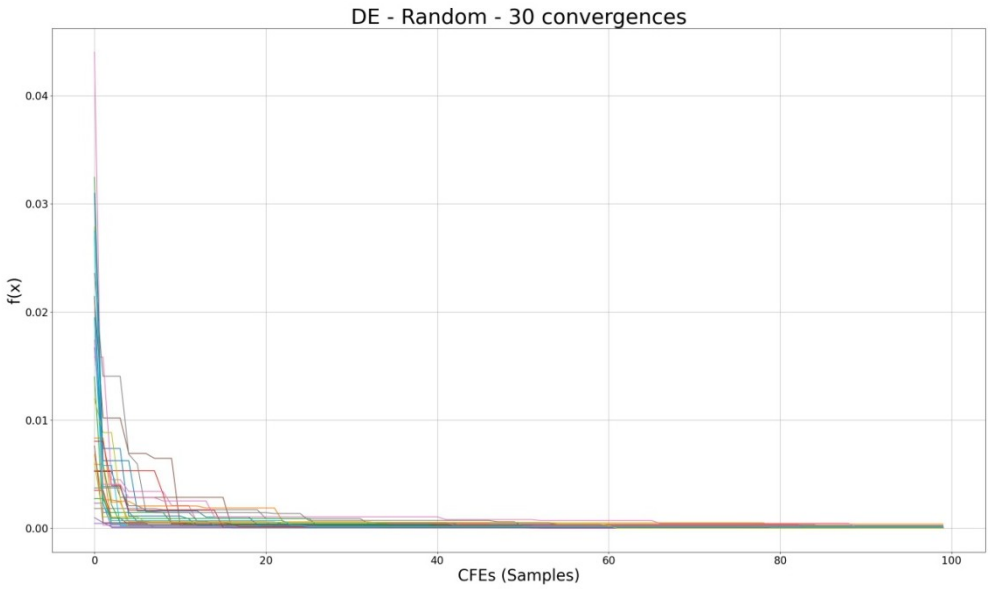


Obrázek 2: 3D reprezentace rozmístění 2D jedinců na hyperploše CF Schwefel

Při zpracovávání evolučních výpočetních technik pomocí výpočetní techniky, a v závislosti na definici účelové funkce, jsou atributy reprezentovány různými datovými typy, reálnými čísly, boolean operátory, binárními hodnotami nebo celými čísly [1][4]. V této práci budou analyzována pouze data jednokriteriálních optimalizačních problémů.

1.3 Konvergence

Konvergence řešení algoritmu představuje směřování doposud získaných ohodnocení účelové funkce k řešením blízkým globálnímu optimu v celé populaci. Je reprezentována křivkou nejlepších nalezených řešení, přičemž její průběh může být konstantní po dobu odpovídající iteracím, v nichž není nalezeno vhodnější řešení. Proces přibližování ke globálnímu extrému při běhu algoritmu musí mít vždy konvergenční charakter [4] v celém jeho průběhu. Každý běh je ukončen v závislosti na definici ukončovací podmínky, kterou lze stanovit jako uspokojivou míru dosažené kvality přiblížení optimu nebo vyčerpání povoleného množství ohodnocení účelové funkce (Cost Function Evaluation, CFE, *f(x)*) daného algoritmu (Budget). Vzhledem k existenci stochastických vlivů u EA jsou získávány pro každý jeho běh různé průběhy. Ukončovací podmínka je řídícím parametrem EA [1][4].



Obrázek 3: Příklad průběhu konvergencí EA pro 30 opakování

1.4 Explorace a exploitace

Multimodalita implikuje nebezpečí uváznutí algoritmu a předčasnou konvergenci. Proto je nezbytné řídící parametry EA volit s ohledem na vyrovnaný podíl schopnosti prohledávat nové oblasti prostoru možných řešení, explorace, a schopnosti reagovat na signály přítomnosti možných optim, exploitace. Rovnováha těchto dvou vlastností je zásadní ochranou proti uváznutí v lokálních extrémech a předčasné konvergenci [1] [4].

1.5 Selekce

Vstupem obecného evolučního procesu v EA je definovaný počet jedinců populace, rodičů, kteří se podílejí na tvorbě jedinců nové generace. Noví jedinci poskytují stejnou nebo lepší kvalitu řešení optimalizačního problému [1]. V závislosti na typu EA jsou vybíráni rodiče na základě vlastní kvality (Fitness), nebo jsou vybíráni náhodně pomocí pravděpodobnostních operací matematickou komparací deterministicky stanovené hodnoty s hodnotou pseudonáhodně vygenerovanou v definovaném rozložení pravděpodobnosti [1]. Oba přístupy lze kombinovat vybráním většího než potřebného počtu rodičů s nejvyšší kvalitou, z nichž dojde ke konečnému výběru pravděpodobnostními metodami. Pořadí operace selekce odpovídá použitému typu EA. Pouhá selekce náhodných jedinců nebo těch nejlepších z populace bez implementace dalších evolučních operací algoritmu by odpovídala metodám heuristik náhodného prohledávání prostoru možných řešení a lokálního prohledávání [1] [4].

1.6 Křížení

Jedinci nové generace mohou být tvořeni kombinacemi částí vybraných jedinců staré generace. Křížení (Crossover) představuje deterministicky a stochasticky řízené nahrazení definovaného počtu atributů vybraného jedince atributy jiného vybraného nebo vytvořeného jedince [1]. Deterministický mechanismus určuje rozsah a počet atributů rodičů, ze kterých bude kombinován potomek, stochasticky je pak stanoveno, s jakou pravděpodobností dojde k uplatnění konkrétního atributu [5]. Metodika použití křížení je závislá na typu EA. Křížení je řídícím parametrem EA a může mít pevnou hodnotu pro celý běh algoritmu nebo může být dynamicky upravována jinými řídícími parametry pro dosažení rovnováhy mezi explorací a exploitací [5]. Vhodnost potomka vzniklého křížením není úměrná pouze kvalitě rodičů, ale také jednotlivým jejich atributům, které mohou mít vlastní relativní kvalitu vzhledem k extrémům účelové funkce. Při přísně deterministickém výběru nejkvalitnějších částí jedinců pro křížení hrozí předčasná konvergence v lokálním extrému a algoritmus bude mít exploitativní sklony. Pro vyrovnané křížení je tedy nutné zapojit také stochastické vlivy [1] [4].

1.7 Mutace

V průběhu této operace dojde k pravděpodobnostně řízené změně vlastností jedince nahrazením některých jeho atributů jinými a vytvoření odvozené varianty jedince [4].

V závislosti na typu EA je proces mutace proveden vektorovými operacemi nebo binárními změnami. Pravděpodobnost mutace je řídícím parametrem EA a může mít pevnou hodnotu pro celý běh algoritmu nebo může být dynamicky upravována jinými řídícími parametry pro dosažení rovnováhy mezi explorací a exploitací, přičemž vysoký vliv mutace na vznikající jedince podporuje exploraci, nižší exploitaci [1] [4].

2 VYBRANÉ GENERICKÉ ALGORITMY

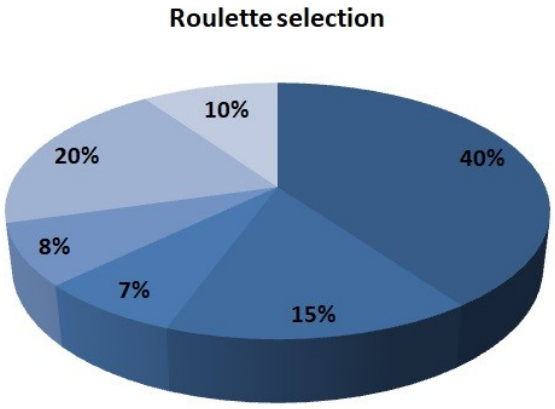
Při řešení optimalizačních problémů lze obecně využít kterýkoliv vhodný typ EA v jeho generické nebo modifikované podobě. Modifikací dochází k utilizaci EA na určitý typ řešeného problému. Principiálně lze rozlišit několik základních větví, lišících se pořadím operací selekce, křížení, mutace a systémem uplatnění těchto operací v rámci jednoho běhu [4][5]. Existence mnoha typů EA v generických i modifikovaných variantách vychází z principu No Free Lunch Theorem (NFLT) [7], který odráží potřebu užití konkrétních EA na konkrétní optimalizační problémy.

2.1 Genetické algoritmy GA

Základní myšlenka GA rezonuje ostatními nástroji vyvíjenými a uplatňovanými k optimalizačním účelům. Inspirace biologickými systémy je v GA promítnuta v životním cyklu kvality jedinců, která je pomocí selekce, křížení a mutace předávána, iterativně rekombinována a zdokonalována ve formě genetické informace, chromozomu. Jedinci populace, chromozomy, jsou zapisováni v binární podobě a jejich atributy, geny, reprezentují přímo binární hodnoty možného řešení optimalizačního problému, zástupné hodnoty reálných čísel nebo hodnoty celých čísel [4]. Binární zápis jedince je vhodný pro řešení diskrétních optimalizačních úloh, například pro problém batohu nebo obchodního cestujícího. Počáteční generace jedinců je vytvořena uvnitř prohledávaného prostoru uplatněním pseudonáhodných procesů s rovnoměrným rozložením pravděpodobnosti. Může být také inicializována na základě výstupů předešlých heuristických analýz problému, což však může mít nepříznivý vliv na diverzitu populace a tím i na výsledný průběh konvergence. Pro každého jedince je vypočtena fitness hodnota, na jejímž základě vstupují jedinci do procesu selekce [1] [4].

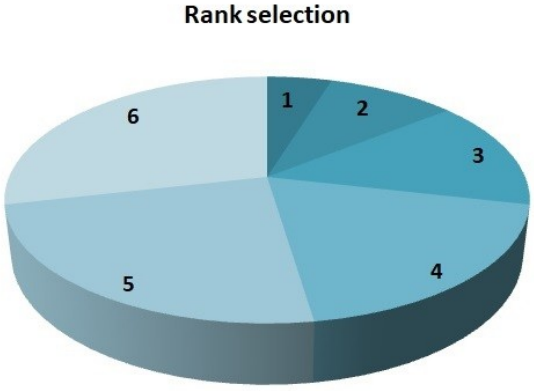
2.1.1 Selekce

Při použití Ruletového pravidla (Roulette) výběru jsou hodnoty fitness uspořádány do řady představující součet všech fitness v populaci, a v této řadě je náhodně vybrán bod, který je ukazatelem na rodiče. Pravděpodobnost výběru rodiče je tedy přímo úměrná hodnotě fitness jedinců.



Obrázek 4: Ruletové pravidlo selekce
s procentuálním vyjádřením
pravděpodobnosti výběru

Variantou ruletového výběru je univerzální stochastické vzorkování (Stochastic Universal Sampling), při kterém jsou náhodným výběrem dvou bodů vybíráni v řadě fitness jedinců najednou dva rodiče. Ke zmírnění vlivu rozdílů velikostí fitness v populaci lze využít metodu výběru dle pořadí (Rank Selection), při níž je rodič vybírán náhodným bodem v řadě, která je tvořena rovnoměrně odstupňovanými reprezentanty pořadí fitness hodnot napříč populací.

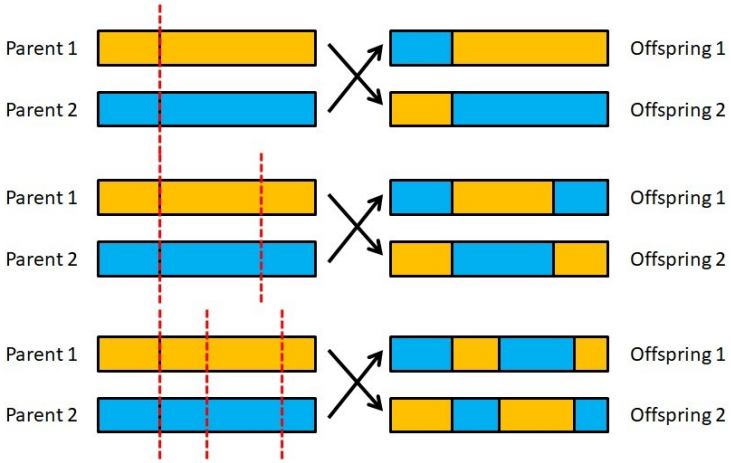


Obrázek 5: Pravidlo Rank Selection
s pořadím jedinců podle fitness

Odlišný přístup k procesu selekce představuje výběr turnajem (Tournament), ve kterém je náhodně vybráno k jedinců, ze kterých je ten s nejlepší hodnotou fitness vybrán jako rodič. Aby nedocházelo křížením a mutací ke ztrátě nejlepších jedinců populace, může být do evolučních procesů promítnut elitismu [4], tedy mechanismus výběru k nejvhodnějších řešení, která jsou bez dalších operací zařazena do další generace [1].

2.1.2 Křížení

Jedinci vybraní jako rodiče vstupují v dalším kroku do procesu křížení, ve kterém dochází ke kombinaci jejich částí za účelem tvorby potomků. Operátor křížení slouží k určení, které části kterého jedince budou vyměněny, sloučeny a využity jako nové chromozomy. Velikost pravděpodobnosti křížení se obvykle pohybuje v rozmezí 75 až 95% [8]. Chromozom rodiče lze rozdělit jedním nebo několika body na dvě a více částí určených k rekombinaci, a to prostřednictvím deterministicky nebo stochasticky řízených pravidel.

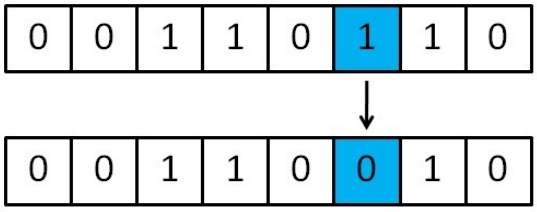


Obrázek 6: Jednobodové, dvoubodové a vícebodové křížení v GA

Metodou rovnoměrného křížení jsou prostřednictvím pravděpodobnostního výběru dedikovány jednotlivé geny rodičů konkrétním potomkům, z nichž jsou sestavovány. Prostřednictvím aritmetické rekombinace jsou jednotlivé geny potomků stanovovány na základě vážených průměrů jednotlivých genů rodičů [1].

2.1.3 Mutace

V chromozomu jedince je pravděpodobnostními procesy vybrán náhodný atribut, u kterého dojde k jeho změně. Pravděpodobnost křížení je obvykle 1% [8]. Velikost změny je vázána s datovým typem konkrétního atributu jedince. V případě reálných čísel dojde k nahrazení pseudonáhodně vygenerovaným číslem v rozsahu stanoveném řídícími podmínkami, u binárního zápisu dojde k záměně 1 za 0 a naopak, což ale může vést ke skokové změně hodnoty celého jedince, proto je vhodné využít zástupné kódování, kde změna jednoho bitu chromozomu znamená malou změnu jedince [1].



Obrázek 7: Jednobodová mutace v GA

2.2 Samo-organizující se migrační algoritmus SOMA

Algoritmus je řazen mezi memetické algoritmy [9] s hejnovou inteligencí (Swarm inteligence) [4] a je založen na soutěžení a kooperaci jedinců při jejich pohybu, migraci po hyperploše v souladu s řídícími parametry a zvolené strategii. Počáteční generace je generována v prostoru možných řešení pseudonáhodně s rovnoměrným rozložením pravděpodobnosti. V průběhu iterace algoritmu, migračního kola, je vyhodnocena fitness všech jedinců a podle zvolené strategie se jedinci po trase definované délky *PathLength* pohybují po krocích *Step* směrem k lepším řešením. Směr pohybu jedinců je ovlivňován rušením (Perturbance), reprezentovaného parametrem *PRT*, který spolu s velikostí populace *PopSize*, délkou pohybu *PathLength* a krokem *Step* tvoří řídící parametry SOMA [1][4][9]. Ukončovacím parametrem je povolený počet migračních kol algoritmu, nebo dosažení uživatelsky volené konstanty minimální vzdálenosti mezi nejlepším a nejhorším řešením v rámci jednoho migračního kola [1] [4].

2.2.1 Migrační kolo

Na základě parametru *PRT* je v migračních kolech vypočítáván pro každého aktivního jedince zvlášť pertrubační vektor *PRTVector*. Ten je vytvořen matematickou komparací hodnoty *PRT* s hodnotou pseudonáhodně vygenerovanou v definovaném rozložení pravděpodobnosti, a zapsán v binární podobě s počtem atributů odpovídající počtu atributů jedince. Rušení pohybu jedinců posiluje diverzitu populace a lze jej chápat jako analogii mutace, hodnota parametru *PRT* jako ekvivalent prahu křížení standardních EA [4]. Pohyb jedince k lepším řešením je pak realizován jen v atributech, dimenzích, pro které je odpovídající atribut *PRTVector* roven 1, ostatní atributy zůstávají neměnné [9].

$$for\ j\ \in (0, 1, \ldots dimension)$$

$$if\ rand_j < PRT: PRTVector_j = 1 = Move;\ else\ PRTVector_j = 0 = Freeze; \quad (3)$$

Tabulka 1: Příklad uplatnění PRT

Atribut	0	1	2	3	4
rand	0,560098	0,275579	0,496681	0,213485	0,181261
PRT	0,3	0,3	0,3	0,3	0,3
rand<PRT	Nepravda	Pravda	Nepravda	Pravda	Pravda
PRTVector	0	1	0	1	1
Jedinec	0,738155	2,593896	1,017954	1,331127	1,206731
Pohyb/Stůj	**Stůj**	**Pohyb**	**Stůj**	**Pohyb**	**Pohyb**

Pohyb jedince probíhá po diskrétních krocích *Step* od 0 do hodnoty *PathLength*, přičemž délka pohybu a velikost kroku jsou zvoleny tak, aby v každé fázi pohybu nedocházelo k překrytí přesné pozice lepšího řešení [9], ale aby tato pozice byla vždy překročena o poměrnou část. Pro každý krok je vypočtena funkční hodnota CFE a po ukončení pohybu je jedinec přesunut do místa nejlepšího nalezeného řešení.

2.2.2 Migrační strategie

Strategiemi je v SOMA stanoveno, které jedince mají aktivní jedinci následovat v migračním kole *Migration*. Základní variantou je *všichni k jednomu* (AllToOne) [9], ve které je zvolen jedinec s nejlepší hodnotou fitness a ten je označen za vůdce (Leader) a všichni ostatní jedinci jej následují a pohybují se po hyperploše směrem k němu. V každém dalším migračním kole je určován nový vůdce. Strategie *všichni ke všem* (AllToAll) [9] stanovuje pohyb každého jedince ke každému jinému jedinci. Dochází k výpočtu velkého množství CFE, a tím i nárůstu výpočetní náročnosti, je však více explorativní a globální extrém může být nalezen s vyšší pravděpodobností [4]. Počet CFE v rámci migračního kola odpovídající strategii SOMA:

$$CFE_{AllToOne} = \frac{(PopSize - 1) \cdot PathLength \cdot Migrations}{Step} \quad (4)$$

$$CFE_{AllToAll} = \frac{PopSize \cdot (PopSize - 1) \cdot PathLength \cdot Migrations}{Step} \quad (5)$$

Modifikacemi uvedených strategií jsou varianty *všichni k jednomu náhodně zvolenému* (AllToOneRand), ve které je vůdce vybírán z populace pro každé migrační kolo náhodně, a *adaptivní všichni ke všem* (AllToAllAdaptive), při níž je vyhodnocováno a aktualizováno nejlepší nalezené řešení po každé migraci jedince k jinému jedinci [4].

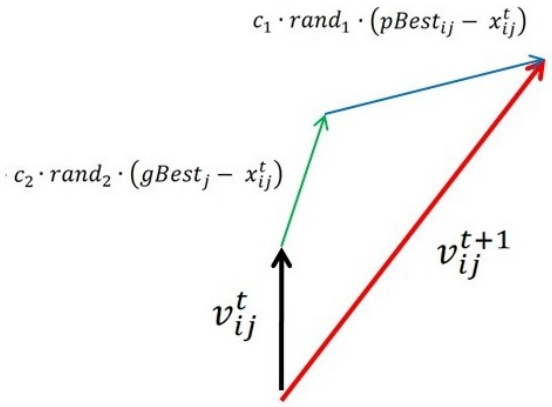
2.3 Particle swarm optimizer PSO

Algoritmus je inspirován hejnovou inteligencí (Swarm intelligence) [4] v přírodě, využívá kooperaci jedinců, částic (Particle), a sdílení znalostí o nalezených řešeních v populaci [4][10]. Počáteční generace je generována pseudonáhodně s rovnoměrným rozložením pravděpodobnosti v prostoru možných řešení. V průběhu hledání globálního optima se jedinci pohybují po hyperploše rychlostí *v*, ve směru lepších řešení. Disponují pamětí vlastních, doposud nejlepších nalezených řešení *pBest*, zároveň je zaznamenáváno globální, doposud nejlepší nalezené řešení v rámci celé populace *gBest* [1]. Vektor nové rychlosti pohybu jedince je ovlivněn jeho rychlostí v předešlé iteraci, rozdílem vektoru aktuální pozice jedince *x* od *pBest* a *gBest*. Po inicializaci populace na začátku běhu algoritmu je rychlost nastavena na hodnotu $v_0 = 0$, nebo generována pseudonáhodně [11]. Váha, setrvačnost, *w* (Inertia weight) představuje koeficient zapomínání rychlosti, a vyjadřuje velikost vlivu původní rychlosti a novou [4]. Hodnota setrvačnosti může být mezi iteracemi upravována, nebo může mít konstantní hodnotu po celou dobu běhu Algoritmu. Prioritní faktor c_1 reprezentuje tendence jedince k návratu k vlastnímu nejlepšímu nalezenému řešení, faktor c_2 tendence jedince k posunu ke globálnímu nejlepšímu řešení. Poměr vlivu prioritních faktorů je moderován násobením pseudonáhodnými čísly $rand_k$ v intervalu [0, 1] s normálním rozložením pravděpodobnosti [10]. Index *i* reprezentuje jedince v rámci generace, index *j* atributy vektorů. Výpočet nové rychlosti v^{t+1} je vyjádřen vztahem [10]:

$$v_{ij}^{t+1} = w \cdot v_{ij}^{t} + c_1 \cdot rand_1 \cdot \left(pBest_{ij} - x_{ij}^{t}\right) + c_2 \cdot rand_2 \cdot \left(gBest_j - x_{ij}^{t}\right) \qquad (6)$$

Nová pozice jedince x_{ij}^{t+1} je dána vektorovým součtem s novou rychlostí [10]:

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \qquad (7)$$



Obrázek 8: Směr pohybu jedince ovlivněný původní rychlostí a hodnotami *pBest* a *gBest*

Iterace je ukončena po přesunu celé populace na nové pozice v prostoru možných řešení, hodnoty CFE všech jedinců jsou porovnány s *pBest* a *gBest* a v případech, kdy je nalezeno lepší řešení dojde k přepisu starého jedince jedincem novým [4] [10].

