Analýza vlivu vah adaptivního procesu diferenciální evoluce

Bc. Martin Cypris

Diplomová práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Martin Cypris**
Osobní číslo:	**A22589**
Studijní program:	**N0613A140022 Informační technologie**
Specializace:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Analýza vlivu vah adaptivního procesu diferenciální evoluce**
Téma práce anglicky:	**Analysis of the Weighted Adaptive Process in Differential Evolution**

Zásady pro vypracování

1. Seznamte se s adaptivními variantami algoritmu diferenciální evoluce a vytvořte jejich přehled.
2. Identifikujte varianty, které využívají v adaptivním procesu vážený průměr.
3. Definujte testovací scénář, na kterém budete testovat vliv váženého průměru na kvalitu optimalizace.
4. Otestujte vybrané varianty adaptivní diferenciální evoluce a statisticky vyhodnoťte výsledky.
5. Proveďte diskuzi výsledků a vytvořte doporučení pro volbu váhového modelu.

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

Seznam doporučené literatury:

1. PRICE, Kenneth V.; STORN, Rainer M. a LAMPINEN, Jouni A. *Differential evolution: a practical approach to global optimization*. Natural computing series. Berlin: Springer, 2005. ISBN 9783540209508. Dostupné také z: https://proxy.k.utb.cz/login?url=https://doi.org/10.1007/3-540-31306-0.
2. TANABE, Ryoji a FUKUNAGA, Alex. Success-history based parameter adaptation for Differential Evolution. Online. In: *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, s. 71-78. ISBN 978-1-4799-0454-9. Dostupné z: https://doi.org/10.1109/CEC.2013.6557555.
3. VIKTORIN, Adam; SENKERIK, Roman; PLUHACEK, Michal; KADAVY, Tomas a ZAMUDA, Ales. Distance based parameter adaptation for Success-History based Differential Evolution. Online. *Swarm and Evolutionary Computation*. 2019, roč. 2019, č. 50, article 100462. ISSN 2210-6502. Dostupné z: https://doi.org/10.1016/j.swevo.2018.10.013.
4. BREST, Janez; MAUCEC, Mirjam Sepesy a BOSKOVIC, Borko. Single objective real-parameter optimization: Algorithm jSO. Online. In: 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2017, s. 1311-1318. ISBN 978-1-5090--4601-0. Dostupné z: https://doi.org/10.1109/CEC.2017.7969456.
5. JANÁČEK, Julius. Statistika jednoduše: *Průvodce světem statistiky*. Grada, 2022. ISBN 978-80-271-1738-3. Dostupné také z: https://www.bookport.cz/AccountSaml/SignIn/?idp=https://shibboleth.utb.cz/idp/shibboleth&returnUrl=/kniha/statistika--jednoduse-11215/.

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

Datum zadání diplomové práce: **5. listopadu 2023**
Termín odevzdání diplomové práce: **13. května 2024**

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

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

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

Ve Zlíně dne 5. ledna 2024

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 …….

podpis diplomanta

ABSTRAKT

Tato práce se zaměřuje na různou aplikaci adaptivních vah v diferenciální evoluci s cílem zjistit, jaký vliv to bude mít na hledání globálního extrému. Práce zkoumá různé modely adaptivních vah, které jsou navrženy tak, aby reagovaly na výkon algoritmu během předchozích iterací. Cílem je posoudit, jak tyto modely ovlivňují konvergenční rychlost algoritmu na škále benchmarkových testů. Výsledky těchto experimentů mají ukázat, zda je možné prostřednictvím adaptivních vah dosáhnout efektivnějších výsledků v rámci standardních optimalizačních scénářů.

Klíčová slova: Defirenciální evoluce, SHADE, optimalizace

ABSTRACT

This thesis focuses on the application of adaptive weights in differential evolution to determine their impact on the search for a global extremum. The study examines various models of adaptive weights, designed to respond to the algorithm's performance during previous iterations. The goal is to assess how these models affect the algorithm's convergence speed across a range of benchmark tests. The results of these experiments are intended to show whether adaptive weights can achieve more efficient outcomes within standard optimization scenarios.

Keywords: Differential evolution, SHADE, optimalization

Tímto bych chtěl poděkovat vedoucímu práce Ing. Adamovi Viktorinovi, Ph.D. za jeho ochotu a cenné rady při tvorbě této práce. Zároveň děkuji všem učitelům, kteří mi poskytli vědomosti bez, kterých by realizace této práce nebyla možná. Taktéž chci poděkovat svým přátelům a rodičům za jejich podporu a trpělivost.