2.4 Diferenciální evoluce DE

Jedná se o algoritmus, jehož evoluční proces je tvořen diferenciální mutací a diskrétní rekombinací jedinců. Počáteční generace je generována pseudonáhodně s rovnoměrným rozložením pravděpodobnosti v prostoru možných řešení. V každé iteraci je z každého jedince, rodiče, vytvořen potomek působením evolučních procesů v pořadí mutace, křížení a selekce pomocí vektorových operací. Řídícími parametry DE jsou mutační konstanta *F* s obvyklou hodnotou 0,5 až 0,9 [12], a práh křížení *CR* s obvyklou hodnotou 0,7 až 0,9 [12]. Ukončovacím parametrem je maximální dovolený počet generací populace *Gmax* [4]. Evoluční procesy v DE jsou řízeny zvolenými strategiemi určujícími typ mutace *m*, počet diferencí *n* uplatněných během mutace a typem křížení *c*. Formální značení zvolené strategie je reprezentováno zápisem *DE/m/n/c* [1] [4].

2.4.1 Mutace

Jedinec určený k mutaci je označen jako aktivní a v závislosti na zvolené strategii je sestaven příslušný mutační, šumový, vektor *v*. Tento vektor je součtem určeného, jiného než aktivního jedince s váženými vektory rozdílů dvojic dalších, jiných než aktivních jedinců, přičemž váhu představuje mutační konstanta *F*. Pro základní variantu *DE/rand/1*

jsou náhodně vybráni tři jedinci [4], různí od aktivního, z nichž dva vstupují do operace váženého rozdílu, diference a jsou sečteny se třetím jedincem. Spolu s aktivním jedincem tato varianta pracuje se čtyřmi rodiči pro jednoho potomka, jedná se o minimální možný počet rodičů a velikost populace v DE. Varianta *DE/rand/2* k sestavení šumového vektoru analogicky využívá aktivního jedince a pěti dalších jedinců pro dvě diference a jednu operaci součtu. Minimální počet rodičů je šest. Výpočet šumového vektoru v_i pro variantu *DE/rand/1*[4][12]:

$$v_i = x_{rand\_1} + F \cdot (x_{rand\_2} - x_{rand\_3}) \tag{8}$$

Výpočet šumového vektoru v_i pro variantu *DE/rand/2*:

$$v_i = x_{rand\_1} + F \cdot (x_{rand\_2} - x_{rand\_3}) + F \cdot (x_{rand\_4} - x_{rand\_5}) \tag{9}$$

Jednou z modifikací je varianta *DE/randrl/1* používající nejlepšího z náhodně vybraných jedinců pro operaci součtu:

$$v_i = x_{rand\_1\_best\_of\_the\_three} + F \cdot (x_{rand\_2} - x_{rand\_3}) \tag{10}$$

V dalších modifikacích tvorby mutačního vektoru *DE/best/1*, *DE/best/2* jsou pro součet s diferencemi vybíráni jedinci s nejlepší hodnotou fitness [12]:

$$v_i = x_{best} + F \cdot (x_{rand\_1} - x_{rand\_2}) \tag{11}$$

$$v_i = x_{best} + F \cdot (x_{rand\_1} - x_{rand\_2}) + F \cdot (x_{rand\_3} - x_{rand\_4}) \tag{12}$$

Mutační vektor je ve variantách *DE/current_to_best/1*, *DE/current_to_rand/1* modifikován přidáním hodnoty *K* [13], násobenou diferencí nejlepšího nebo náhodného jedince od aktivního jedince, který figuruje současně v operaci součtu. Proměnná *K* je pseudonáhodně generované číslo v intervalu [0, 1] s normálním rozložením pravděpodobnosti [13]:

$$v_i = x_{current} + K \cdot (x_{best} - x_{current}) + F \cdot (x_{rand\_1} - x_{rand\_2}) \tag{13}$$

$$v_i = x_{current} + K \cdot (x_{rand\_1} - x_{current}) + F \cdot (x_{rand\_2} - x_{rand\_3}) \tag{14}$$

Obdobně je mutační vektor sestaven u variant *DE/rand_to_best/1*:

$$v_i = x_{current} + K \cdot (x_{best} - x_{rand\_1}) + F \cdot (x_{rand\_2} - x_{rand\_3}) \tag{15}$$

2.4.2 Křížení

Atributy aktivního jedince jsou nahrazovány atributy mutačního vektoru metodikou závislou na typu křížení *c*. U binomického křížení je postupně, pro každý atribut aktivního jedince, provedena matematická komparace hodnoty *CR* s hodnotou pseudonáhodně vygenerovanou v definovaném rozložení pravděpodobnosti v intervalu [0, 1], a pokud platí, že *CR* je větší nebo rovno tomuto náhodnému číslu, je atribut aktivního jedince nahrazen odpovídajícím atributem mutačního vektoru [4]. Zápis základní varianty s binomickým křížením je reprezentován zápisem *DE/rand/1/bin*:

$$\begin{aligned} &for\ j\ \in (0, 1, \ldots dimension) \\ &\quad if\ rand_j < CR: \\ &\quad individual\_attribute_j = mutant\_attribute_j\ ; \\ &else: \\ &individual\_attribute_j = individual\_attribute_j; \end{aligned} \tag{16}$$

Tabulka 2: Příklad uplatnění CR při křížení

Atribut	0	1	2	3	4
rand	0,513262	0,880787	0,181286	0,96478	0,402596
CR	0,7	0,7	0,7	0,7	0,7
rand < CR	Pravda	Nepravda	Pravda	Nepravda	Pravda
Mutovaný vektor	1,207382	3,381736	2,753085	3,174235	1,537404
Starý jedinec	4,005833	0,825025	4,075671	0,055454	3,708294
Nový jedinec	**1,207382**	**0,825025**	**2,753085**	**0,055454**	**1,537404**

V exponenciálním křížení je nejprve pseudonáhodně zvolen atribut jedince jako výchozí pro aplikaci křížení. Výchozí atribut je vždy nahrazen atributem mutačního vektoru. Je pseudonáhodně vygenerován parametr β v definovaném rozložení pravděpodobnosti v intervalu [0, 1]. Zároveň je inicializován parametr délky křížení *L* na hodnotě 1. Cyklus dalšího nahrazování atributy mutačního vektoru je zahájen, jestliže je *L* ostře menší než počet atributů *N* zmenšený o výchozí atribut, a zároveň dokud je hodnota β neostře menší než *CR*. Tato podmínka je v platnosti pro smyčku i v dalších opakováních, přičemž parametr β je pokaždé nově generován. Tímto opatřením exponenciálně klesá pravděpodobnost nahrazení atributů v dalších opakováních smyčky [1] [13].

2.4.3 Selekce

Po uplatnění procesů mutace a křížení pro celou populaci vznikají noví jedinci, kteří jsou na základě fitness porovnáváni s původními, aktivními jedinci. Do nové generace jsou akceptováni jedinci s vyšší hodnotou fitness [1][4].

3 BENCHMARKING A SOUDOBÉ ALGORITMY

Prioritním úkolem soudobých EA je nasazení k řešení reálných optimalizačních problémů. Taková nasazení představují vysoké nároky na výpočetní výkon prostředku, na kterém je EA nasazen [14], a tím také vysoké finanční a časové výdaje. Z tohoto důvodu je nezbytné EA po ukončení produkčního stadia tvorby testovat, upravovat a eliminovat základní chyby, až po finální ladění. Za tímto účelem existují platformy umožňující testování EA před jejich komerčním nasazením [14].

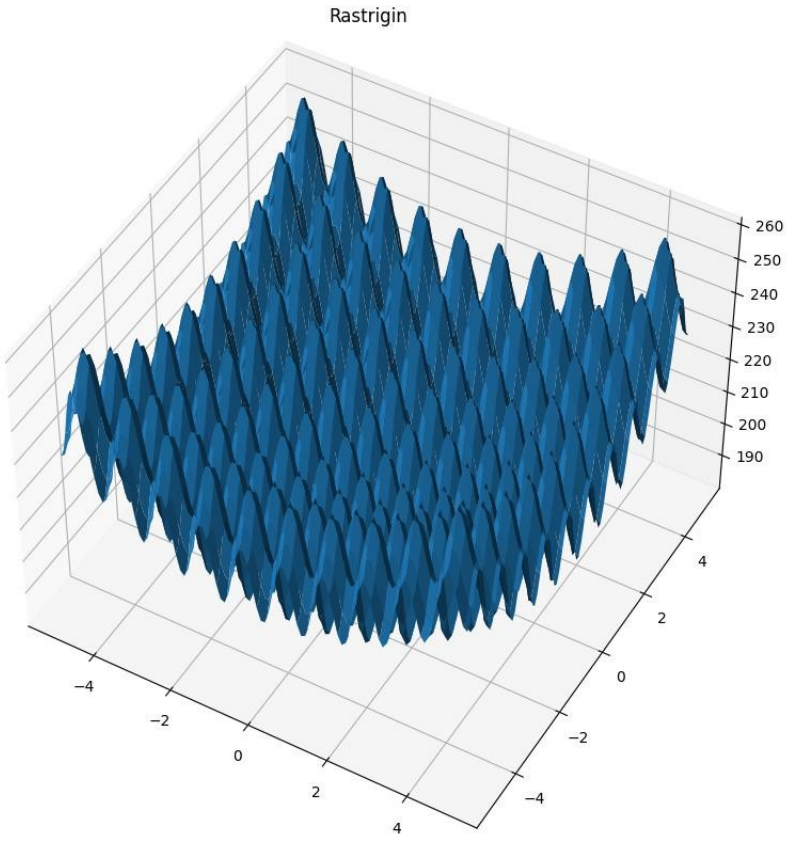
3.1 Benchmarking

Relevanci řešení poskytnutých EA lze ověřit prostřednictvím jejich výstupů na souboru definovaných, známých, abstraktních, testovacích problémů [15]. U většiny z nich jsou známy matematické definice účelových funkcí a globální extrémy [15]. Testování je možné ale také provádět na reálných optimalizačních problémech srovnáním výsledků zkoušeného algoritmu s výsledky jiných EA nasazených v minulosti. Testovací problémy, funkce, představují optimalizační problémy se škálou obtížností reprezentovaných patologiemi ve formě multimodality, plochy v okolí extrému nebo šumem [15]. Množinou takových funkcí je testovací sada (Benchmark set) [4]. Pro každou funkci je testovaný EA spuštěn 30 až 51 krát. Kritériem hodnocení může být nalezené možné řešení při dosažení povoleného maximálního počtu provedených CFE (Budget based), přesnost přiblížení se globálnímu optimu (Target based), rychlost dosažení optima, nebo jejich kombinace [4][15]. Použitím neparametrických testů jsou vyhodnocovány statistické údaje konvergencí pro všechny běhy algoritmu v určených časových okamžicích [16].

3.1.1 Testovací funkce

Matematické předpisy testovacích funkcí jsou reprezentovány rovnicemi, na jejichž levých stranách jsou funkční hodnoty vyjadřující CFE, a na pravých stranách vstupy funkčních argumentů x_i, z nichž každý vstupuje do matematické operace stanovené předpisem funkce [15]. Počet argumentů odpovídá počtu dimenzí testovaného problému a je ekvivalentem souřadnic polohy jedince X v D rozměrném prostoru možných řešení. Funkční hodnota je potom rovna sumě výsledků matematických operací nad všemi argumenty. V závislosti na předpisu testovací funkce mohou do těchto operací vstupovat také souřadnice daného jedince v sousední dimenzi x_{i+1} [15]. Příklad zápisu funkce Rastrigin:

$$f(x) = A \cdot D + \sum_{i=1}^{D} [x_i^2 - A \cdot cos(2\pi x_i)] \quad ; X = [x_1, x_2 \dots x_D] \; ; \; A = \; 10 \tag{17}$$



Obrázek 9: 3D reprezentace Rastriginovy funkce

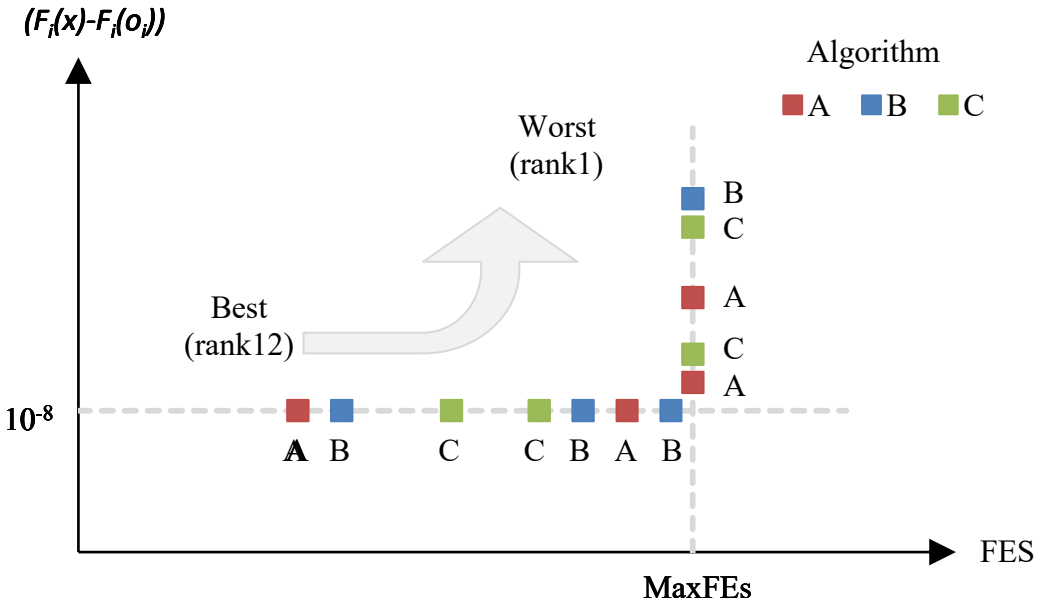
3.2 IEEE Congress on Evolutionary Computation Competition

Institut pro elektrotechnické a elektronické inženýrství (Institute of Electrical and Electronic Engineers, IEEE) je mezinárodní organizace sdružující odborníky z profesí souvisejících s elektronikou, zabývající se standardizací, vzděláváním a pokrokem v této oblasti. Publikování vědeckých prací prostřednictvím IEEE zvyšuje prestiž autora nebo kolektivu autorů v rámci vědecké komunity [17]. Institut každoročně pořádá Kongres o evolučních výpočetních technikách (Congress on Evolutionary Computation, CEC), v rámci kterých probíhají soutěže (Competition) týmů nasazujících vlastní varianty existujících EA na soubory testovacích funkcí (Testbed). Počet a matematické definice funkcí, velikost populace, dimenzionalitu problémů, velikost prostoru možných řešení a další parametry jsou veřejně dostupné před zahájením kongresu a jsou každoročně měněny [17].

3.2.1 IEEE CEC 2022

V roce 2022 se bude soutěž týmů v nasazení EA konat v rámci světového kongresu počítačové inteligence (World Congress on Computational Intelligence, WCCI) pořádaného pod záštitou IEEE [17]. Parametry dvanácti testovacích funkcí a kritéria pro hodnocení algoritmů při jejich řešení jsou zveřejněny v technické zprávě přístupné na webových stránkách IEEE [18]. Matematické definice jsou k dispozici v programovacích jazycích C, Python a Matlab. Globální extrémy všech funkcí představují jejich minimum. Velikost prohledávaného prostoru je [-100, 100] v dimenzích *10D* a *20D* a je požadováno 30 běhů algoritmu na každý problém [18]. Za účelem sjednocení výchozích podmínek je součástí každého problému náhodně vygenerovaná prvotní populace 1000 jedinců podle rovnoměrného rozložení v prostoru možných řešení, spolu s pravidly pro generování prvotních populací pro každý další běh algoritmu. Maximální možný počet CFE je stanoven hodnotou *MaxFEs* (Maximum Function Evaluations) na 200 000 pro *10D* a 1000 000 pro *20D*. Všechny funkce mají náhodně posunutý globální extrém o hodnotu o_i podle poskytnutých tabulek a funkce jsou rotovány užitím rotačních matic M, které jsou součástí dokumentace balíčku. Mimo jedné unimodální a čtyřech samostatných základních funkcí jsou součástí zadání také tři funkce hybridní, tvořené N funkčními předpisy $g_i(x)$ ve stanovených poměrech p_i a ve stanovených intervalech σ_i a omezenou dimenzionalitu. Tento přístup může mít za následek ztrátu spojitosti funkčních hodnot a více se přibližuje optimalizačním problémům reálného světa s mnoha komponentami. Součástí balíčku jsou také čtyři kompozitní funkce, kde kombinace N posunutých a rotovaných funkčních předpisů $g_i(x)$ je komponentou funkce $F(x)$, a platí $F(g_i(x))$. Vliv jednotlivých komponent je regulován vahou ω_i, kontrolou výšky λ_i, jsou distribuovány ve stanovených poměrech σ_i a globální minimum je stanoveno na základě proměnné $bias_i$. Základní funkce, hybridní i kompozitní varianty, jsou sestaveny ze šestnácti matematických předpisů f_1 až f_{16}. Kritéria pro hodnocení soutěžních EA nejsou zaměřena pouze na přesnost nalezeného řešení, ale také na rychlost. Jsou založena na Wilcoxonově testu a hodnotí velikost odchylky dosažené algoritmem po vyčerpání *MaxFEs* od globálního minima, vyjádřené vztahy pro chybu funkce $(F_i(x)-F_i(o_i))$, $F_i(o_i)=F_i^*$, nebo dosažení mezní hodnoty této odchylky se stanovenou velikostí 10^{-8} před vyčerpáním *MaxFEs*. Test využívá známkování (rank) jednotlivých konvergencí jednotlivých algoritmů na obou ukončovacích kritériích, jejichž součet (sum) tvoří celkové hodnocení

algoritmu *SR* (Sum-Rank) [18]. Tento test bude použit v praktické části a bude označován jako Rank-test.



Obrázek 10: Graf hodnocení Sum-Rank-testu CEC 2022

Tabulka 3: Vyhodnocení Rank-testu Sum-Rank-testu CEC 2022

Rank order	A	B	C	C	B	A	B	A	C	A	C	B	**SR**
Rank	12	11	10	9	8	7	6	5	4	3	2	1	
Algorithm A	12					7		5		3			**27**
Algorithm B		11			8		6					1	**26**
Algorithm C			10	9					4		2		**25**

Dosažené hodnoty CFE (FES) jsou dále v tabulkách zaznamenávány v šestnácti okamžicích každého běhu algoritmu podle stanoveného vztahu [18]:

$$FES_{point} = \left\lfloor D^{\frac{k}{5}-3} \cdot MaxFES \right\rfloor \quad ; k = [k_1, k_2 \dots k_{16}] \tag{18}$$

Ukončovací hodnota FE_{term} je zapsána do tabulky výsledků algoritmu jako sedmnáctá v pořadí, konvergence EA je tak zaznamenávána v osmnácti bodech od prvotní populace po terminální FES [18]. Výsledky jsou doplněny o záznam statistických hodnot konvergencí pro každý testovaný problém pro obě stanovené dimenze. Evaluačním faktorem je také složitost algoritmů a její dopad na výpočetní výkon pomocí výpočetních časů etalonu T_0, samostatné první testovací funkce T_1, a celého balíku funkcí T_2

na hardware soutěžního týmu [18]. Soubor výsledků algoritmů je poté předán organizátorům CEC k závěrečnému vyhodnocení a volbě vítězů [17].

3.3 Soudobé algoritmy

V současnosti jsou při řešení optimalizačních problémů využívány algoritmy, principielně založené na generických variantách EA [1]. Modifikovány jsou metody selekce, křížení i mutace, metody vzájemné komunikace a koordinace jedinců v rámci populace, řídící parametry algoritmů jsou dynamicky upravovány, populace jsou rozdělovány do sub-populací a je modifikována jejich velikost [4]. Algoritmy spolupracují s vestavěnými (Embedded) nástroji v podobě lokálních prohledávačů, nebo jsou voleny hybridní přístupy použitím několika konkurenčních typů EA [4].

3.3.1 jDE

Samo-adaptační DE (Self-Adaptive DE, jDE) je jednou z prvních upravených variant a využívá evoluční proces pro ladění řídících parametrů. Mutační konstanta F_i a práh křížení CR_i jsou dynamicky upravovány ve stanovených mezích na základě výsledku komparace pseudonáhodně generovaných čísel $rand_2$ a $rand_4$ z intervalu [0, 1] v normálním rozložení hustoty pravděpodobnosti, s konstantami $\tau_1 = \tau_2 = 0.1$ podle následujících vztahů [13]:

$$
\begin{gathered}
if\ rand_2 < \tau_1: \\
F_i^{t+1} = F_l + rand_1 \cdot F_u \\
else: \\
F_i^{t+1} = F_i^t \\
if\ rand_4 < \tau_2: \\
CR_i^{t+1} = rand_3 \\
else: \\
CR_i^{t+1} = CR_i^t
\end{gathered} \tag{19}
$$

Čísla $rand_3$ a $rand_4$ jsou generována stejným způsobem jako $rand_1$ a $rand_2$, proměnné $F_l=0.1$ a $F_u=0.9$ [13]. Hodnota mutační konstanty pro další iteraci se pohybuje v intervalu [0.1, 1], hodnota prahu křížení v intervalu [0, 1] [13].

3.3.2 SaDE

Tato samo-adaptační varianta pracuje s archivem výsledků hledání za účelem přizpůsobení mutační strategie algoritmu. Pro každou strategii k je uchovávána informace o její úspěšnosti, a mechanizmy algoritmu, užitím pravděpodobnostních nástrojů, zvolí jednu z uložených strategií pro každého jedince. Mutační konstanta F_i je pseudonáhodně generována z intervalu [0.5, 0.3] v normálním rozložení, adaptačním parametrem je tedy pouze práh křížení CR_i, který je generován stejným způsobem z intervalu [CR_{mk}, 0.1], kde CR_{mk} představuje medián hodnot CR_i uložených v paměti pro strategii k [3] [13].

3.3.3 ARPSO

V základní verzi PSO je využita paměť nejlepších doposud nalezených řešení k modifikaci pohybu jedinců, kteří jsou tak přitahování (attraction) jeden k druhému, což může vést ke ztrátě diverzity populace a předčasné konvergenci. Modifikovaná verze ARPSO (Attraction – Repulsion PSO, přitažlivost – repulze PSO) pracuje s přepínáním mezi režimy přitažlivosti a repulze na základě kontroly diverzity populace, a je jím řízen běh algoritmu (diverzity guided). Algoritmus obsahuje operátor *dir* [-1, 1] řídící režim ARPSO [19]:

$$v_{ij}^{t+1} = w \cdot v_{ij}^{t} + dir\left(c_1 \cdot rand_1 \cdot \left(pBest_{ij} - x_{ij}^{t}\right)\right) + dir\left(c_2 \cdot rand_2 \cdot \left(gBest_j - x_{ij}^{t}\right)\right) \quad (20)$$

Při běhu algoritmu je v každé iteraci vyhodnocován stav operátoru *dir* a hodnota diverzity populace, měřená jako vzdálenost k průměrnému bodu (Distance to average point, DAP). Jestliže dojde k poklesu diverzity v režimu přitažlivosti pod uživatelsky nastavený práh *divLO*, dojde k přepnutí operátoru *dir* do režimu repulze. Jestliže dojde k překročení hodnoty diverzity případě běhu algoritmu v režimu repulze nad uživatelsky nastavenou hodnoty *divHI*, dojde k přepnutí operátoru *dir* do režimu [19]:

$$if\ (dir > 0\ AND\ diversity < divLO):$$

$$dir = -1$$

$$if\ (dir < 0\ AND\ diversity > divHI):$$

$$dir = 1 \quad (21)$$

3.3.4 OLPSO

Tato varianta PSO využívá pokročilé metody ortogonálního učení (Orthogonal Learning PSO) založené na ortogonálním experimentálním designu (Orthogonal Experimental Design, OED). OLPSO využívá OED k sestavování ortogonálních polí faktorů a úrovní, a umožňuje nalezení optimálních kombinací atributů *pBest* a *gBest* pro každou dimenzi optimalizačního problému. Experimentálními faktory jsou atributy jedinců v dané dimenzi, úrovně představuje preference vlastního *pBest* a *gBest*. Ortogonální pole minimalizuje počet možných kombinací atributů z hodnoty 2^D na $2^{[\log2(D+1)]}$ čímž je zrychlen proces sestavování rychlostního vektoru při vyšší stabilitě pohybu jedinců při pohybu po hyperploše [20].

3.3.5 SOMA T3A

Strategie T3A (Team To Team Adaptive, adaptivní strategie tým k týmu) je sestavena z operací inicializace, organizace populace, migrace a aktualizace. Ve fázi organizace je z populace pseudonáhodně vybrána skupina *m* jedinců, z nichž je na základě hodnoty CFE vybráno *n* jedinců, tým, určených k migraci. Ke každému jedinci ze skupiny migrantů je následně z celé populace pseudonáhodně vybrána skupina *k* jedinců, z nichž je vybrán jedinec s nejlepší hodnotou CFE jako leader, přičemž výběr leadera probíhá *n*-krát. Skupiny *m* a *k* jsou vybírány vždy z celé populace, může tedy nastat situace, kdy se leader vyskytuje také ve skupině migrantů *n*. V takovém případě je tento jedinec ve skupině migrantů *n* ignorován. Adaptačním faktorem SOMA T3A je iterativní zvyšování velikosti perturbačního vektoru PRT s ohledem na poměr počtu provedených ohodnocení účelové funkce CFE_i a maximálního povoleného počtu ohodnocení *maxCFEs* [21].

$$PRT = 0.08 + 0.90 \cdot \frac{CFE_i}{maxCFEs} \tag{22}$$

Alternativa parametru *PathLength* pro tuto verzi SOMA je stanoven parametrem počtu provedených skoků jedince N_{jumps} o velikosti *Step*, který je dalším adaptačním faktorem algoritmu. V každém opakování algoritmu je snižována jeho velikost [22].

$$Step = 0.15 - 0.08 \cdot \frac{CFE_i}{maxCFEs} \tag{23}$$

Díky adaptačním faktorům má pohyb jedinců po hyperploše zpočátku více explorativní charakter, který se postupně mění v charakter exploitativní [1] [21][22].

3.3.6 ESP-SOMA

Jedná se o adaptivní variantu SOMA, která prostřednictvím souboru strategií a perturbačního parametru (Ensemble Strategies and Pertrubation parameter, ESP) modifikuje řídící parametry a použitou strategii algoritmu. Každému jedinci je přiřazena vlastní migrační strategie z trojice AllToOne, AllToAll a AllToOneRand, a velikost *PRT* z množiny hodnot [0.1, 0.3, 0.5, 0.7, 0.9]. Pro tuto variantu SOMA je zaveden řídící parametr *gap*, mezera, v rozmezí celočíselných hodnot 7 až 19, určující maximální povolený počet neúspěšných iterací jedince, po jehož překročení je jedinci přiřazena jiná migrační strategie a jiná hodnota *PRT* [23].

3.4 State of the art algoritmy

3.4.1 L-SHADE

Adaptivní varianta diferenciální evoluce založená na historii úspěchů, doplněná o funkci deterministicky řízené redukce populace pomocí lineární funkce (Success-History based Adaptive DE with Linear Population Size Reduction, L-SHADE), a vítěz CEC 2014. Skupina řídících parametrů generické verze DE je rozšířena o paměť hodnot mutačního faktoru $M_{F,k}$ a prahu křížení $M_{CR,k}$, schopnou zaznamenávat $k=(1, 2, ...,H)$ hodnot. Mutovaný vektor je tvořen s využitím mutační strategie current_to_pBest/1/bin. Jedinec *pBest* je pravdivostními operacemi vybrán ze skupiny N nejlepších jedinců v rámci generace prostřednictvím parametru p [0, 1], mutovaný vektor má pak podobu [24]:

$$v_i = x_{current} + F_{current} \cdot (x_{pBest} - x_{current}) + F \cdot (x_{rand\_1} - x_{rand\_2}) \qquad (24)$$

Velikost parametru p má vliv na charakter běhu algoritmu, přičemž nízká hodnota má za následek exploitativní chování a naopak. Po procesu křížení a selekce jsou v paměti uloženy hodnoty řídících parametrů CR_i a F_i pro každého jedince v případech, kdy bylo nalezeno lepší řešení do polí S_F a S_{CR}. Po ukončení evolučního procesu pro jednu generaci, jsou s využitím váženého lehmerova průměru $mean_{WL}$ do paměti M_F a M_{CR} zaznamenány průměrné hodnoty těchto řídících parametrů k využití pro tvorbu další generace pravdivostními operacemi. Váha představuje velikost zlepšení CFE mezi rodičem x_i a zkušebním vektorem u_i. Počet jedinců nové generace N_{G+1} je redukován s ohledem na poměr počtu provedených ohodnocení účelové funkce CFE_i a maximálního povoleného počtu ohodnocení *maxCFE*, a uživatelsky stanovené hodnotě minimálního počtu jedinců v populaci N_{min} a velikosti počáteční populace N_{init} [24].

$$N_{G+1} = round\left(\left(\frac{N_{min} - N_{init}}{maxCFE}\right) \cdot CFE_i + N_{init}\right) \quad (25)$$

Do nové generace nejsou přeneseni jedinci s nejhoršími nalezenými řešeními. Vzhledem k použité mutační strategii je nezbytné stanovit N_{min} ne nižší než 4 jedinci [24].

3.4.2 Db-SHADE

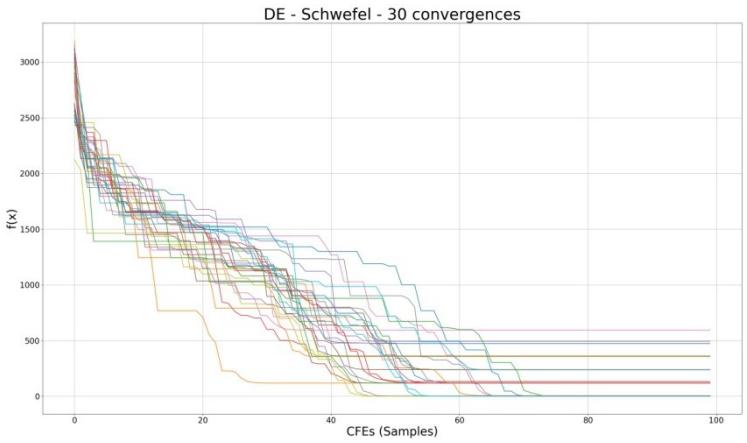
Algoritmus odvozený od varianty SHADE s adaptací mutačního faktoru *F* a prahu křížení *CR* založené na euklidovské vzdálenosti atributů zkušebního vektoru u_i a rodiče x_i (Distance based, Db). Vážený Lehmerův průměr $mean_{WL}$ je vypočítáván s využitím váhy odpovídající velikosti pohybu jedince, čímž je podporováno explorativní chování a je zachovávána diverzita populace [25].

4 VNITŘNÍ DYNAMIKA ALGORITMU

Proces hledání nejlepšího řešení optimalizačního problému je ovlivněn souhrnem faktorů ovlivňujících pohyb jedinců po hyperploše a tím tvar výsledné konvergenční křivky. Sociální chování populace a následování lepších řešení, vzájemná komunikace a předávání informace o vlastních, ale i populačních nejlepších doposud nalezených řešeních, vytváření shluků nad možnými extrémy, a rychlost pohybu v prostoru možných řešení, jsou ovlivňovány řídícími parametry algoritmu a v ideálním případě pracují ve vzájemné rovnováze [4][5].

4.1 Trajektorie konvergence a předčasná konvergence

Vzhledem k zapojení stochastických jevů vykazují různé běhy algoritmu na konkrétním problému měřitelné rozdíly, které jsou statisticky vyhodnocovány k získání komplexní představy o výkonu algoritmu. Periodicita a velikost změn konvergenční křivky od prvotní populace po nalezení globálního extrému, jsou ukazatelé výkonu na daném problému, přičemž může docházet k předčasné konvergenci v podobě stagnace v rovině, nebo uváznutí v lokálním extrému multimodální účelové funkce. Tvar konvergenční křivky je určován poměrem mezi explorací a exploitací v daném okamžiku. Plochý průběh představuje dominanci explorace, vysoká strmost křivky je projevem exploitace algoritmu. Typické průběhy konvergenčních křivek jsou složeny z částí vykazující střídavě převahu obou tendencí. Konstantní průběh hodnot CFE od nestandardního okamžiku běhu algoritmu až do ukončovací podmínky ukazuje na předčasnou konvergenci [26].



Obrázek 11: Příklad předčasné konvergence běhů algoritmu DE na funkci Schwefel

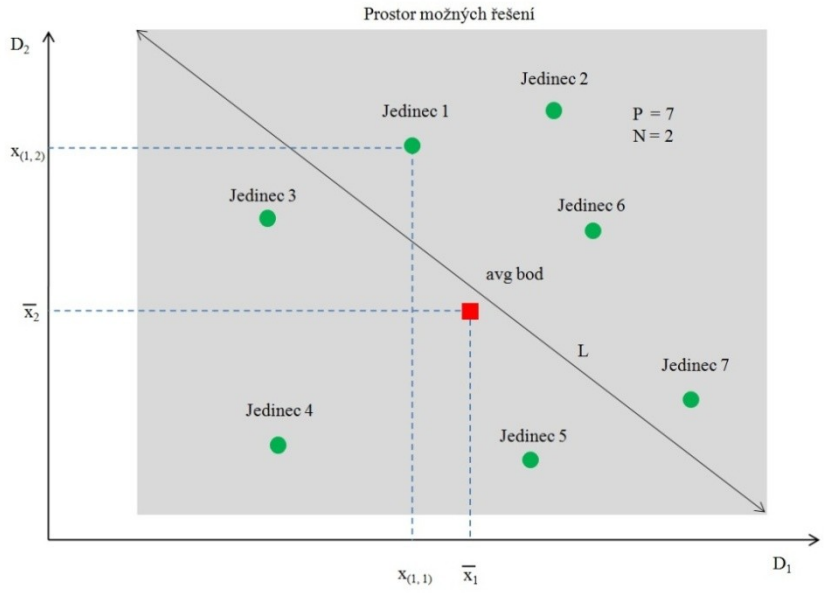
4.2 Diverzita populace

Výkon EA je závislý na míře rozdílnosti poloh jedinců v populaci, diverzitě, která je důležitým předpokladem pro ochranu před předčasnou konvergencí. Vysoký tlak selekce elitních jedinců má za následek snahu procházet již prohledaný prostor možných řešení, zároveň může velký vliv rekombinačních mechanizmů způsobit postupnou podobnost jedinců. Oba vlivy mohou vést ke konečné homogenitě populace a uváznutí v lokálním extrému. Diverzitu, různorodost, lze vyjádřit velikostí vzájemných rozdílů u odpovídajících atributů jedinců, a její kontrola a udržování po celou dobu běhu algoritmu podporuje exploraci. Pokud je diverzita populace zachovávána i v okamžiku nalezení slibného řešení, s vyšší pravděpodobností dojde k eliminaci předčasné konvergence a k vyhledání více lokálních optim u multimodálních problémů [1] [4] [27].

4.2.1 Stanovení míry diverzity populace

Metody výpočtu diverzity využívají vztahy pro výpočet euklidovské vzdálenosti jedinců v prostoru možných řešení, a statistických veličin derivovaných z atributů jedinců [28]. Variantou stanovení diverzity je výpočet vzdálenost jedinců od průměrného bodu *avg* (Distance to Average Point, DAP) [26], ve které figuruje velikost diagonály L prostoru možných řešení, velikost populace P, souřadnice jedinců $x_{i,j}$, souřadnice bodu *avg* $\overline{x_j}$ a dimenzionalita problému N. Index i identifikuje jedince v populaci, index j představuje pořadí atributu jedinců a *avg*, dimenzi. Výpočet DAP je vyjádřen vztahem [28][29]:

$$diversity(P) = \frac{1}{|L| \cdot |P|} \cdot \sum_{i=1}^{|P|} \sqrt{\sum_{j=1}^{N} \left(x_{ij} - \overline{x_j}\right)^2} \tag{26}$$



Obrázek 12: Schéma populace a bodu průměrného bodu *avg*

Obdobou DAP je zprůměrněná diverzita populace (Averaged Population Diversity, APD), jejíž formulace nezohledňuje velikost prohledávaného prostoru, využívá výpočtu euklidovské vzdálenosti od střední hodnoty atributů celé populace a průměruje ji počtem jedinců *N*, je vyjádřena vztahem [27]:

$$DI = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} \sum_{j=1}^{D} \left(x_{ij} - \overline{x_j}\right)^2} \tag{27}$$

Dalším přístupem je stanovení dimenzionální diverzity (Dimension-Wise Diversity, DWD), který je počítán jako průměr sumy rozdílů atributů každého jedince od společného mediánu [28]:

$$Div = \frac{1}{D} \cdot \sum_{j=1}^{D} \sum_{i=1}^{N} median(x_j) - x_{ij} \tag{28}$$

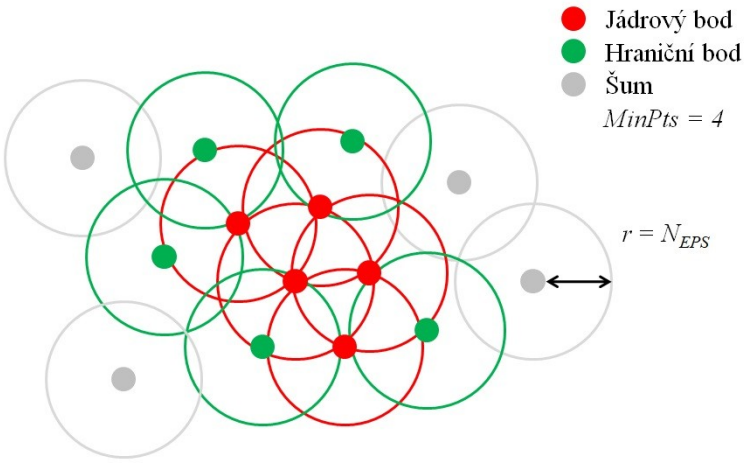
4.2.2 Mechanizmy ke kontrole diverzity populace

Algoritmy řízené kontrolou diverzity (Diversity Guided EA, DGEA) [19] využívají nástrojů pro úpravu řídících parametrů pro každý evoluční proces. Velikost změny diverzity populace je deterministickými metodami ověřována při každé změně atributů jedince a připuštěny jsou pouze změny vedoucí k jejímu zachování. Mechanizmy řízení

udržují diverzitu ve stanoveném rozmezí a reagují jejím zvyšováním směrem k d_{high} nebo snižováním k d_{low}, podle vývoje populace bez ohledu na kvalitu nalezených řešení. U dynamických optimalizačních problémů může být výhodné, když $d_{high} = d_{low}$ a diverzita balancuje na prahové hodnotě. Mechanizmy zásahu do evolučního procesu mohou být řízeny sledováním fitness hodnot každého jedince a jsou aktivovány, například když jsou nalezena dvě řešení, která jsou v rámci stanovené tolerance stejná. Tento mechanismus je využit u některých GA (No fitness duplicates) [30].

4.3 Clustery v populaci

Při analýze pohybu populace po hyperploše v průběhu konvergence EA mohou jedinci v rámci generace vytvářet shluky, clustery. Dochází k vzájemnému přiblížení jedinců v rámci všech atributů, dimenzí, přičemž toto přiblížení vykazuje celá populace nebo skupiny jedinců. Vznik clusterů je ovlivňován typem EA, nastavením řídících parametrů a vlastnostmi optimalizačního problému, a jsou protiváhou diverzity populace [28], pokud je měřena s ohledem na velikost prohledávaného prostoru [29]. Clustering je projevem expoitace a v okolí globálního optima vede k jeho rychlejšímu nalezení. Zároveň může ale v okolí lokálních extrémů být příčinou předčasné konvergence [25]. Základní varianta DE ukazuje nižší sklony k vytváření clusterů díky absencí navigace pohybu ostatních jedinců ve směru lepších řešení [13]. Následování vhodnějších řešení zbytkem populace je typické pro algoritmy PSO, jejichž populace jsou v ideálním případě clusterem pohybujícím se po prostoru možných řešení [5]. Algoritmus SOMA, v míře závislé na zvolené strategii, také disponuje mechanismem následování lepších řešení, avšak díky odlišně koncipovanému prohledávání hyperplochy má sklony tvořit clustery pozvolněji v průběhu konvergence [1][9]. Shluky v populaci lze klasifikovat na základě vzájemného postavení jedinců a jejich vzájemné vzdálenosti. Pro účely této práce je v praktické části využíván nástroj DBSCAN [34][35], který deterministicky hodnotí clustery na základě hustoty jedinců (density based clustering). Uživatelskými parametry DBSCAN jsou velikost sousedství jedince $N_{Eps}(p)$ a minimální počet bodů v sousedství *MinPts* nutných k tomu, aby byl hodnocený jedinec označen jako jádrový bod (core). Jedinec v sousedství jádrového bodu, který ve vlastním sousedství nesplňuje podmínku *MinPts* je označen jako hraniční (border). Jedinci nesplňující podmínku jádrového ani hraničního bodu jsou označeni jako šum (noise) [34]. Metrika, způsob výpočtu vzdálenosti a dosažitelnosti jedinců v sousedství *metric* je volitelným nastavením DBSCAN. [35].



Obrázek 13: Princip hodnocení clusteringu nástrojem DBSCAN

4.4 Dynamika pohybu

Projevem explorace EA je schopnost jedinců pohybovat se po celém rozsahu prostoru možných řešení. Dynamika pohybu jedinců je vyjádřením míry pohybu jedinců a její stagnace nebo zastavení představuje uváznutí v lokálním extrému nebo nalezení optima. Předčasná konvergence může být indikována stejnými hodnotami nalezených řešení u nejhorších i nejlepších jedinců v populaci, rozdílné hodnoty ukazují na nepřerušené prohledávání hyperplochy i při nalezení potenciálního extrému. Pohyb jedinců lze analyzovat na základě velikosti dílčích pohybů v průběhu konvergence, skoků, a celkové sumy všech skoků za celou konvergenci. Je-li porovnávána míra pohybu jedinců vzhledem k délce trvání běhu algoritmu, která je pro celou populaci shodná, lze označit velikost pohybu za rychlost jedince, která je jedním z výstupů analýzy v praktické části této práce [36].

4.5 Robustnost algoritmu

Teoretickým východiskem EA je konzistentní výkon na skupině podobně definovaných problémů za podobně definovaných podmínek [37]. Robustnost algoritmu lze tedy chápat jako schopnost konkrétního algoritmu produkovat uspokojivé výstupy na množině problémů s blízkými charakteristikami [38]. NFLT ukazuje nutnost přizpůsobovat EA konkrétním problémům, zejména s ohledem na dimenzionalitu problému. V této práci robustnost generických, optimalizačním problémům nijak specificky přizpůsobených algoritmů, testována na trojici EA s osmi základními testovacími funkcemi, a pomocnou

veličinou při tvorbě výstupů praktické části. Robustnost algoritmu je manifestována stabilitou výkonu na skupině testovacích funkcí, a představuje rovnováhu mezi diverzitou populace a clusteringem, exploitací a explorací [37][38].

PRAKTICKÁ ČÁST

5 APLIKACE PRO IDENTIFIKACI SLABÝCH MÍST EVOLUČNÍCH ALGORITMŮ

Konzolová aplikace s názvem evalGeval (Evolutionary Algorithm Evaluation, Evaluace Evolučních Algoritmů) je vytvořena v programovacím jazyce Python a jejím účelem je provádění statické analýzy výstupů evolučních algoritmů k identifikaci jejich slabých míst. Výstupní data algoritmů představují vstupy aplikace. Analýza je zaměřena na výstupy minimalizačních jednokriteriálních úloh. Programovací jazyk byl zvolen z důvodu možnosti multiplatformního využití. Konzolové rozhraní provádí uživatele procesem zadávání vstupních dat, informuje uživatele o probíhajících operacích, jejich úspěšném vykonání, a zároveň upozorňuje na nastalé chyby a problémy. Po končení analýzy nabízí volbu opakování zadávání vstupních dat. Aplikace je složena ze čtyř modulů. `main.py` definuje uživatelské rozhraní a odezvy aplikace v konzolovém prostředí, `csv_test.py` provádí kontrolu vstupních souborů s daty, `csv_reader.py` slouží ke čtení obsahu vstupních souborů a vytváření polí dat ve formátu vhodném k provedení analýzy, `analyzer.py` vyhodnocuje z uložených polí konvergence algoritmů, diverzitu populace, clustering a dynamiku pohybu jedinců po hyperploše, vytváří a ukládá výstupy aplikace. K aplikaci je připojen soubor `README.md` s instrukcemi pro tvorbu vstupních souborů ve formátu *csv* (Comma Separated Values, hodnoty oddělené čárkami) [39]. Pro účely aplikace jsou jako primární oddělovače využity středníky, aby byla usnadněna práce v tabulkových procesorech při převodu desetinných teček na čárky a nedošlo tak k rozpadu integrity uložených dat. Řádky souborů definují obsah specifický pro každý soubor a jsou ukončeny standardními `CRLF` (Carriage Return, Line Feed) znaky [39]. Struktura dat uložená ve vstupních souborech má zásadní význam pro správný průběh analýzy a je popsána v souboru `README.md`. Text v aplikaci je v anglickém jazyce z důvodu mezinárodní univerzálnosti. Aplikace využívá ke své činnosti balíků Python `os`, `numpy`, modul `pyplot` z knihovny `matplotlib`, třídu `Lightcsv`, a třídu `DBSCAN` z balíku `sklearn.cluster`.

5.1 Vstupy aplikace

Vzhledem k potenciální vysoké výpočetní náročnosti analýzy kompletních výstupů evolučních algoritmů, zejména kompletních záznamů populací, a s ohledem na velikost výstupních dat, představujících přílohu diplomové práce, byl upřednostněn přístup provedení analýzy pouze na záznamech populací a konvergencí z každého běhu algoritmu,

vznikajících uložením dat každou n-tou iteraci. Tyto záznamy budou dále označovány jako vzorky, v grafech jsou řady vzorků na osách *x* označeny jako CFEs (Samples) (Cost Function Evaluatuions, počet ohodnocení účelové funkce, vzorky). K analýze je pro každý běh algoritmu 100 vzorků průběhů konvergencí v jednom *csv* souboru, a 100 vzorků populací ve druhém csv souboru, vždy od první generace až po ukončovací podmínku. Okamžiky vzniku vzorků populací odpovídají okamžikům vzniku vzorků průběhu konvergencí. Počet vzorků pro jeden běh je zohledněn také ve výstupních grafech, jejichž osa *x* bude mít v této práci hodnoty odpovídající indexům vzorků 0 až 99. Aplikace nemá vytvořen žádný limit velikosti populace, počtu dimenzí, ani počtu záznamů, které tak mohou obsahovat úplná výstupní data algoritmu. Vstupními soubory jsou dva záznamy dat ve formátu *csv* pojmenované pro účely této práce podle použitého algoritmu, testovací funkce a typu obsažených dat `algorithm_fnumber_typeofdata.csv`. Prvním z nich je soubor obsahující záznamy vzorků populací, pojmenovaný například `DE_f6_p.csv`, pro všechny opakování běhu algoritmu. Opakování jsou definována samostatnými řádky obsahujícími populace s jedinci, oddělenými jednoduchou mezerou, jejichž atributy jsou odděleny středníky bez mezer. Formální zápis populací jedinců o třech atributech, dimenzích, pro dvě opakování běhu, bude mít v souboru následující formát:

$$\begin{aligned} &1\ \ x_{0,0};x_{0,1};x_{0,2}\ x_{1,0};x_{1,1};x_{1,2}\ ;\ \dots\ ;\ x_{n,0};x_{n,1};x_{n,2} \\ &2\ \ x_{0,0};x_{0,1};x_{0,2}\ x_{1,0};x_{1,1};x_{1,2}\ ;\ \dots\ ;\ x_{n,0};x_{n,1};x_{n,2} \end{aligned} \tag{29}$$

Reálný zápis populací o deseti jedincích se třemi atributy pro dvě opakování běhu algoritmu má ve výstupu konzoly následující podobu:

```
1 75.25108263500499;-375.17980658672957;218.39402756417383 45.83289741342696;-28.660480090774172;-92.94557541879436
  -374.51893855457683;-204.78879693579597;-153.6845026613022 227.7304773632277;-61.827611317189O5;31.895016291530283
  357.41598286048156;-163.3401541090397;112.3899389284702 -115.97886653082969;-81.12971700286784;84.63571077982658
  377.6945902950474;-471.2521027522144;172.8762770882929 -260.87381075098904;-391.4233033239941;-253.85775546588084
  239.54163644363302;-78.99819877198809;335.0485461129216 80.15217512113736;-465.71510006703176;-238.57285370126948
2 -157.91853719855033;363.3016733860909;-132.3431479330788 25.988097724499312;-141.75887044907773;142.68716045203223
  -105.88006054054034;279.7593637005705;-123.55021212631056 -366.73067549933273;384.1461928207026;61.09060893682192
  -183.4569439478442;-454.35273473177364;309.1847263393163 -97.91757910380551;-65.26444673535269;471.8943992966949
  -459.4509456391733;132.12309931281368;168.00095236678658 390.5304434664737;-280.59812503368994;465.46898884923326
  395.91463409767516;94.80089074706143;396.71827798236654 -28.221151873395343;-368.5060104617995;291.11966200450024
```

Obrázek 14: Ukázka výpisu vzorků populací v konzole

Druhým vstupem je soubor, který obsahuje průběh konvergence daného algoritmu, pojmenovaný například `DE_f6_c.csv`. Opakování běhu jsou definována samostatnými řádky. Předepsaným formátem je zápis hodnot jednoho běhu do jednoho řádku, s hodnotami oddělenými středníky bez mezer. Formální zápis CFE pro tři opakování běhu bude mít v souboru následující formát:

```
1 CFE0;CFE1;CFE2;...;CFEn
2 CFE0;CFE1;CFE2;...;CFEn
3 CFE0;CFE1;CFE2;...;CFEn
```
(30)

Reálný zápis CFE dvou opakování běhu algoritmu má ve výstupu konzoly následující podobu:

Obrázek 15: Ukázka výpisu vzorků konvergencí v konzole

Podmínkou správné funkce aplikace jsou stejné parametry výstupu EA ve dvojicích vstupních souborů. Těmito parametry jsou počet opakování algoritmu a počet zaznamenaných vzorků konvergencí a populací.

5.2 Tvorba vstupů aplikace

Pro účely této práce a k získání dat pro vstupní soubory aplikace byly vytvořeny samostatné moduly, využívající ke svému běhu vazbu na modul s kódem evolučního algoritmu a parametry běhu algoritmu, využívající knihovnu Python `numpy`. Modul k vytváření souborů s daty k analýze není standardní součástí aplikace. Vkládání dat do souborů je provedeno pomocí smyček pracujících s datovými poli, které jsou produkty evolučního algoritmu. K využití tohoto modulu bylo třeba v těle algoritmu definovat pole s průběhem konvergence, pole se záznamy vzorků populací, a tato pole podle určených pravidel naplňovat daty.

```
for r in repetition
    create first generation
    evaluate
    store population into array
    store sample position into array
    store convergence into array

    for b in (budget - first evaluation)
        evolve
        evaluate
        store convergence into array

        store every n-th population into array
        store sample position into array

        if generation = last
        store population into array
        store sample position into array
```

Obrázek 16: Pseudokód umístění záznamu dat do polí pro vstupní soubory uvnitř algoritmu

Vzorky populace analyzované v této práci jsou uloženy v denormalizované podobě každou n-tou generaci, čehož je docíleno komparací operace modulo počítadla generace a konstanty n-tého opakování s nulou:

```
if generation % runman.nth == 0:
    samples.append(X_denormalized.tolist())
    samplegeneration.append(generation)
```
(31)

K analýze bylo získáno 100 vzorků populací a 100 vzorků konvergencí v každém běhu každého algoritmu. Výsledná pole jsou ve smyčkách zpracovávána v souladu s parametry algoritmu od nejvyšší instance opakování běhu, až po jednorozměrná data. V případě vzorků populací je zpracováváno pěti-rozměrné pole polí od kompletního počtu opakování až po atribut jedince.

```
All population samples in all repetitions:
samples [[[[150.51714948862127, -345.5779979522282, -140.26437365981417...

All population samples in one repetition:
samples[0] [[[150.51714948862127, -345.5779979522282, -140.26437365981417...

One population sample:
samples[0][0] [[150.51714948862127, -345.5779979522282, -140.26437365981417...

Individulal:
samples[0][0][0] [150.51714948862127, -345.5779979522282, -140.26437365981417...

Individual attribute:
samples[0][0][0][0] 150.51714948862127
```

Obrázek 17: Výstupy obsahu multidimenzionálního pole se vzorky v konzole

Ve vnitřní smyčce zápisového modulu byla vytvořena podmínka pro vkládání oddělovacího znaku středníku mezi atributy, mezery mezi jedince populace a symbolu pro nový řádek *\n* po skončení čtení dat z jednoho opakování algoritmu. Data konvergencí jsou zpracovávána obdobnou smyčkou a uložena v jednorozměrném poli v jednom řádku souboru, počet řádků odpovídá počtu opakování běhu algoritmu. Záznamy posledního jedince ve vzorku populace a poslední CFE v opakování běhu nejsou opatřeny oddělujícími znaky. Poslední řádky, záznamy, obou souborů nejsou opatřeny znakem pro další řádek. Postup pro tvorbu vícerozměrných polí byl zvolen s ohledem na operace následné analýzy nástroje pro identifikaci slabých míst evolučních algoritmů se zaměřením na dosažitelnost libovolné instance dat.

5.3 Běh aplikace

Po spuštění aplikace je uživatel dialogy vyzván k zadání systémových cest k souborům se vzorky konvergencí a populací, a umožňuje provést analýzu ve dvou režimech, v závislosti na uživatelských vstupech. V případě zadání cest ke dvěma samostatných *csv* souborům je aktivován souborový režim, který umožní provedení jedné analýzy na dvojici souborů se vzorky populací a konvergencí. Umístění souborů při činnosti v souborovém režimu není nijak omezeno.

```
////////////////////////////////////////////////
***  evalGeval 2022  ******************************
////////////////////////////////////////////////

For information on file formats consult instruction file README.md
Follow required file formats precisely
```

Obrázek 18: Logo aplikace s prvotními instrukcemi

```
Select path to populations record file or
path to folder with multiple samples files:
D:/UTB/DP/pyhelp/data_container/populations/de_fó_p.csv
Select path to convergences record file or
path to folder with multiple convergences files:
D:/UTB/DP/pyhelp/data_container/convergences/de_fó_c.csv
```

Obrázek 19: Dialog pro vložení cest k samostatným csv souborům

V případě potřeby provedení analýzy na více souborech současně v rámci adresářového režimu je nezbytné data rozdělit do dvou adresářů. V jednom adresáři soubory se vzorky

populací, ve druhém soubory se vzorky konvergencí. Pojmenování souborů a adresářů při přípravě dat je uživatelskou volbou a není aplikací nijak omezeno. Po zadání cest ke dvěma adresářům obsahujícím data je aktivován adresářový režim a analýza bude provedena pro každou dvojici vstupních souborů se vzorky populací a konvergencí.

```
Select path to populations record file or
path to folder with multiple samples files:
D:/UTB/DP/pyhelp/data_container/populations
Select path to convergences record file or
path to folder with multiple convergences files:
D:/UTB/DP/pyhelp/data_container/convergences
```

Obrázek 20: Dialog po vložení cest k adresářům s csv soubory

Aplikace upozorní uživatele v případě zadání systémové cesty, která neexistuje, nebo se některá z cest shoduje s již zadaným údajem. Za účelem kontroly vstupních souborů je vytvořena funkce `csv_test`, která v adresářovém režimu porovnává počty souborů v zadaných adresářích, v obou režimech provádí kontrolu počtu řádků v zadaných nebo nalezených souborech a jejich přípony. Při testování vrací funkce hodnotu počtu opakování algoritmu, velikost populace, počet dimenzí a počet vzorků zaznamenaných v jednom běhu algoritmu. Proces proběhne u každé dvojice souborů, a v adresářovém režimu jsou na jeho konci porovnány parametry všech dvojic.

```
RUNNING FOLDER-MODE

checking file de_f1_p.csv
checking file de_f1_c.csv
checking file de_f2_p.csv
checking file de_f2_c.csv
checking file de_f3_p.csv
checking file de_f3_c.csv
```

Obrázek 21: Probíhající kontrola *csv* souborů v adresářovém režimu

Všechny dvojice souborů musí mít stejné parametry. V případě konfliktu je uživatel informován příslušným dialogem a vyzván k opakování kroku. Jediným údajem, který nelze z uložených dat extrahovat, a je nezbytný k provedení analýzy, je velikost prohledávaného prostoru, a je nutné ji uživatelsky zadat. Následuje dialog s výzvou k zadání způsobu uložení výstupních souborů. Uživatel má na výběr stiskem klávesy *Enter*

automaticky vytvořit podadresář s názvem *evalGeval_results* v umístění vstupních souborů nebo adresářů, nebo je možno zadat plnou cestu k jinému existujícímu adresáři. Dialog je v kódu aplikace ošetřen proti zadání neplatné cesty nebo přepisu existující složky.

```
Select whole searchspace dimension (i.e. for [-5, 5] type 10):
10
Select destination folder for CSV and JPEG results files
or hit ENTER to create folder evalGeval_results inside
D:\UTB\DP\pyhelp\data_container\populations
```

Obrázek 22: Dialog pro zadání velikosti prohledávaného prostoru a umístění výstupních souborů aplikace

Po tomto posledním kroku volby vstupních parametrů je uživateli předložena sumarizace zadaných dat a dialog s výzvou k volbě pro opakování zadávacího procesu, spuštění analýzy nebo ukončení programu.

```
SINGLE-FILE MODE SUMMARY:
Population convergence record file:          D:/UTB/DP/pyhelp/data_container/convergences/de_fó_c.csv
Population samples record file:              D:/UTB/DP/pyhelp/data_container/populations/de_fó_p.csv
Results files destination:                   D:\UTB\DP\pyhelp\data_container\populations\evalGeval_results
Number of algorithm repetitions:             30
Population size:                             50
Number of dimensions:                        10
Searchspace dimension:                       10.0
Computed number of samples per repetition:   100

Proceed to analysis?
Type y to proceed, n to repeat the parameter input process or q to quit evalGeval
```

Obrázek 23: Sumarizace zadaných údajů a výzva k uživatelské volbě

Od tohoto okamžiku je běh aplikace automatizován a je nejprve vytvořen adresář pro ukládání výsledných souborů a poté je spuštěno čtení dat ze vstupních souborů *csv* prostřednictvím modulu `csv_reader.py`.

```
Proceed to analysis?
Type y to proceed, n to repeat the parameter input process or q to quit evalGeval
y
Running reader on de_f1_p.csv
done...
Running reader on de_f1_c.csv
done...
Analyzing convergences...
done...
Analyzing population diversity...
done...
Analyzing population clustering...
done...
Analyzing populations dynamics...
done...
```

Obrázek 24: Informace o zpracovávaných souborech a prováděných operacích analýzy

5.4 Použité evoluční algoritmy

Pro účely této práce byly použity dva evoluční algoritmy v generických podobách, představující základní přístupy k optimalizaci. DE, PSO a pokročilá SOMA T3A, přičemž lze předpokládat vyrovnanější průběhy konvergencí než u dvojice generických algoritmů, proto výsledky SOMA T3A představují etalon. Algoritmy byly vytvořeny v programovacím jazyce Python a nejsou součástí standardní aplikace. V kódu každého z nich byly implementovány mechanismy pro zápis polí potřebných pro vznik vstupních souborů aplikace k provedení analýzy. DE je zvolena ve variantě *DE/rand/1* s prahem křížení $Cr = 0.9$ [1] a mutační konstantou $F = 0.5$ [1]. Pro PSO v základní variantě jsou stanoveny prioritní faktory $c_1 = c_2 = 2$ [1], setrvačnost $w = 0.8$ [1]. Algoritmus SOMA se strategií *TeamToTeamAdaptive* s parametry $N_{jumps} = 45$, $m = 10$, $n = 5$, $k = 10$ [22]. Algoritmy byly na testovacích funkcích spouštěny pouze jako zdroje dat pro analýzu, jejímž účelem je odhalit slabá místa těchto algoritmů. Hodnocení výkonu jednotlivých algoritmů není cílem této práce, dále v práci budou ale použity hodnotící metody, které si kladou za cíl identifikovat nedostatky a selhání algoritmů na testovacích funkcích a využít je jako markanty slabých míst.

5.5 Testovací funkce

Optimalizační problémy pro analýzu výstupů algoritmů byly zvoleny jako minimalizační, a bylo vybráno nebo definováno osm testovacích funkcí f_1 až f_8 s argumenty ve formě atributu `att` a dimenze `dimension`. Pro vytvoření dat pro analýzu v aplikaci byly

algoritmy DE, PSO i SOMA na každé testovací funkci spuštěny 30 krát, s populací o velikosti 50 jedinců v *10D* prostoru. Jako ukončovací podmínka byl zvolen maximální povolený počet ohodnocení účelové funkce *maxCFEs = 5000 x D = 50000* [40].

5.5.1 Dejong1 – f_1

Předpis funkce: $$f(x) = \sum_{i=1}^{D} x_i{}^2 \tag{32}$$

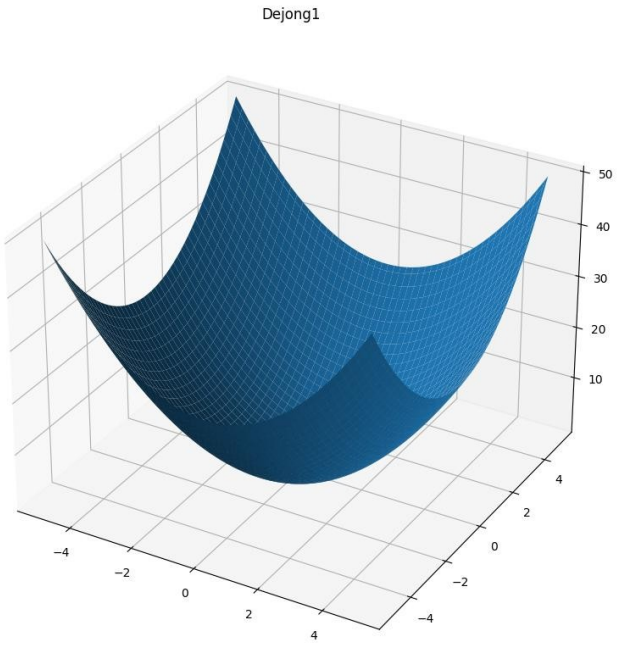
Velikost prostoru: [-5, 5]

Globální minimum 2D: $(x_1, x_2) = (0, 0)$

Funkční hodnota minima 2D: $y = 0$

Zápis v kódu Python:

```
def dejong1(att, dimension):
    cfe = 0
    for i in range(dimension):
        cfe += att[i] ** 2
    return cfe
```
(33)



Obrázek 25: 3D reprezentace funkce f_1 Dejong1

5.5.2 Dejong2 - f_2

Předpis funkce: $$f(x) = \sum_{i=1}^{D-1} 100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2 \quad (34)$$

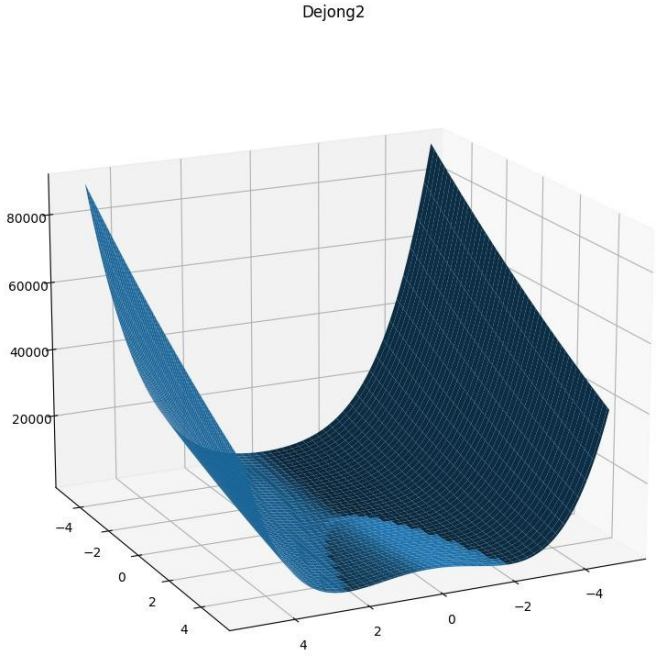
Velikost prostoru: [-5, 5]

Globální minimum 2D: $(x_1, x_2) = (1, 1)$

Funkční hodnota minima 2D: $y = 0$

Zápis v kódu Python:

```
def dejong2(att, dimension):
    n = dimension
    cfe = 0
    for i in range(0, n - 1):
        cfe += 100 * (att[i] ** 2 - att[i + 1]) ** 2 +
(att[i] - 1) ** 2
    return cfe
```
(35)



Obrázek 26: 3D reprezentace funkce f_2 Dejong2

5.5.3 Schwefel - f_3

Předpis funkce: $$f(x) = \sum_{i=1}^{D} -x_i sin\left(\sqrt{|x_i|}\right) \quad (36)$$

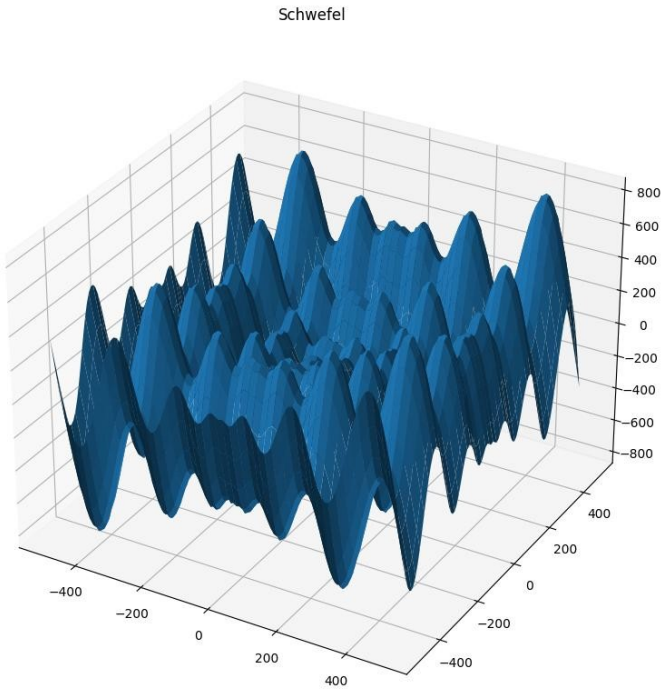
Velikost prostoru: [-500, 500]

Globální minimum 2D: $(x_1, x_2) = (420.969, 420.969)$

Funkční hodnota minima 2D: $y = 2(-418.983)$

Zápis v kódu Python:

```
def schwefel(att, dimension):
    cfe = 0
    for i in range(dimension):
        cfe += (-att[i]) * np.sin(np.sqrt(abs(att[i])))
    return cfe
```
(37)



Obrázek 27: 3D reprezentace funkce f_3 Schwefel

5.5.4 Náhodná funkce (Random) – f_4

Předpis funkce: $$f(x) = \sum_{i=1}^{D} rand\ x_i \tag{38}$$

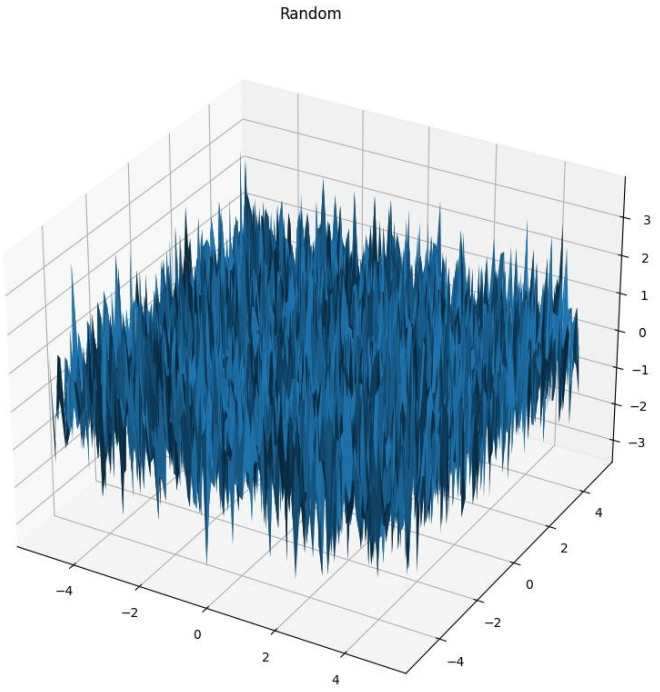
Velikost prostoru: [-5, 5]

Globální minimum 2D: $(x_1, x_2) = (rand, rand)$

Funkční hodnota minima 2D: $y = rand$

Zápis v kódu Python:

```
def random(att, dimension):
    n = dimension
    cfe = 0
    for i in range(dimension):
        cfe = np.random.random()
    return cfe
```
(39)



Obrázek 28: 3D reprezentace funkce f_4 Random

5.5.5 Saturovaná inversní koule (Inversed saturated sphere) – f_5

Předpis funkce:

$$f(x) = \sum_{i=1}^{D} \begin{cases} \text{if } -x_i^2 \leq -10\text{D}: -10\text{D} \\ \text{else}: -x_i^2 \end{cases} \tag{40}$$

Velikost prostoru: [-5, 5]

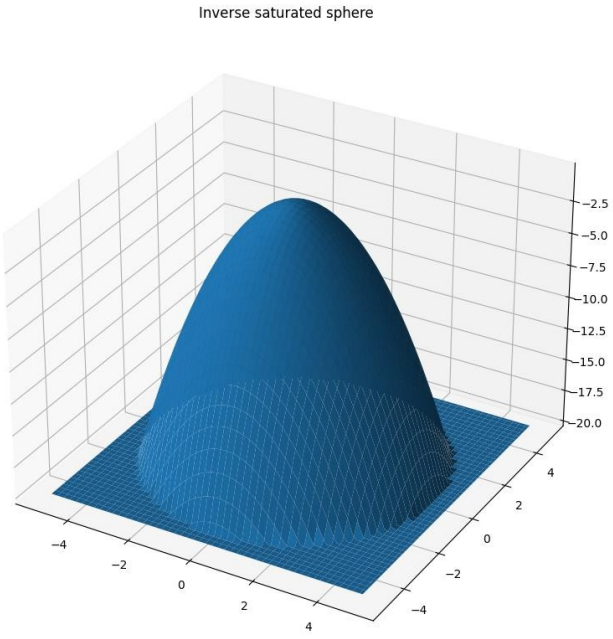
Globální minimum 2D: $(x_1, x_2) = (0, 0)$

Funkční hodnota minima 2D: $y = -20$

Zápis v kódu Python:

```
def invsphere(att, dimension):
    cfe = 0
    for i in range(dimension):
        if -1 * att[i] ** 2 <= -10 * dimension:
            cfe = -10 * dimension
        else:
            cfe = -1 * att[i] ** 2
    return cfe
```

(41)



Obrázek 29: 3D reprezentace funkce f_5 Inverse saturated sphere

5.5.6 Svah (Slope) – f_6

Předpis funkce:
$$f(x) = \sum_{i=1}^{D} x_i \tag{42}$$

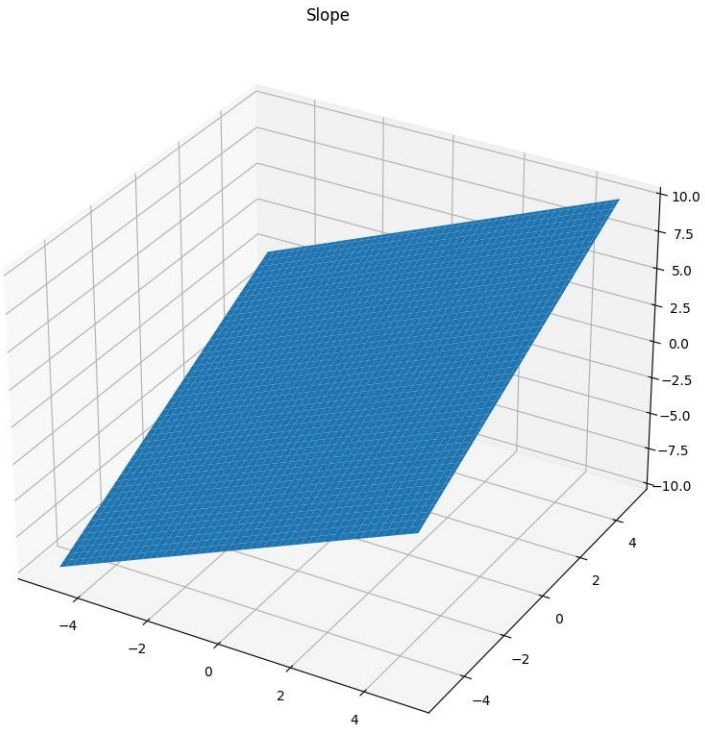
Velikost prostoru: [-5, 5]

Globální minimum 2D: $(x_1, x_2) = (-5, -5)$

Funkční hodnota minima 2D: $y = -10$

Zápis v kódu Python:

```
def slope(att, dimension):
    n = dimension
    cfe = 0
    for i in range(dimension):
        cfe += (-att[i])
    return cfe
```
(43)



Obrázek 30: 3D reprezentace funkce f_6 Slope

5.5.7 Saturovaná koule (Saturated sphere) – f_7

Předpis funkce:

$$f(x) = \sum_{i=1}^{D} \begin{cases} \text{if } x_i^2 > 5D: x_i^2 \\ \text{else: } 5\text{D} \end{cases} \tag{44}$$

Velikost prostoru: [-5, 5]

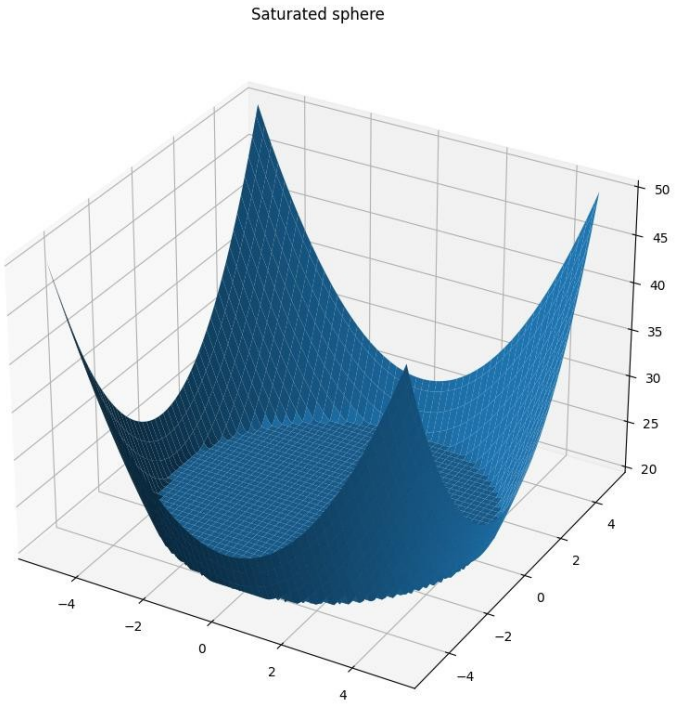
Globální minimum 2D: $(x_1, x_2) = (0, 0)$

Funkční hodnota minima 2D: $y = 0$

Zápis v kódu Python:

```
def satsphere(att, dimension):
    cfe = 0
    for i in range(dimension):
        if att[i] ** 2 <= 5 * dimension:
            cfe = 5 * dimension
        else:
            cfe = att[i] ** 2
    return cfe
```

(45)



Obrázek 31: 3D reprezentace funkce f_7 Saturated sphere

5.5.8 Rovina (Plateau) – f_8

Předpis funkce: $$f(x) = 0 \quad (46)$$

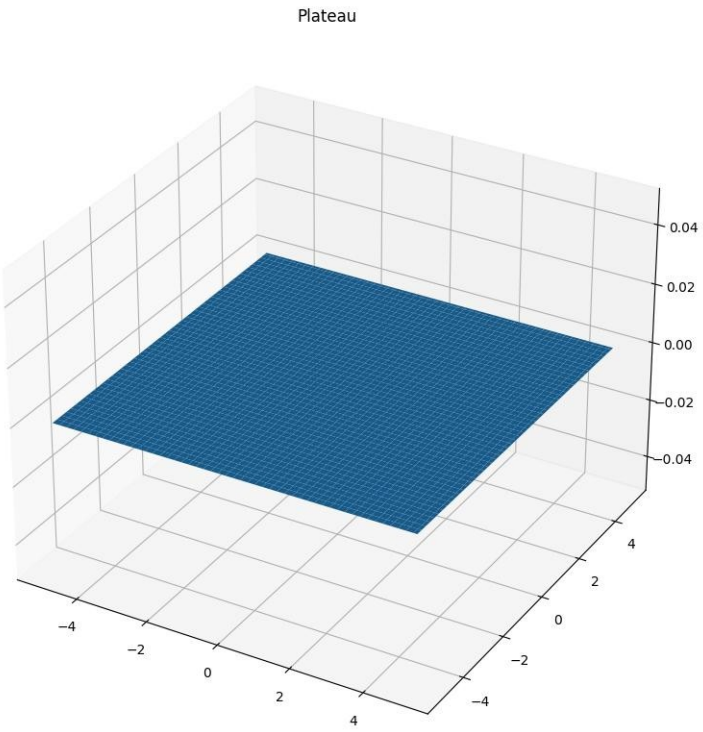
Velikost prostoru: [-5, 5]

Globální minimum 2D: $$(x_1, x_2) = (celý\ prostor)$$

Funkční hodnota minima 2D: $$y = 0$$

Zápis v kódu Python:

```
def plateau(att, dimension):
    n = dimension
    cfe = 0
    for i in range(dimension):
        cfe = 0
    return cfe
```
(47)



Obrázek 32: 3D reprezentace funkce f_8 Plateau

Použité testovací funkce byly zvoleny s ohledem mimo jiné na analýzu rychlosti konvergence (f_1, f_6), škálovatelnosti (f_2), schopnosti úniku z lokálních extrémů (f_3, f_4) a nelezení vícero globálních extrémů (f_5, f_7) či studium zanořování (konvergence populace) v nepřítomnosti gradientu (f_7, f_8).

5.6 Výstupy aplikace

Po zadání a ověření vstupních dat spustí aplikace analýzu vstupů, o čemž uživatele informuje prostřednictvím konzoly. Soubory výstupů jsou v aplikaci zapisovány na konci provedené analýzy. Aplikace vytvoří výstupní data z jedné kombinace souborů se záznamy populací (*Inpput_1*) a souboru záznamů průběhu konvergencí (*Input*_2) prostřednictvím dialogu v aplikaci podle konvence:

`Inpit_1_Input_2_typeofdata.csv`

`Inpit_1_Input_2_typeofdata.jpeg`

Při použití v praxi bude mít název výstupního *csv* souboru se sumarizací pro záznam populací *DE_f4_p.csv* a záznam konvergencí *DE_f1_c.csv* podobu *DE_f1_p_DE_f1_c_summary.csv.* Pouze pro účely této práce, za účelem přehlednosti, je u výstupních souborů *Input_1* název algoritmu a *Input_2* označení testovací funkce. Položka `typeofdata` je stanovena aplikací na základě souvisejících analyzovaných dat. Obrazové výstupní soubory jsou grafy ve formátu jpeg a tabulky s daty ve formátu *csv*. Tato data jsou obsažena ve čtyřech *csv* souborech. Prvním z nich je *summary.csv* obsahující data ze všech běhů analyzovaného algoritmu. Obsahuje hlavičku záznamů, přičemž každý další řádek představuje jeden běh algoritmu. Ukládanými daty jsou index běhu *Run_index*, pravdivostní údaj o předčasné konvergenci daného běhu *premature*, nejlepší dosažená CFE *bestfx*, index vzorku, ve kterém byla nejlepší hodnota CFE dosažena *bestfx_reach*_index. Dalšími sloupci jsou hodnoty popisující procentuální velikost diverzity za celý běh, a její maximální hodnotu, minimální hodnotu, a hodnotu v okamžiku nalezení nejlepší dosažené CFE. V posledním sloupci je uveden nejvyšší počet vytvořených clusterů ze všech běhů algoritmu.

Run_index	Premature	bestfx	bestfx_reach_index	Max_div%	Min_div%	bestfx_div%	Max_clusters
0	T	4.165523021071316e-06	17	56.47108592741481	2.1679146488540773	4.175113455829787	1
1	F	9.72491961515054e-07	84	48.37923043573356	1.596023981496151	1.8445731785633461	2
2	T	2.9153909082424434e-06	74	51.209065157474654	1.1780341215780339	1.3135435573959386	3
3	T	7.34347869912888e-06	59	50.33132979177813	1.3658643929945262	4.026725533904739	2
4	T	6.349511590508783e-06	27	51.664282171032674	1.3275771411017587	2.986608252432895	2
5	T	1.8207895922106587e-06	40	51.81292637501949	1.1074551721908366	1.5639142591684296	1
6	T	1.3143640699153636e-06	1	50.75202085242323	1.0162672016187395	50.75202085242323	2
7	F	2.0765429984415107e-06	87	50.35261036439547	1.109850660570525	3.3272139945321513	1
8	T	2.5372574828930183e-06	80	50.338086377439296	1.5023527403523065	1.6033313991422282	2
9	F	2.890391884724508e-07	2	51.712235947212235	0.8963485792749804	10.204522410410737	1
10	F	1.5670129693834411e-07	28	50.947530436330894	0.8221288380186885	2.584889569060531	3

Obrázek 33: Náhled obsahu souboru *PSO_f4_summary.csv*

Dalšími třemi soubory jsou záznamy průběhu diverzity populace *diversities.csv*, vývoje clusteringu *clustering.csv* a vývoje průměrné velikosti skoku v populaci *avg_jumps.csv*. Soubory neobsahují hlavičku, řádky představují počet běhů algoritmu, jednotlivé záznamy v řádku jsou odděleny středníky. Jako doplněk k *csv* souborům jsou aplikací vytvořeny během analýzy také šest obrazových souborů. Soubor *avg_jumps.jpeg* obsahuje graf procentuálně vyjádřených průměrných velikostí skoků jedinců mezi vzorky populací za všechny běhy algoritmu, *premature.jpeg* ukazuje výsledky analýzy předčasné konvergence, *characteristics.jpeg* obsahuje skupinu grafů, nejhorší a nejlepší průběh konvergencí, diverzity populací, clusteringů a velikosti průměrných skoků pro obě konvergence. V grafu *characteristics.jpeg* jsou svislými čarami zobrazeny okamžiky dosažení nejlepšího řešení pro nejlepší i nejhorší průběh. Grafy v souboru *characteristics.jpeg* vztahující se k nejlepšímu průběhu jsou zobrazeny modře, nejhorší červeně.

Obrázek 34: Grafy v souboru characteristics.jpeg

Dalšími obrazovými soubory jsou konvergencí za všechny běhy algoritmu *convergences.jpeg*, graf diverzit populací *diversities.jpeg* a graf tvorby clusterů v populacích. Celkem výstup obsahuje 6 obrazových souborů a 4 soubory *csv*. Pro všechny kombinace algoritmů a testovacích funkcí f_1 až f_8 je v této práci vytvořeno 96 csv souborů a 144 obrazových souborů *jpeg*, celkem 240 souborů.

6 FUNKCE APLIKACE

6.1 Konvergence a předčasná konvergence

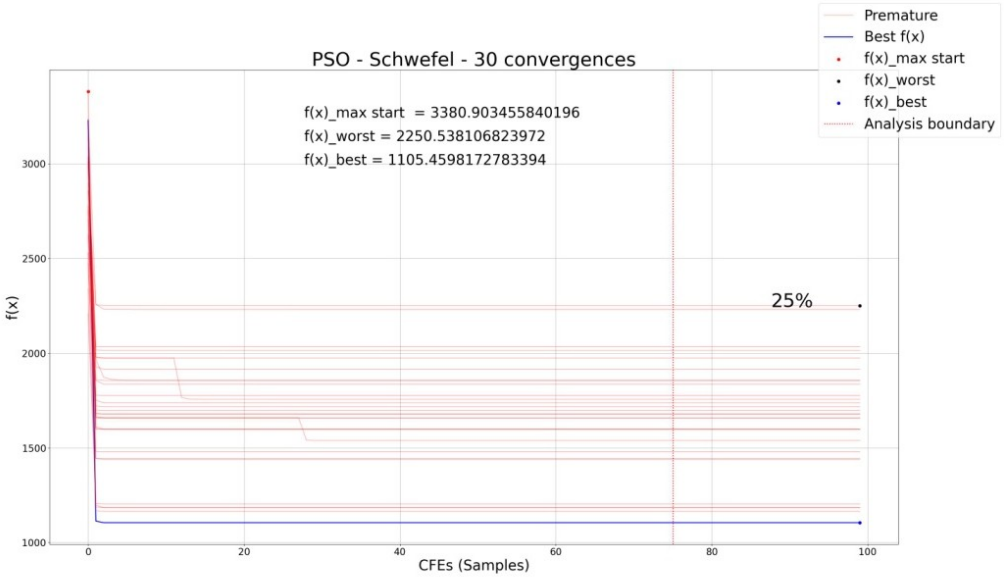
U všech použitých testovacích funkcí jsou známa globální minima nebo oblasti představující plochu globálního minima. U těchto funkcí lze analýzou průběhů konvergencí určit, kdy došlo k uváznutí v lokálním minimu a předčasné konvergenci. Uváznutí je charakteristické průběhem konvergence rovnoběžným s časovou, vzorkovací osou, a to od posledního okamžiku změny do okamžiku ukončovací podmínky s CFE výrazně odlišnou od hodnoty globálního optima. Do pole konvergencí je v algoritmu ukládána vždy jen doposud nejlepší nalezená hodnota jedince a lze tak určit okamžik změny. V okolí těchto okamžiků lze sledovat chování a vlastnosti populace při pohybu nahyperploše, a určit tak možné příčiny nalezení nesprávného optima. Analyzátor konvergence obsahuje nástroj pro odhad předčasné konvergence na základě porovnávání hodnot CFE stanovené části konce běhu algoritmu s nejlépe dosaženou hodnotou v rámci všech opakování. V kódu analyzátoru není stanovena hodnota globálních optim testovacích funkcí, z důvodu použitelnosti aplikace na výstupech algoritmů nasazených na reálných optimalizačních problémech s neznámými hodnotami extrémů. Z tohoto důvodu je výsledkem analýzy pouze odhad předčasné konvergence. Velikost zkoumané oblasti je určena na posledních 25% počtu vzorků konvergencí, které jsou ve smyčce ukládány do samostatných polí ke komparaci. K provedení porovnání je aplikací vypočtena tolerance výsledků CFE ve zkoumané oblasti na principu zaokrouhlování, jehož míra je deterministicky stanovena v závislosti na rozsazích CFE na začátku a na konci běhu algoritmu. Jsou porovnávány hodnoty nejvyšší první hodnoty *CFEmaxstart* s nejnižší koncovou hodnotou *CFEbest* pro tvorbu tolerance pro rozlišení nejlepší dosažené konvergence od výrazně horších, a dále nejvyšší koncové hodnoty *CFEworst* a nejlepší dosažené *CFEbest* pro tvorbu tolerance pro stanovení neměnného průběhu konvergence ve zkoumané oblasti. Tolerance pracují na principu zaokrouhlování k nejbližšímu číslu o 3 řády níže, než je rozdíl porovnávaných hodnot.

$$CFEmaxstart - CFEbest = xx.xxxxxxxxx$$

$$Distinguis\ ing\ tolerance => compare\ all\ rounded\ to\ xx.x \tag{48}$$

$$CFEworst - CFEbest = x.xxxxxx$$

$$Continuous\ tolerance => compare\ all\ rounded\ to\ x.xx \tag{49}$$

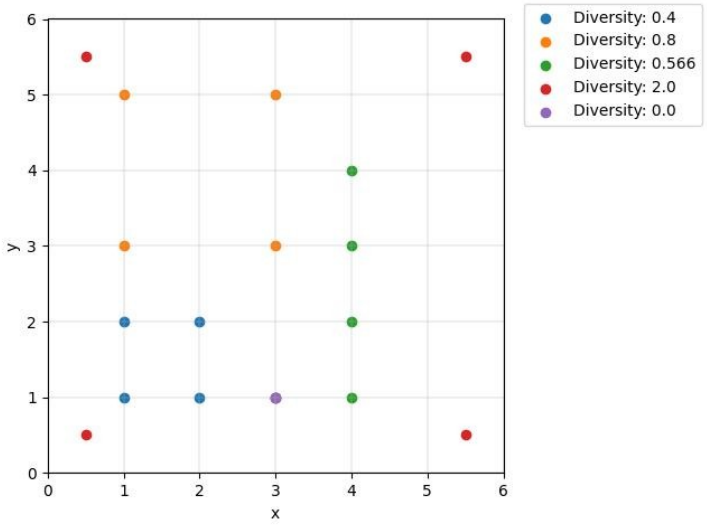


Obrázek 35: Příklad vyhodnocení předčasné konvergence

6.2 Diverzita populace

K analýze diverzity populace je v aplikaci použita metoda vzdálenosti k průměrnému bodu (Distance to Average Point, DAP), která bere v úvahu velikost prohledávaného prostoru podle vzorce uvedeného v teoretické části této práce, v kapitole Diverzita populace. Za tímto účelem byla v modulu `analyzer.py` vytvořena funkce `disttoavg`, která ve smyčce pracuje postupně se všemi atributy každého jedince v populaci za účelem výpočtu průměrného bodu `avgpoint`. V další smyčce jsou atributy průměrného bodu srovnávány s odpovídajícími atributy všech jedinců v populaci pomocí vzorce DAP. Výsledná hodnota je uložena do pole `diversitydata` pro všechny vzorky populací ve všech opakováních. Vyjádření diverzity populace v grafech představuje procentuální hodnotu maximální teoretické diverzity populace, která byla vypočtena pro populaci 50 jedinců v 10*D* prostoru v situaci, kdy mají všechny atributy poloviny populace hodnotu minima, a atributy druhé poloviny populace hodnoty maxima. Průměrný bod se nachází uprostřed diagonály prostoru. Pro obě varianty velikostí prohledávaného prostoru [-5, 5] a [-500, 500] byla podle vzorce DAP experimentálně vypočtena stejná hodnota *diversityMax* = 1.1180339887498942. Aplikace vypočítá hodnotu *diversityMax* podle parametrů vstupních souborů a uživatelsky zadané velikosti prohledávaného prostoru. Velikosti diverzit populací jsou ukládány do pole hodnot `diversitydata` pro všechny vzorky populací ve všech opakováních, ve formě procentuální velikosti maximální teoretické diverzity populace. Nulová hodnota diverzity populace podle tohoto výpočtu

znamená situaci, kdy mají všichni jedinci populace shodné všechny své odpovídající atributy, nacházejí se v jednom bodě. V souboru *diversities.jpeg* jsou znázorněny procentuální velikosti diverzit populace pro každý vzorek, v každém běhu, záznamy diverzit pro všechny běhy algoritmu jsou uloženy v souboru *diversities_jumps.csv*.



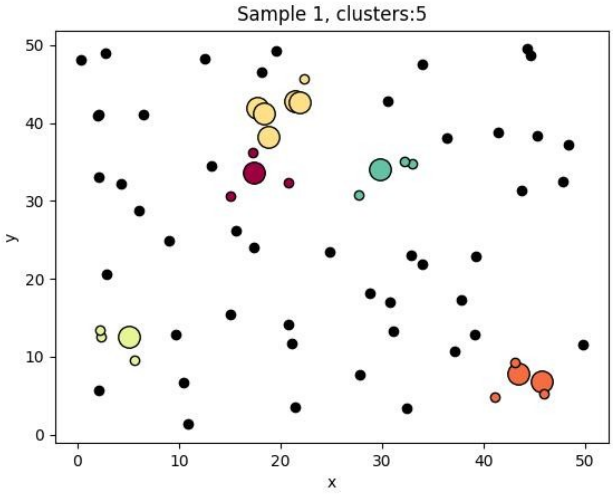
Obrázek 36: Příklad různých hodnot diverzity pěti populací o čtyřech jedincích ve 2*D* reprezentaci v prostoru [0, 6]

```
DIVERSITY:
diversitydata len 2
diversitydata [[25.09719368053059, 29.242536081190437], [24.796786493413773, 27.189297052303843]]
diversitydata0 [25.09719368053059, 29.242536081190437]
diversitydata00 25.09719368053059
```

Obrázek 37: Příklad výpisu konzoly s polem polí záznamů reálných hodnot diverzit dvou vzorků populací ve dvou opakováních s adresováním podle indexů

6.3 Clustery v populaci

Pro detekci vytváření clusterů je v aplikaci využit nástroj DBSCAN z knihovny `sklearn.cluster`, který poskytuje grafické i numerické výstupy. Grafické výstupy jsou tvořeny grafy a reprezentují pouze vzájemné vztahy dvou atributů jedince. Pro analýzu vzorků populací v 10*D* prostoru je pro účely této práce použita numerická reprezentace. Hodnoty parametrů N_{Eps}= *1% searchspace*, *MinPts* = *4* [31]. Metrika nástroje DBSCAN byla ponechána výchozí, *default*, a odpovídá měření euklidovských vzdáleností jedinců v sousedství.



Obrázek 38: Příklad 2*D* reprezentace clusteringu v náhodné populaci

Podobně jako u záznamů diverzit populací je pro každý vzorek populace jednotlivých běhů algoritmu zaznamenáván počet vzniklých clusterů do pole `cludata`. Údaje o počtu clusterů jsou součástí výstupního souboru `summary.csv` a tvoří také samostatný soubor `clustering.csv`.

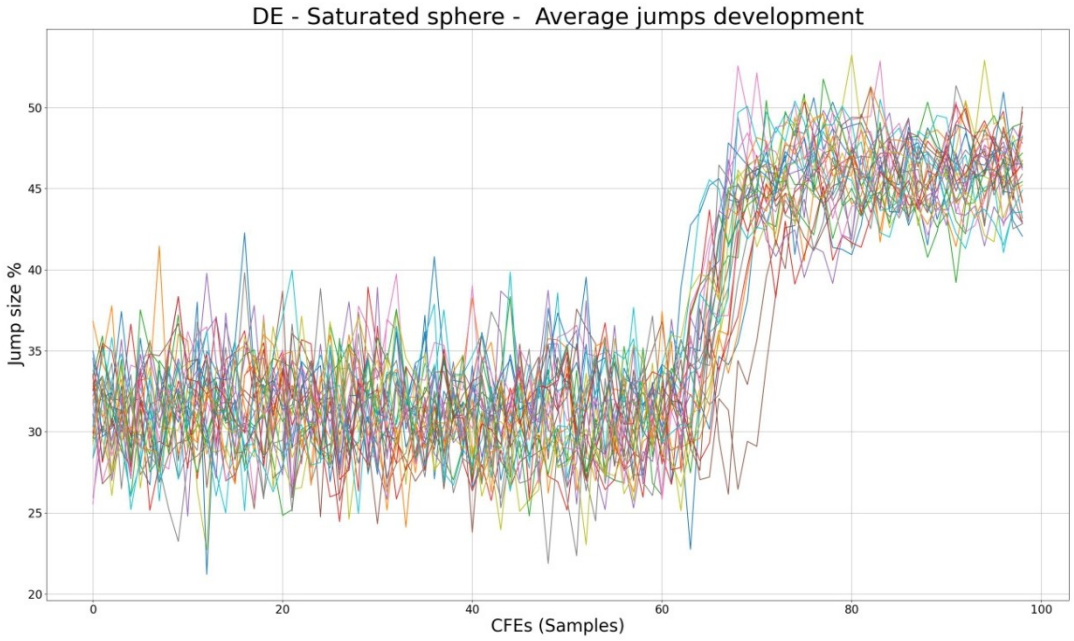
6.4 Dynamika pohybu

Při analýze dynamiky je v aplikaci sledován pohyb jedinců po hyperploše výpočtem euklidovské vzdálenosti polohy jedince ve všech dimenzích mezi po sobě jdoucími vzorky. Ve smyčce jsou tyto rozdíly zpracovávány u všech jedinců v populaci, v každém vzorku a každém opakování a jsou ukládány do pole polí `jumpsdata`. Výsledkem analýzy jsou všechny jednorázově překonané vzdálenosti jedinců, skoky. Maximální teoretická velikost skoku pro prohledávaného prostoru [-5, 5] v *10D* je velikost diagonály prohledávaného prostoru s hodnotou *Jumpmax* = 31,62277, pro velikost prostoru [-500, 500] *Jumpmax* = 3162,27766. Velikost diagonály prostoru je vypočítána podle následujícího vztahu:

$$Jump_{max} = \sqrt{\sum_{j=1}^{D} \left(x_{j_{max}} - x_{j_{min}}\right)^2} \tag{50}$$

Aplikace vypočítá hodnotu *Jumpmax* podle uživatelsky zadané velikosti prohledávaného prostoru. V souboru *avg_jumps.jpeg* jsou znázorněny průměrné velikosti skoků jedinců

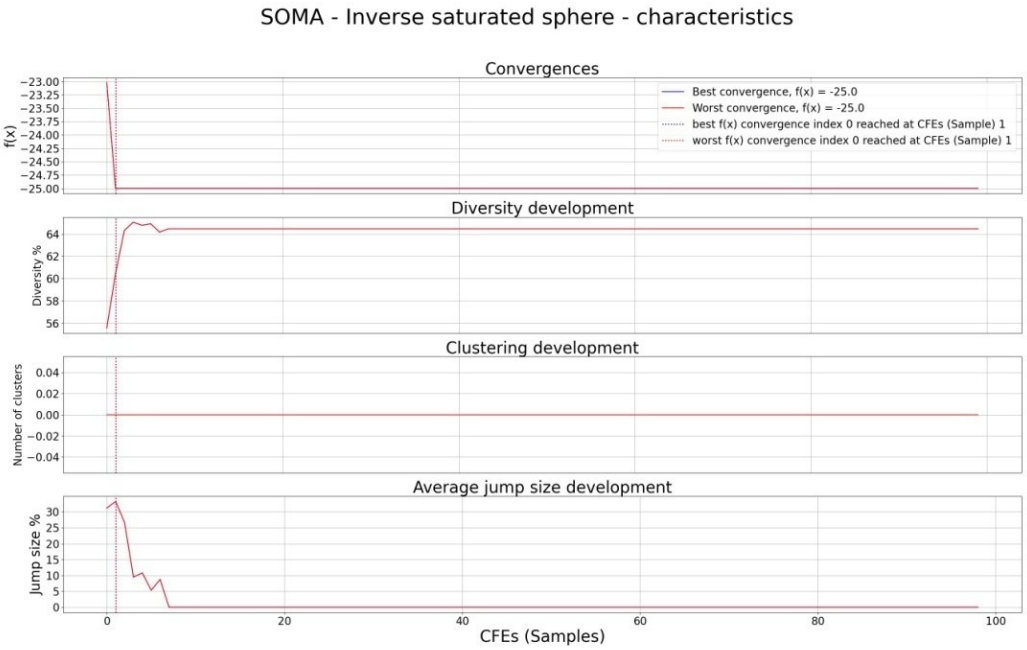
po hyperploše pro každý vzorek, v každém běhu, záznamy průměrných skoků pro všechny běhy algoritmu jsou uloženy v souboru *avg_jumps.csv.* Tato data jsou podpůrnými informacemi k analýze clusteringu, diverzity a předčasné konvergence, a vyjadřují míru explorace v každé konvergenci algoritmu. Velikost skoku je v grafech znázorněna jako procentuální velikost maximálního teoretické ho skoku *Jumpmax*.



Obrázek 39: Průměrné velikosti skoků ve všech opakováních u DE na funkci f_5

7 INTERPRETACE VÝSLEDKŮ

Výsledky analýzy jsou hodnoceny v případech, kdy došlo k výraznému selhání konvergencí algoritmu v porovnání s ostatními dvěma. Bude hodnocena míra předčasné konvergence v kontextu s tvorbou clusterů a diverzitou populace. Pro daný případ budou komentovány výsledky analýzy pro trojici algoritmů na jedné funkci. Hodnocení výsledků analýzy se vždy vztahuje pouze na konkrétní instanci algoritmu, v dalším textu práce jsou reference na algoritmus vždy referencemi na instance daného typu algoritmu s konkrétními řídícími parametry, použité pro tvorbu vstupů aplikace v této práci. Při provádění analýzy nastaly situace, kdy algoritmus díky počátečnímu rovnoměrnému rozložení nalezl globální optimum již v druhém vzorku. Díky tomu nebyly deterministické nástroje analyzátoru schopny vyhodnotit nejlepší ani nejhorší průběhy konvergencí, diverzit a clusteringů populací, a zobrazené křivky znázorňují průběhy charakteristik vztahujících se pouze ke konvergenci s indexem 0.



Obrázek 40: Příklad grafu charakteristik s vyhodnocením pouze pro konvergenci s indexem 0

Všechny výstupní soubory a grafy pro všechny kombinace funkcí a algoritmů jsou součástí přílohy této práce.

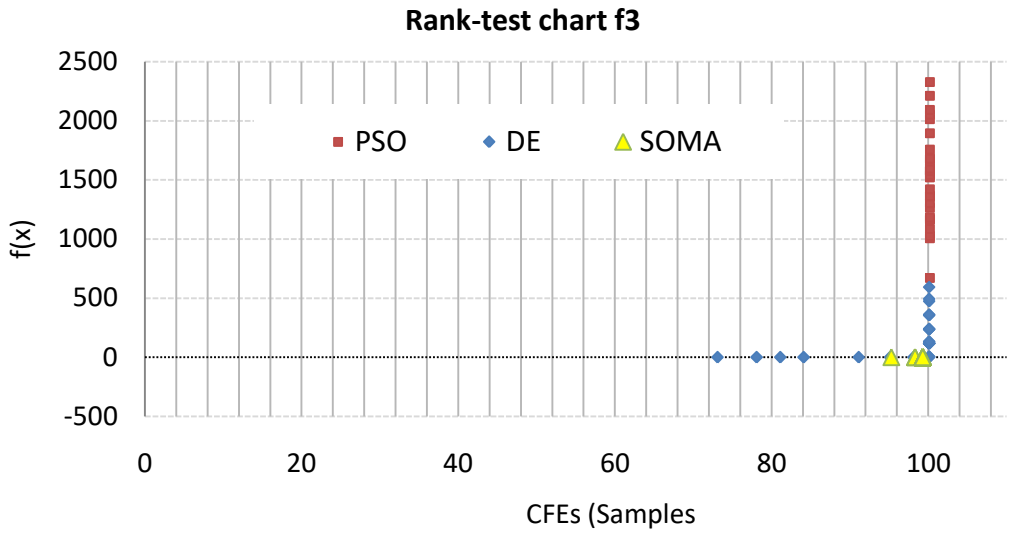
7.1 Robustnost algoritmu

Stanovení této charakteristiky není výstupem vytvořené aplikace, avšak využívá data shromažďovaná v průběhu analýzy, a je zde uvedena pro doplnění kontextu zkoumaných dat. Pro účely této práce je robustnost algoritmu stanovena jako velikost dosaženého skóre Rank-testu trojice algoritmů pro jednotlivé testovací funkce.

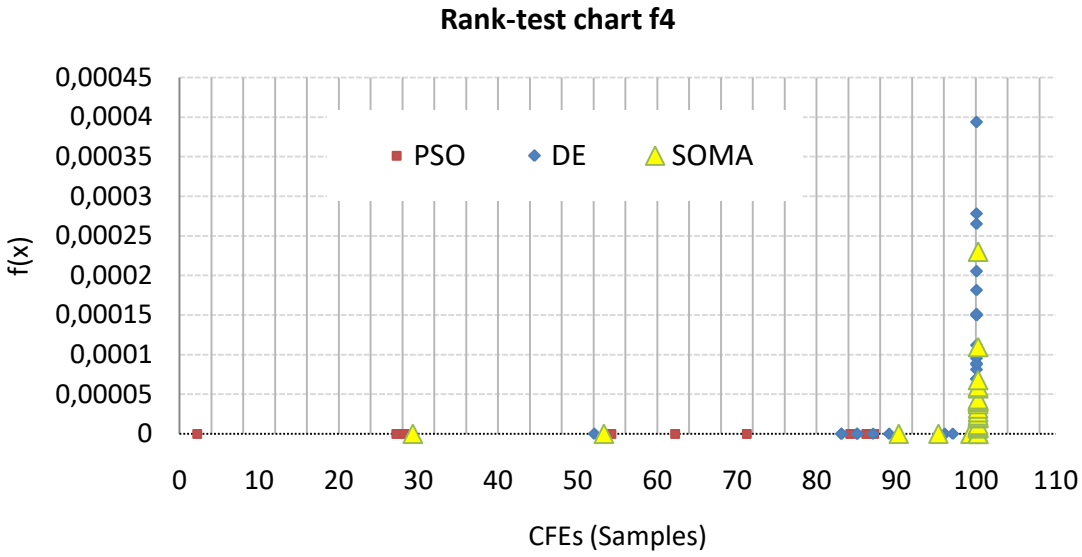
Tabulka 4: Výsledky Rank-testu na všech testovacích funkcích

	DE	PSO	SOMA
f_1	1422	2006	667
f_2	1365	465	2265
f_3	1574	465	2056
f_4	1889	1500	1706
f_5	2265	2265	2265
f_6	1469	1725	901
f_7	2265	465	1365
f_8	0	0	0
Celkem	12246	8892	11227

Rank-test je v této práci použit jako podpora k rozlišení algoritmů se slabými místy, samotný výkon není pro analýzu relevantní. Výsledky ukazují míru univerzálnosti a stability výstupů pro f_1 až f_8. Hodnota přiblížení ke globálnímu optimu je odvozena z analýzy konvergence uvedené v kapitole Konvergence a předčasná konvergence, a je rovna hodnotě *distinguishing tolerance*, která definuje dosaženou hodnotu, která je rovna nejlepší hodnotě CFE, pokud se její hodnota nachází v toleranci 10^{-5} . Tato hodnota je v souboru *summary.csv* ve sloupci *premature.csv* označena jako *F* (False). Místo v průběhu konvergence, ve kterém došlo k dosažení této hodnoty je v souboru určeno sloupcem *bestfx_reach_index*. Hraniční hodnotu představuje ukončovací podmínka dosažení povoleného počtu provedených ohodnocení účelové funkce, a je reprezentována posledním vzorkem konvergence s indexem 99, ačkoliv vizualizace v grafu ukazuje na hodnotu rovnající se počtu vzorků 100.



Obrázek 41: Příklad vizualizace Rank-testu pro funkci f_3

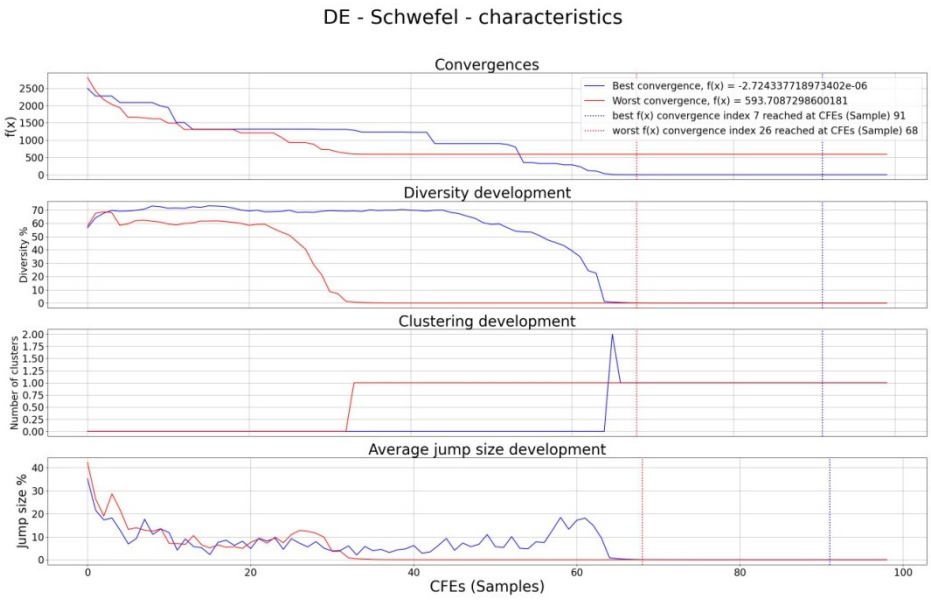


Obrázek 42: Příklad vizualizace Rank-testu pro funkci f_4

7.2 Výstupy pro f_3 Schwefel

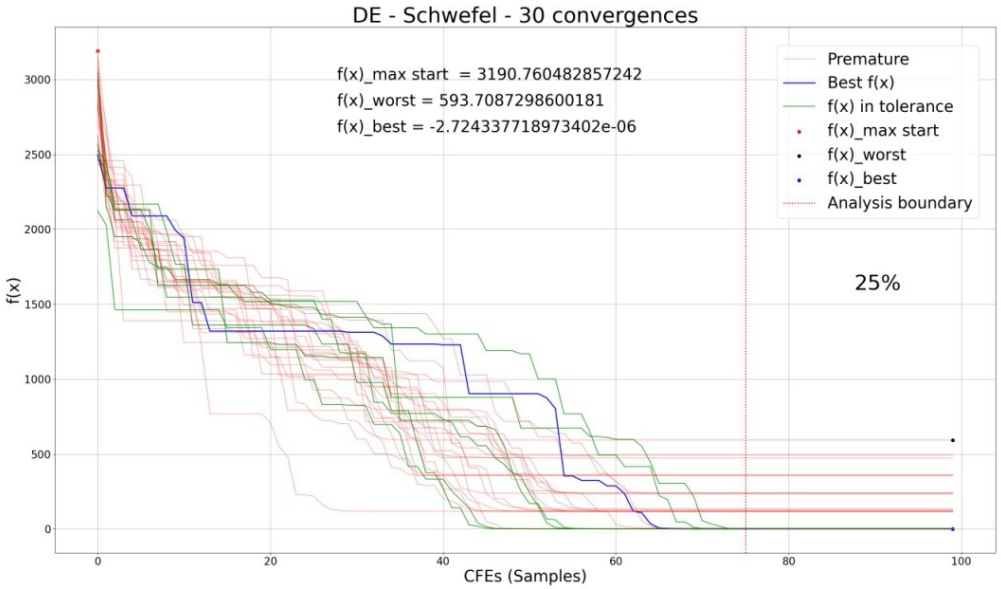
7.2.1 DE

Algoritmus DE se v Rank-testu umístil na druhém místě. Analýza detekuje vznik clusterů, skokový pokles diverzity při jejich vzniku, a minimalizaci průměrného pohybu jedinců.

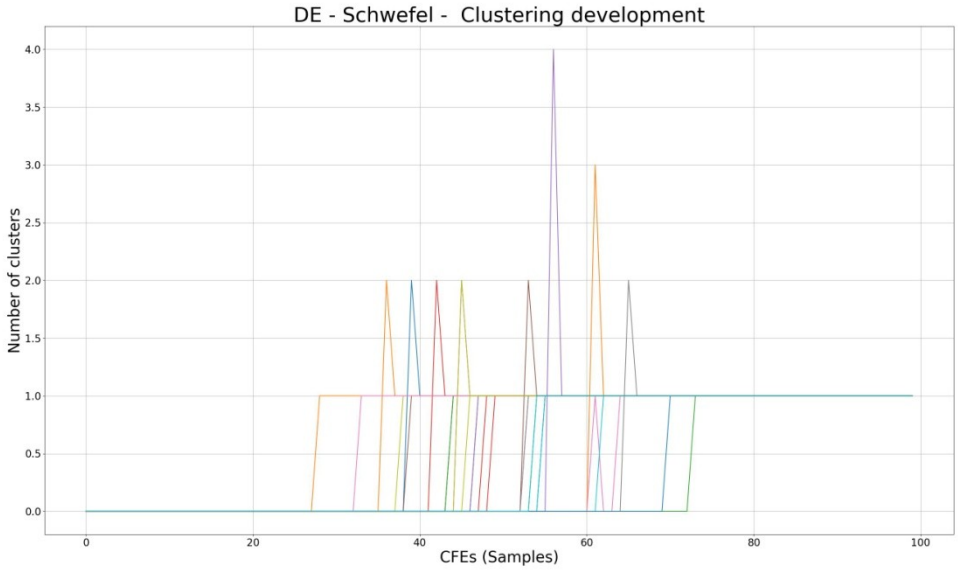


Obrázek 43: Charakteristiky algoritmu DE na funkci f_3

Clustery vznikají v každém běhu algoritmu, jejich maximální počet je 4. U nejhoršího průběhu konvergence s indexem běhu 26, došlo v bodě *CFEs (Samples)* = 34 ke vzniku jednoho clusteru, poklesu diverzity o 8,6881%, a snížení velikosti průměrného skoku na 0,8139%. Ve zbývající části tohoto běhu nedošlo k nalezení lepší hodnoty *f(x)*, a díky převaze exploitace došlo k uváznutí v lokálním minimu na hodnotě *f(x)* = 593,7087. V nejlepším průběhu s indexem běhu 7 konvergence DE v bodě *CFEs (Samples)* = 66 došlo ke vzniku jednoho, krátkodobě dvou clusterů, poklesu diverzity populace o 12,1338%, a snížení velikosti průměrného kroku na 0,4471 %. Nejlepší nalezené řešení pro všechny běhy má hodnotu *f(x)* = -2,7243x10^{-6}. Pro účely této práce lze díky znalosti globálního optima problému konstatovat, že vznik clusteru a ztráta diverzity v tomto běhu vedly k aproximaci nejlepšího možného řešení v bodě *CFEs (Samples)* = 91. Vznik clusterů v počátečních fázích konvergence DE spojený se značným poklesem diverzity populace může ukazovat na exploitativní tendence algoritmu. Spolu s výsledky analýzy předčasných konvergencí všech běhů, tvořících 76,66% jejich celkového počtu, lze převahu exploiatce nad explorací označit jako slabé místo algoritmu.



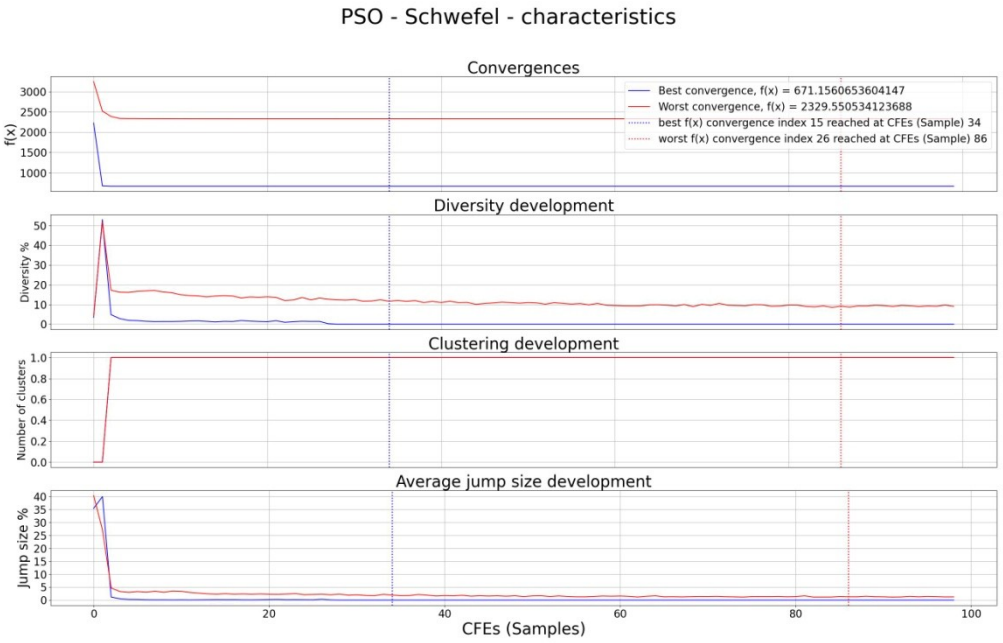
Obrázek 44: Analýza předčasné konvergence algoritmu DE na funkci f_3



Obrázek 45: Analýza clusteringu algoritmu DE na funkci f_3

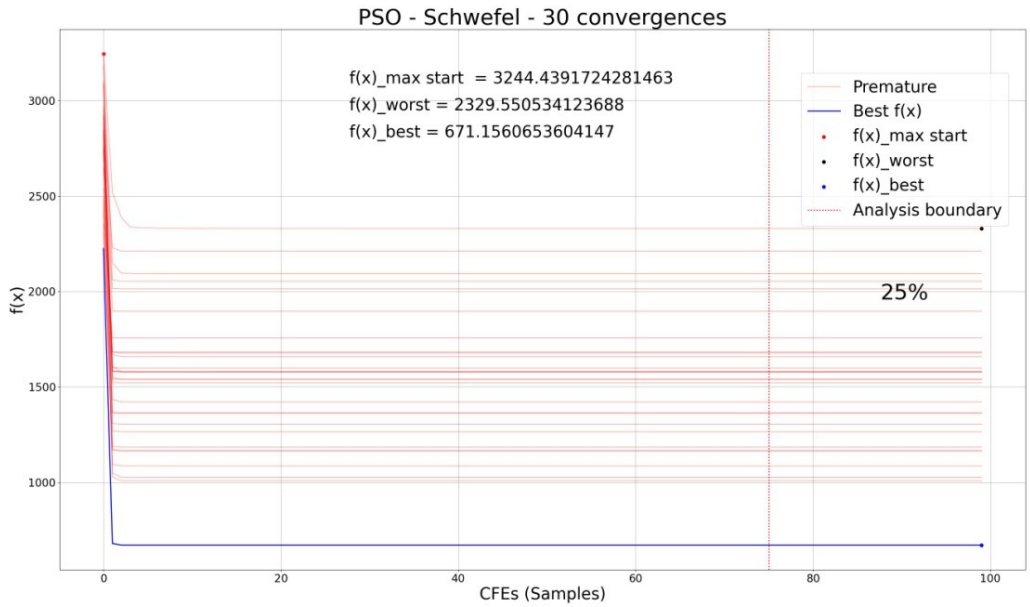
7.2.2 PSO

Algoritmus PSO získal se v Rank-testu umístil na třetím místě. Jeden cluster vzniká u všech běhů v bodě *CFEs (Samples)* = 2.



Obrázek 46: Charakteristiky algoritmu PSO na funkci f_3

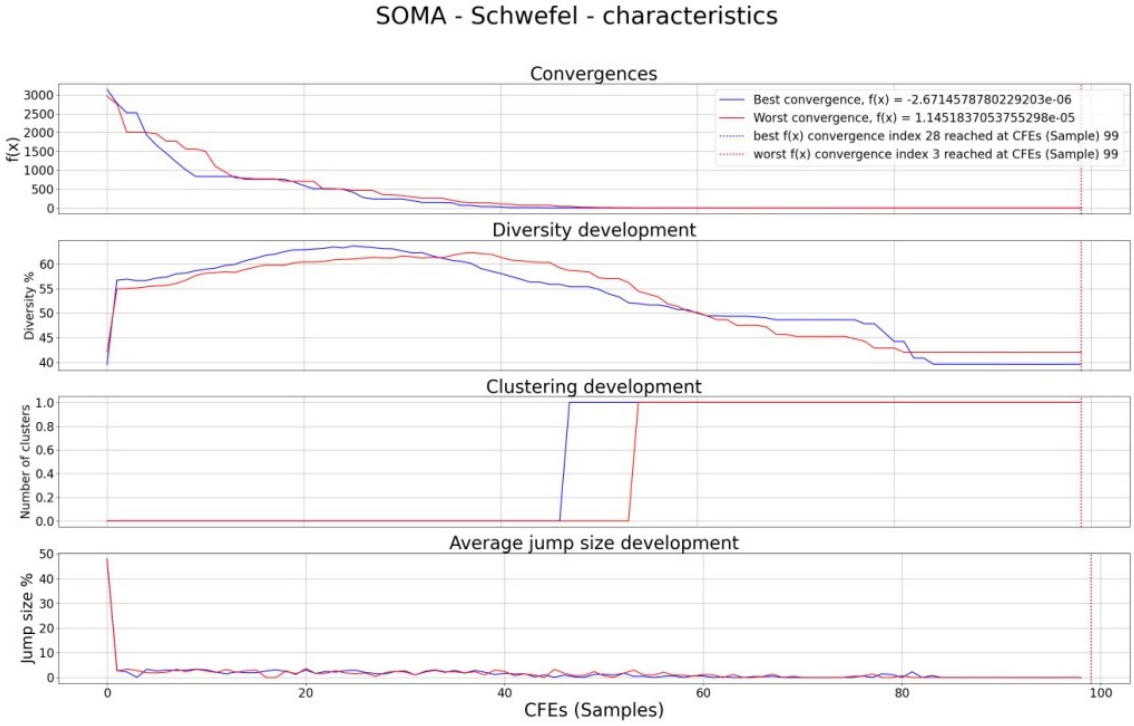
V nejhorším průběhu konvergence s indexem 26 přetrvává po celou dobu běhu 1 cluster v populaci, jehož vznik doprovází výrazný pokles diverzity o 34,7245% a zastavení konvergence algoritmu. Velikost průměrného skoku v populaci klesla na hodnotu 2,9847%. Ve zbývající části tohoto běhu nedošlo k nalezení lepší hodnoty *f(x)*. Z těchto údajů vyplývá, že došlo k uváznutí v lokálním minimu na hodnotě *f(x)* = 2329.5505. Nejlepší průběh konvergence PSO s indexem 15 ukazuje shodný průběh tvorby clusteru v bodě *CFEs (Samples)* = 2, diverzita populace poklesla po jeho vzniku o 48,2099%, velikost průměrného kroku na 2,9847%. Nejlepší nalezené řešení pro všechny běhy má hodnotu *f(x)* = 671.1560. Pro účely této práce lze díky znalosti globálního optima problému konstatovat, že v žádném z běhů PSO nedošlo k nalezení nebo aproximaci nejlepšího možného řešení. Vznik clusteru a ztráta diverzity v tomto běhu vedly k uváznutí v lokálním extrému. Spolu s výsledky analýzy předčasných konvergencí všech běhů, tvořících 100 % jejich celkového počtu, lze výraznou převahu exploiatce nad explorací označit jako slabé místo algoritmu.



Obrázek 47: Analýza předčasné konvergence algoritmu PSO na funkci f_3

7.2.3 SOMA T3A

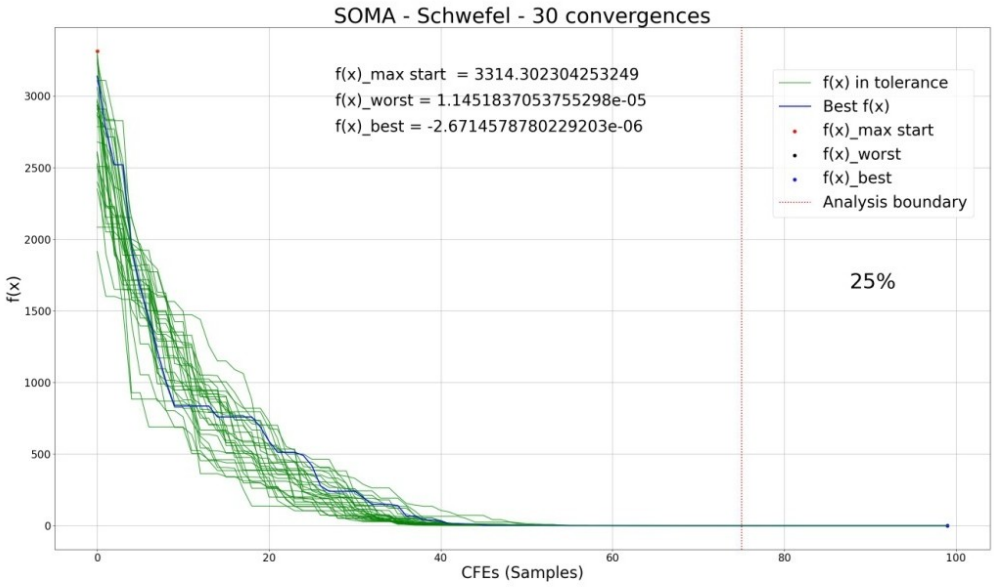
Algoritmus SOMA T3A byl Rank-testem vyhodnocen jako nejlepší na této funkci.



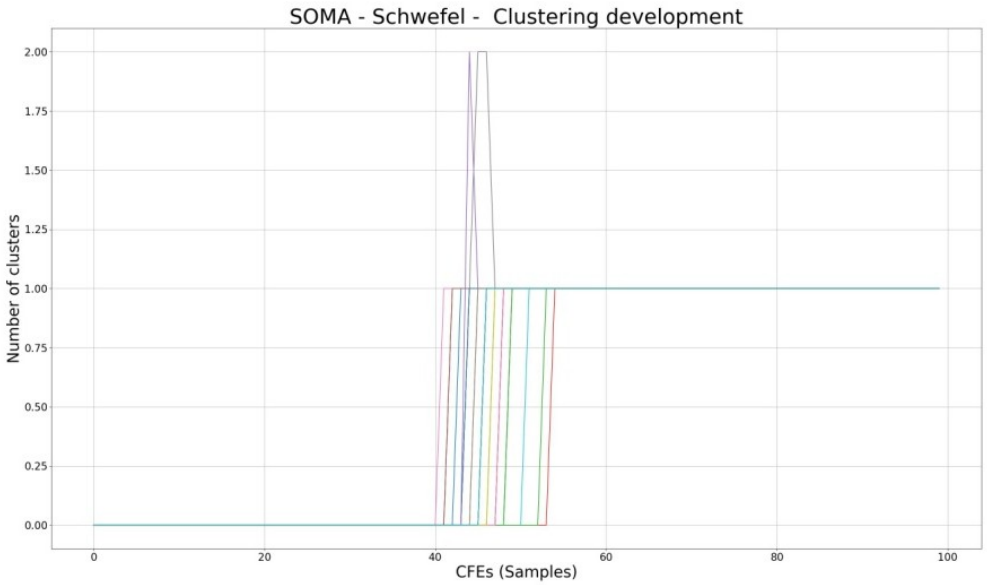
,

Obrázek 48: Charakteristiky algoritmu SOMA na funkci f_3

Clustery vznikají v každém běhu algoritmu, jejich maximální počet je 2. Pro nejhorší průběh konvergence s indexem běhu 3, došlo v bodě *CFEs (Samples)* = 54 ke vzniku jednoho clusteru, diverzita v tomto bodě představuje hodnotu 54,4437%, velikost průměrného skoku s hodnotou 2,9804%. Velikost skoků má sestupnou tendenci odpovídající postupnému snižování diverzity populace. K nalezení nejnelepšího řešení $f(x) = 1{,}1451 \times 10^{-5}$ došlo v bodě *CFEs (Samples)* = 99. V nejlepším průběhu konvergence s indexem běhu 28 vznikl v bodě *CFEs (Samples)* = 47 jediný cluster, diverzita populace dosahuje 55,3533%, velikost průměrného kroku má hodnotu 0,9501%. Obě hodnoty se však v dalších částech běhu algoritmu snižují a jejich sestupná tendence odpovídá postupnému snižování diverzity populace. Nejlepší nalezené řešení pro všechny běhy má hodnotu $f(x) = -2{,}6714 \times 10^{-6}$ a bylo nalezeno v bodě *CFEs (Samples)* = 99. Pro účely této práce lze díky znalosti globálního optima problému konstatovat, že došlo k aproximaci nejlepšího možného řešení. Spolu s výsledky analýzy předčasných konvergencí všech běhů, tvořících 0 % jejich celkového počtu, se tato instance algoritmu SOMA T3A jeví jako vhodně zvolená na daný problém a z analýzy nejsou patrná žádná slabá místa.



Obrázek 49: Analýza předčasné konvergence algoritmu SOMA na funkci f_3



Obrázek 50: Analýza clusteringu algoritmu SOMA na funkci f_3

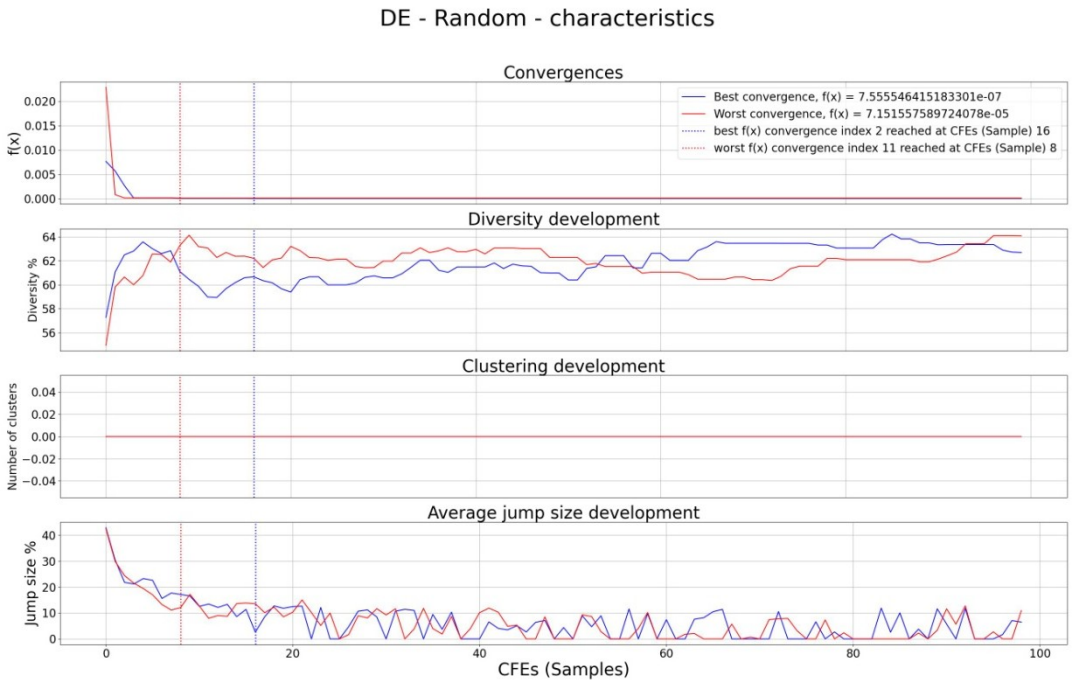
7.2.4 Srovnání analýz algoritmů

Instance algoritmu DE vykazovala prudký pokles diverzity po vytvoření clusterů, což mělo za následek uváznutí v lokálním extrému, oproti SOMA T3A, jejíž průběhy diverzit měly rovnoměrně sestupný charakter i v okamžicích vzniku clusterů, až do okamžiku nalezení nejlepších řešení. PSO ve srovnání s oběma předešlými algoritmy ukazuje prudkou ztrátu diverzity populace v jednom clusteru na začátku běhu. Algoritmus SOMA T3A dokázal jako jediný nalézt globální minimum problému v obou případech hodnocených konvergencí, DE pouze ve své nejlepší konvergenci, PSO globální extrém nenalezlo ani v jedné konvergenci.

7.3 Výstupy pro f_4 Random

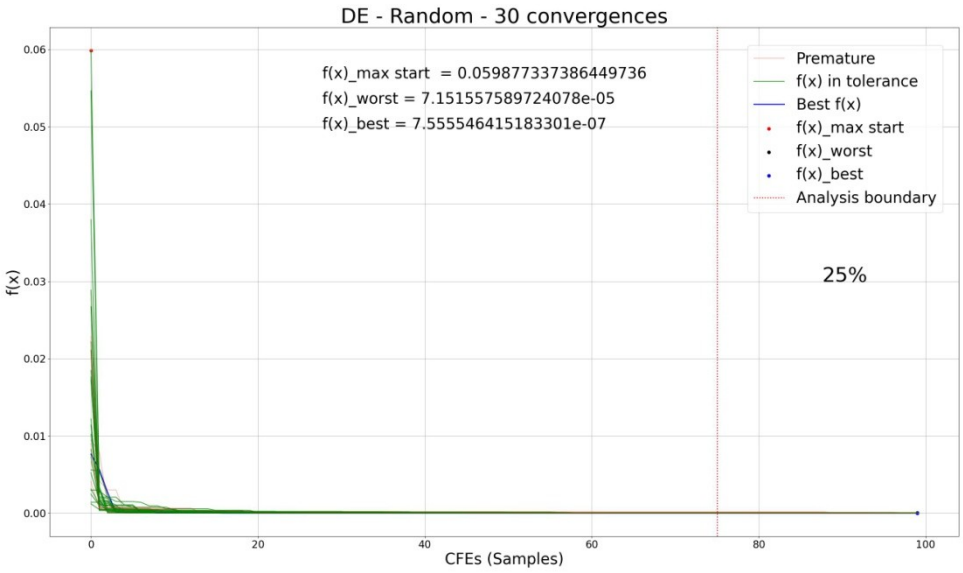
7.3.1 DE

Algoritmus DE byl Rank-testem vyhodnocen jako nejlepší na této funkci. V žádném běhu algoritmu nevznikají clustery.



Obrázek 51: Charakteristiky algoritmu DE na funkci f_4

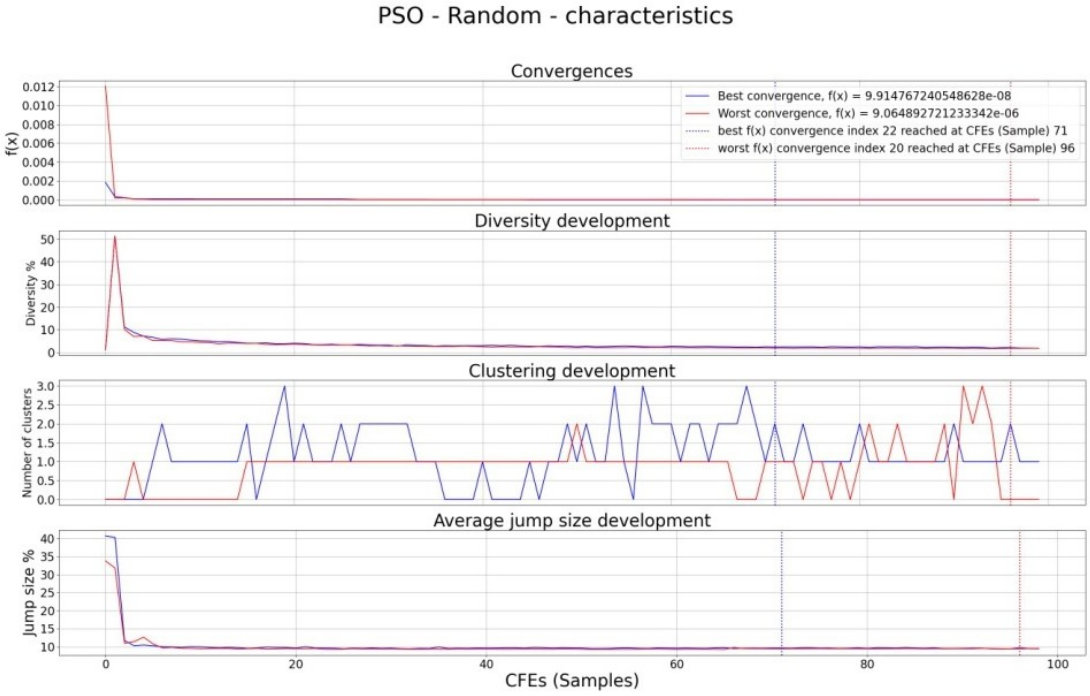
Pro nejhorší průběh konvergence s indexem běhu 11, došlo k nalezení nejlepší hodnoty $f(x) = 7{,}1515\text{x}10^{-5}$ v bodě *CFEs (Samples)* = 8, diverzita populace měla hodnotu 63,2679%. V nejlepším průběhu konvergence s indexem běhu 2 byla nalezena nejlepší hodnota $f(x) = 7{,}5555\text{x}10^{-7}$ v bodě *CFEs (Samples)* = 16, diverzita populace měla hodnotu 60,6588%. Oba průběhy vykazují postupný nárůst diverzity populací s několika poklesy, přičemž na konci obou běhů měla diverzita hodnotu pro nejhorší běh 64,0829%, pro nejlepší 62,6859%. Průměrné velikosti průměrných skoků od okamžiků nalezení nejlepších řešení až do konce běhů měly pro nejhorší průběh velikost 5,090694%, pro nejlepší 4.729177%. Pro účely této práce lze díky znalosti globálního optima problému konstatovat, že došlo k aproximaci nejlepšího možného řešení. V obou průbězích lze pozorovat rovnováhu exploiatce a explorace. Spolu s výsledky analýzy předčasných konvergencí všech běhů, tvořících 23,33% jejich celkového počtu, se tato instance algoritmu DE jeví jako vhodně zvolená na daný problém a z analýzy nejsou patrná žádná slabá místa.



Obrázek 52: Analýza předčasné konvergence algoritmu DE na funkci f_4

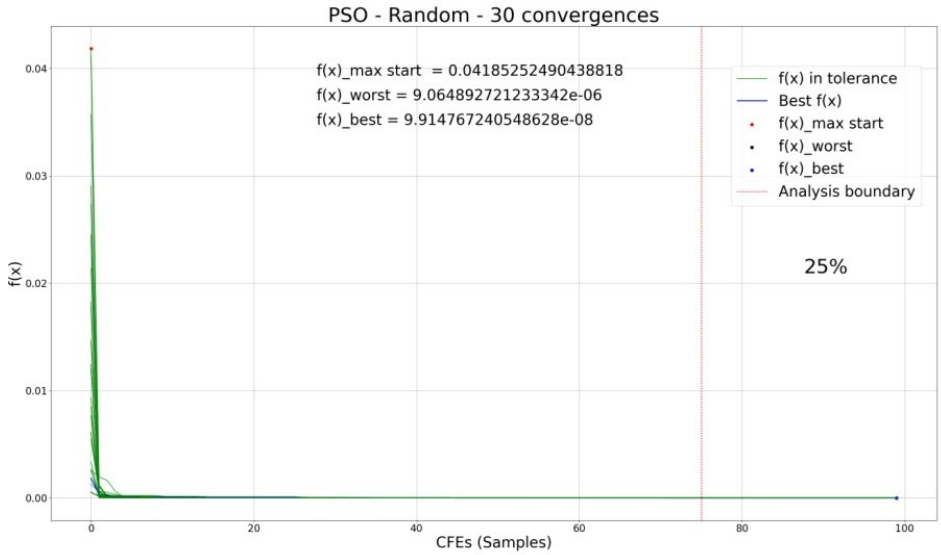
7.3.2 PSO

Algoritmus PSO se v Rank-testu umístil na třetím místě. Clustery vznikají v populaci v každém běhu algoritmu, jejich maximální počet je 3.



Obrázek 53: Charakteristiky algoritmu PSO na funkci f_4

V nejhorším průběhu konvergence s indexem běhu 20 dochází od bodu *CFEs (Samples)* = 3 ke vzniku clusteru, kterému předchází prudký pokles diverzity na hodnotu 7,2976%. Cluster trvá po dobu jednoho vzorku, další se objeví v bodě *CFEs (Samples)* = 16. Další průběh clusteringu má rovnoměrný charakter s výjimkou vzniku dvou clusterů v bodě *CFEs (Samples)* = 51, od bodu *CFEs (Samples)* = 68 pak dochází ke střídání počtu clusterů, avšak diverzita populace již nepřekročí 3%. Nejlepší nalezené řešení pro tento běh má hodnotu $f(x) = 9{,}0648 \times 10^{-6}$ v bodě *CFEs (Samples)* = 96 při hodnotě diverzity 1,91%. Nejlepší průběh konvergence s indexem běhu 22 také ukazuje vznik clusteru na začátku konvergence v bodě *CFEs (Samples)* = 6 doprovázený poklesem diverzity na 6,0706%. Od tohoto bodu dochází ke střídání počtu clusterů, avšak diverzita populace již nepřekročí 6%. Nejlepší nalezené řešení pro všechny běhy má hodnotu $f(x) = 9{,}9147 \times 10^{-8}$ v bodě *CFEs (Samples)* = 71 při hodnotě diverzity 2,458%. Průměrné velikosti průměrných skoků od nejlepších řešení až do konce běhů měly pro nejhorší průběh velikost 9,5373%, pro nejlepší 9,5642%. Pro účely této práce lze díky znalosti globálního optima problému konstatovat, že došlo k aproximaci nejlepšího možného řešení. Oba průběhy vykazují značnou ztrátu diverzity na začátcích konvergencí, u obou však došlo k přiblížení ke globálnímu minimu navzdory výraznému exploitativnímu charakteru. Spolu s výsledky analýzy předčasných konvergencí všech běhů, tvořících 0% jejich celkového počtu, se tato instance algoritmu PSO lze výraznou převahu exploiatce nad explorací označit jako slabé místo i přesto, že došlo k nalezení optima ve všech konvergencích.



Obrázek 54: Analýza předčasné konvergence algoritmu PSO na funkci f_4

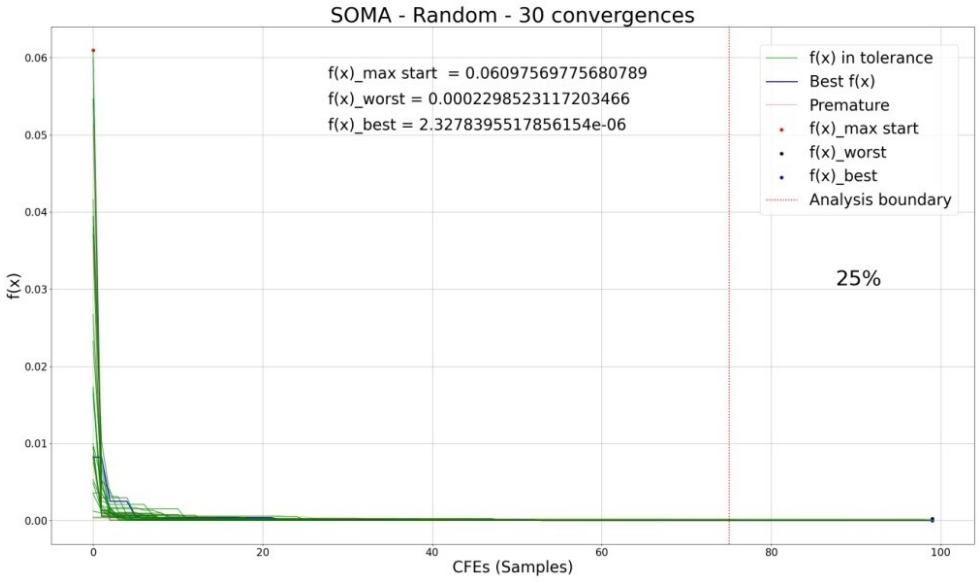
7.3.3 SOMA T3A

Algoritmus SOMA T3A se v Rank-testu umístil na druhém místě. Ke vzniku clusterů nedošlo v žádném z běhů algoritmu.

Obrázek 55: Charakteristiky algoritmu SOMA na funkci f_4

V nejhorším průběhu konvergence s indexem běhu 12 dochází k nárůstu diverzity populace o hodnotu 2,764%, poté diverzita udržuje průměrnou hodnotu 57,4385%, a do konce běhu

nepřesáhne 58%. Nejlepší řešení s hodnotu *f(x)* = 0,0004495757814990675 je pro tento běh nalezeno v bodě *CFEs (Samples)* = 25, při diverzitě populace s hodnotou 57,8781%. V nejlepším průběhu konvergence s indexem běhu 10 je nalezeno nejlepší řešení ze všech běhů s hodnotu *f(x)* = 1,4790x10^{-6} v bodě *CFEs (Samples)* = 86, při diverzitě populace s hodnotou 57,1322%. Průměrné velikosti průměrných skoků od nejlepších řešení až do konce běhů měly pro nejhorší průběh velikost 1,5057%, pro nejlepší 1,3938%. Pro účely této práce lze díky znalosti globálního optima problému konstatovat, že došlo k aproximaci nejlepšího možného řešení. Oba průběhy se vyznačují stabilitou průběhů a průměrných skoků i po nalezení nejlepšího řešení a vyznačují se rovnováhou explorace a exploitace. Spolu s výsledky analýzy předčasných konvergencí všech běhů, tvořících 16,66% jejich celkového počtu, se tato instance algoritmu SOMA T3A jeví jako vhodně zvolená na daný problém a z analýzy nejsou patrná žádná slabá místa.



Obrázek 56: Analýza předčasné konvergence algoritmu SOMA na funkci f_4

7.3.4 Srovnání analýz algoritmů

Instance algoritmu SOMA T3A vykazovala stabilní průběh diverzity populace a v žádném jejím běhu nevznikly clustery, stejně jako u algoritmu DE, jehož diverzita měla také stabilní průběhy. U PSO došlo na začátku algoritmu ke značné ztrátě diverzity, která měla po sestupu stabilní charakter, přičemž docházelo k intenzivní tvorbě clusterů. Uváznutí v lokálním minimu však u algoritmu PSO nenastalo. Všechny algoritmu byly schopny aproximovat globální minimum testovací funkce v obou hodnocených bězích.

ZÁVĚR

Cílem této práce bylo vytvoření nástroje pro identifikaci slabých míst evolučních algoritmů, pomocí analýzy vybraných aspektů vnitřní dynamiky algoritmu. V teoretické části byla popsána základní terminologie oblasti evolučních algoritmů, fundamentální principy přístupů k řešení optimalizačních úloh a východiska těchto přístupů. V práci byl kladen důraz na rovnováhu mezi exploitací a explorací, jako produktu vyrovnaného vlivu deterministických a stochastických komponent evolučních algoritmů. Disbalance těchto vlivů má za následek nekonzistentní výkon a selhávání při hledání globálních optim. Účelem této práce nebylo hodnotit výkon jednotlivých instancí algoritmů, ale pokusit se stanovit, za jakých podmínek dochází k selhání, nebo jaké jsou příčiny vzniku patrných rozdílů výsledků instancí při snaze o nalezení globálního extrému optimalizačního problému. Za tímto účelem byla sestavena specifická sada testovacích funkcí, cílená na lepší pochopení vnitřní dynamiky populace pomocí indikátorů diverzity a dalších charakteristik. Součástí této práce je softwarový nástroj, který byl vytvořen k analýze již existujících výstupů evolučních algoritmů, za účelem extrapolace a přiřazení hodnot těmto jevům, jejichž průběhy jsou měřítky funkčnosti algoritmu. Aplikace evalGeval poskytuje uživateli strukturovaný a vizualizovaný popis dynamiky populace ve formě velikosti pohybu jedinců, skoků, tvorby clusterů v populaci, a průběhu její diverzity. Aplikace umožňuje uživateli centralizovaný pohled na vnitřní procesy algoritmu a usnadňuje tak orientaci v komplexu vlivů na činnost algoritmu. Analýzu je možné provést hromadně na množině dvojic souborů, které jsou výstupy instancí evolučních algoritmů. V rámci ověření funkčnosti vytvořené aplikace bylo provedeno srovnání průběhů konvergencí generických algoritmů DE a PSO s etalonem ve formě pokročilé SOMA, přičemž byly zohledněny ukazatele vývoje populace při pohybu po hyperploše, v kontextu vhodnosti nalezených řešení. V poslední kapitole práce byla provedena interpretace výsledků analýzy prostřednictvím komentovaných grafických a tabulkových výstupů aplikace. Interpretace výsledků ukázala možnost detekce nechtěného chování algoritmu, například výrazná ztráta diverzity populace nebo uváznutí v lokálním extrému při vzniku clusterů. V případě interpretace výstupů analýzy trojice algoritmů na funkci f_4 Random, lze detekovat neočekávané chování ve formě ztráty diverzity populace, a to i v situaci, kdy jsou algoritmem nalézána nejlepší řešení. Kompletní grafické i tabulkové výstupy aplikace pro všechny kombinace algoritmů a testovacích funkcí jsou součástí přílohy této práce, stejně jako zdrojové kódy aplikace evalGeval.

SEZNAM POUŽITÉ LITERATURY

[1] ZELINKA Ivan Zuzana OPLATKOVÁ Miloš ŠEDA Pavel OŠMERA a František VČELAŘ. *Evoluční výpočetní techniky: principy a aplikace*. 1. české vyd. Praha: BEN, 2009, 534 s. ISBN 978-80-7300-218-3.

[2] Yang Yu, Xin Yao, Zhi-Hua Zhou., *Artificial Intelligence., On the approximation ability of evolutionary optimization with application to minimum set cover*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1016/j.artint.2012.01.001

[3] O. C. Ibe., ScienceDirect, 2014., *Fundamentals of Applied Probability and Random Processes*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1016/C2013-0-19171-4

[4] doc. Ing. Roman Šenkeřík, Ph.D.., Rozvoj výzkumně zaměřených studijních programů na FAI (VyStuP FAI)., *Bioinspirované optimalizační metody*, [online]. [cit. 2022-28-2]. Dostupné z: CZ.02.2.69/0.0/0.0/16_018/0002381

[5] Sloss, Andrew N. and Gustafson, Steven., *Neural and Evolutionary Computing., 2019 Evolutionary Algorithms Review*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.48550/arXiv.1906.08870

[6] K. Deb., Burke, E., Kendall, G. (eds) Search Methodologies. Springer, Boston, MA, 2014., *Multi-objective Optimization*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1007/978-1-4614-6940-7_15

[7] D. H. Wolpert and W. G. Macready., *IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67-82*, April 1997., No free lunch theorems for optimization, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/4235.585893

[8] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, S. Prasath., Information 2019, 10, 390, 2019., *Choosing Mutation and Crossover Ratios for Genetic Algorithms-A Review with a New Dynamic Approach*, [online]. [cit. 2022-28-2]. Dostupné z: http://dx.doi.org/10.3390/info10120390

[9] Zelinka, Ivan., *SOMA — Self-Organizing Migrating Algorithm. In: New Optimization Techniques in Engineering. Studies in Fuzziness and Soft Computing, vol 141. Springer, Berlin, Heidelberg*, 2004., [online]. [cit. 2022-28-2]. Dostupné z: http://dx.doi.org/10.1007/978-3-540-39930-8_7

[10] Poli, R., Kennedy, J. & Blackwell, T.., Swarm Intell., *Particle swarm optimization*., [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1007/s11721-007-0002-0

[11] A. Engelbrecht., *2012 IEEE Congress on Evolutionary Computation*, 2012, pp. 1-8., Particle swarm optimization: Velocity initialization, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/CEC.2012.6256112

[12] S. Das, S. S. Mullick, P. N. Guganthan., *Swarm and Evolutionary Computation* 27, 2016., Recent Advances in Differential Evolution – An Updated Survey, [online]. [cit. 2022-28-2]. Dostupné z: http://dx.doi.org/10.1016/j.swevo.2016.01.004

[13] M. Georgioudakis, V. Plevris., *Frontiers in Built Environmnents, Computation Methods in Structural Engineering*, 2020., A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.3389/fbuil.2020.00102

[14] R. Sala and R. Müller., *2020 IEEE Congress on Evolutionary Computation (CEC), 2020., Benchmarking for Metaheuristic Black-Box Optimization: Perspectives and Open Challenges*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/CEC48606.2020.9185724

[15] O. Mersmann, M. Preuss, H.Trautmann., Researchgate, 2010., *Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis*, [online]. [cit. 2022-28-2]. Dostupné z: https://dx.doi.org/10.1007/978-3-642-15844-5_8

[16] M. Hellwig, H.G. Beyer., Swarm Evolutionary Computation, 2018., *Benchmarking evolutionary algorithms for single objective real-valued constrained optimization – A critical review*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1016/j.swevo.2018.10.002

[17] WCCI2022., IEEE WCCI2022., *IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE*, [online]. [cit. 2022-28-2]. Dostupné z: https://wcci2022.org/

[18] P. N. Suganthan., 2022-SO-BO., *P-N-Suganthan/2022-SO-BO*, [online]. [cit. 2022-28-2]. Dostupné z: https://github.com/P-N-Suganthan/2022-SO-BO/blob/main/CEC2022%20TR.pdf

[19] J. Riget, J. S. Vesterstrøm., *EVALife Technical Report no. 2002-02., A Diversity-Guided Particle Swarm Optimizer*, [online]. [cit. 2022-28-2]. Dostupné z:

https://www.researchgate.net/publication/2532296_A_Diversity-Guided_Particle_Swarm_Optimizer_-_the_ARPSO

[20] Z. Zhan, J. Zhang, Y. Li and Y. Shi., *IEEE Transactions on Evolutionary Computation , 2011., Orthogonal Learning Particle Swarm Optimization*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/TEVC.2010.2052054

[21] Q. B. Diep., 2019 IEEE Congress on Evolutionary Computation (CEC), *2019., Self-Organizing Migrating Algorithm Team To Team Adaptive – SOMA T3A*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/CEC.2019.8790202

[22] Diep, Q.B., Zelinka, I., Das, S., Senkerik, R. ., *Communications in Computer and Information Science book series (CCIS,volume 1092)., SOMA T3A for Solving the 100-Digit Challenge*. In: Zamuda, A., Das, S., Suganthan, P., Panigrahi, B. (eds) Swarm, Evolutionary, and Memetic Computing and Fuzzy and Neural Computing. SEMCCO FANCCO 2019 2019. [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1007/978-3-030-37838-7_14

[23] A. Viktorin, R. Senkerik, M. Pluhacek , T. Kadavy., GECCO 2020 Companion - Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. 2020., *Ensemble of strategies and perturbation parameter based SOMA for optimal stabilization of chaotic oscillations*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1145/3377929.3398145

[24] R. Tanabe and A. S. Fukunaga., 2014 IEEE Congress on Evolutionary Computation (CEC), 2014., *Improving the search performance of SHADE using linear population size reduction*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/CEC.2014.6900380

[25] A. Viktorin, R. Senkerik, M. Pluhacek, T. Kadavy, A. Zamuda., 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017., *Distance based parameter adaptation for differential evolution*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/SSCI.2017.8280959

[26] Joaquín Derrac, Salvador García, Sheldon Hui, Ponnuthurai Nagaratnam Suganthan., Information Sciences., *Analyzing convergence performance of evolutionary algorithms: A statistical approach*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1016/j.ins.2014.06.009

[27] M. Pluhacek, A. Viktorin, T. Kadavy, A. Kazikova., 021 IEEE Symposium Series on Computational Intelligence (SSCI), 2021., *On the common population diversity*

measures in metaheuristics and their limitations, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/SSCI50451.2021.9660135

[28] R. W. Morrison, K. De Jong., Artificial Evolution, 5th International Conference, Evolution Artificielle, EA 2001, Le Creusot, France, October 29-31, Lecture Notes in Computer Science 2310:31-41, 2001., *Measurement of Population Diversity*, [online]. [cit. 2022-28-2]. Dostupné z: http://dx.doi.org/10.1007/3-540-46033-0_3

[29] M. Bhattacharya., arXiv, 2014., *Diversity Handling In Evolutionary Landscape*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.48550/arxiv.1411.4148

[30] ZHENYU, Yang, YUEJIN, Tan, RENJIE, He., Mathematical Problems in Engineering, Hindawi Publishing Corporation, 2014., *Diversity Controlling Genetic Algorithm for Order Acceptance and Scheduling Problem*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1155/2014/367152

[31] A. Viktorin, R. Senkerik, M. Pluhacek , T. Kadavy., MENDEL Soft Computing Journal, 2018., Clustering *Analysis of the Population in Db_SHADE Algorithm*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.13164/mendel.2018.1.009

[32] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas and A. C. Ponce Leon F. de Carvalho., IEEE Transactions on Systems, Man, and Cybernetics, Part C*., A Survey of Evolutionary Algorithms for Clustering*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/TSMCC.2008.2007252

[33] A. Viktorin, R. Senkerik, M. Pluhacek , A. Zamuda., 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016., *Steady success clusters in Differential Evolution*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1109/SSCI.2016.7850252

[34] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. Blondel, M., Prettenhofer, P., Weiss R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.., Journal of Machine Learning Research, 2011., Scikit-learn*: Machine Learning in Python*, [online]. [cit. 2022-28-2]. Dostupné z: https://scikit-learn.org/stable/install.html

[35] Scikit learn 1.0.2., Modules., *2.3.7 DBSCAN*, [online]. [cit. 2022-28-2]. Dostupné z: https://scikit-learn.org/stable/modules/clustering.html#dbscan

[36] Zelinka, Ivan., Swarm Evolutionary Computation, 2015., *A Survey on Evolutionary Algorithms Dynamics and its Complexity - Mutual Relations, Past, Present and*

Future, [online]. [cit. 2022-28-2]. Dostupné z: http://dx.doi.org/10.1016/j.swevo.2015.06.002

[37] A, Gaspar-Cunha, J. A. Covas., Comput Optim Appl 39, 75–96 (2008)., *Robustness in multi-objective optimization using evolutionary algorithms*, [online]. [cit. 2022-28-2]. Dostupné z: https://doi.org/10.1007/s10589-007-9053-9

[38] A. Ahari, M. R. Saadatmand, M. S. Panahi, A. A. Atai., Applied Mathematics and Computation 215(9):3222-3229, 2010., *On the limitations of classical benchmark functions for evaluating robustness of evolutionary algorithms*, [online]. [cit. 2022-28-2]. Dostupné z: http://dx.doi.org/10.1016/j.amc.2009.10.009

[39] Y. Shafranovich., SolidMatrix Technologies, Inc., 2005., *rfc4180*, [online]. [cit. 2022-28-2]. Dostupné z: https://datatracker.ietf.org/doc/html/draft-shafranovich-mime-csv-05

[40] H. Pohlheim., GEATbx.com., *Examples of Objective Functions*, [online]. [cit. 2022-28-2]. Dostupné z: http://www.geatbx.com/download/GEATbx_ObjFunExpl_v38.pdf

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

EA	Evoluční algoritmus
GA	Genetický algoritmus
ES	Význam třetí zkratky
DE	Diferenciální evoluce
SOMA	Self-Organizing Migrating Algorithm
PSO	Particle Swarm Optimization
CF	Cost function
CFE	Cost Function Evaluation
NFLT	No Free Lunch Theorem
PRT	Pertrubance
IEEE	Institute of Electrical and Electronic Engineers
CEC	Congress on Evolutionary Computation
WCCI	World Congress on Computational Intelligence
MaxFEs	Maximum Function Evaluations
SR	Sum-Rank
SaDE	Self-Adaptive Differential Evolution
ARPSO	Attraction-Repulsion Particle Swarm Optimization
OLPSO	Orthogonal Learning Particle Swarm Optimization
SOMA T3A	Self-Organizing Migrating Algorithm Team To Team Adaptive
ESP-SOMA	Ensemble Strategies and Pertrubation parameter Self-Organizing Migrating Algorithm
L-SHADE	Success-History based Adaptive Differential Evolution with Linear Population Size Reduction
Db-SHADE	Distance based Success-History based Adaptive Differential Evolution
DAP	Distance to Average Point
APD	Averaged point Diversity

DWD	Dimension-Wise Diversity
DGEA	Diversity Guided Evolutionary Algorithm
CFEs	Cost Function Evaluations
CSV	Comma Separated Values
CRLF	Carriage Return Line Feed

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ELEKTRONICKÝCH PŘÍLOH

Příloha P I: Adresář projektu aplikace evalGeval v jazyce Ptyhon

Příloha P II: Adresář se záznamy vzorků populací a konvergencí algoritmů DE, PSO a SOMA T3A

Příloha P III: Adresář s výstupy analýzy aplikace evalGeval