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

OBSAH

ÚVOD

V posledních desetiletích se staly evoluční algoritmy klíčovým nástrojem v oblasti strojového učení a optimalizace. Tyto algoritmy, inspirované biologickými evolučními procesy, se osvědčily v mnoha aplikacích, kde tradiční metody selhávají z důvodu komplexnosti nebo dimenzionality problému. Jedním z takových algoritmů je diferenciální evoluce (DE), která je známá pro svou schopnost efektivně nalézat globální optima v nelineárních, nekonvexních, a multimodálních prostředích. Diferenciální evoluce, která byla poprvé představena Stornem a Pricem v roce 1995, se rychle rozšířila díky své jednoduchosti, robustnosti a vysoké efektivitě.

Nicméně, úspěch diferenciální evoluce závisí na správném nastavení jejích kontrolních parametrů, jako je velikost populace, mutační konstanta a pravděpodobnost křížení. Tyto parametry mají zásadní vliv na konvergenční rychlost a schopnost algoritmu nalézt skutečné globální optimum. Vzhledem k tomu, že optimální hodnoty těchto parametrů se mohou výrazně lišit v závislosti na specifických charakteristikách problému, vědci a inženýři často čelí náročnému úkolu jejich kalibrace.

Tento problém vedl k rozvoji adaptivních a samoadaptivních strategií v rámci diferenciální evoluce, které umožňují algoritmu dynamicky upravovat své parametry v průběhu evolučního procesu. Tyto metody mají potenciál výrazně zlepšit výkonnost algoritmu tím, že umožní lepší prozkoumání a využití vyhledávacího prostoru, a současně zredukují potřebu ručního nastavování parametrů.

Práce se zaměřuje na analýzu vlivu adaptivních vah v diferenciální evoluci, což je přístup, který se snaží zlepšit schopnost algoritmu adaptovat se na měnící se prostředí optimalizovaného problému. Specificky se tato práce zabývá vývojem a testováním různých modelů adaptivních vah, které mohou modifikovat distribuci pravděpodobnosti použitých operátorů (mutace, křížení) v závislosti na historickém výkonu populace. Hlavním cílem je zjistit, zda tyto adaptivní strategie mohou zlepšit efektivitu diferenciální evoluce v různých typech optimalizačních problémů.

Diplomová práce je rozdělena do několika klíčových sekcí. Úvodní kapitola poskytuje přehled diferenciální evoluce a zdůrazňuje význam adaptace parametrů. Teoretická část práce se věnuje podrobnému popisu mechanismů a strategií diferenciální evoluce, včetně analýzy různých metod pro adaptaci jejich parametrů. Následuje praktická část, která představuje výsledky experimentů. Zde jsou prezentovány a diskutovány výsledky

experimentů prováděných s různými modely adaptivních vah na standardizovaných benchmarkových funkcích. Tato práce zakončuje zhodnocením zjištěných výsledků a diskuzí o tom, jak mohou adaptivní strategie přispět k efektivitě diferenciální evoluce.

I. TEORETICKÁ ČÁST

1 DIFERENCIÁLNÍ EVOLUCE

Diferenciální evoluce (DE) byla poprvé představena v roce 1995 jako součást doktorské práce R. Storna pod vedením K. Price. Jedná se o jednoduchý optimalizační nástroj, inspirovaný algoritmem genetického žíhání, který byl publikován K. Pricem o rok dříve. Motivací pro vývoj diferenciální evoluce byla nedostatečná efektivita genetického žíhání při řešení optimalizačních problémů. Původní algoritmus byl adaptován tak, aby namísto používání binárních řetězců a logických operací využíval reálná čísla a aritmetické operace. [1]

Ačkoli je diferenciální evoluce na první pohled jednoduchým nástrojem, dokáže efektivně konkurovat složitějším optimalizačním algoritmům. Tento úspěch je z velké části závislý na správném výběru a nastavení jejích klíčových parametrů. Diferenciální evoluce se zaměřuje na tři hlavní parametry: velikost populace, mutační konstantu F a práh křížení CR, přičemž každý z nich hraje zásadní roli v účinnosti algoritmu. Optimalizace těchto parametrů se může lišit v závislosti na specifických potřebách daného optimalizačního problému, což vyžaduje jejich pečlivé přizpůsobení pro dosažení optimálních výsledků. [2]

Manuální kalibrace parametrů může být zejména u komplexnějších problémů velmi časově náročná. V reakci na tuto výzvu se začaly vyvíjet metody, které umožňují parametrům adaptovat se automaticky. Tyto samoadaptivní algoritmy dynamicky upravují řídící parametry v průběhu procesu optimalizace, aby zvýšily svou efektivitu při hledání globálního optima. Tato inovace znamená významný pokrok ve způsobu, jakým můžeme přistupovat k složitým optimalizačním výzvám, a otevírá cestu k efektivnějším řešením. [2]

1.1 Parametry diferenciální evoluce

Základní diferenciální evoluce se řídí čtyřmi klíčovými parametry, které mají zásadní vliv na efektivitu procesu hledání globálního extrému. Tyto základní parametry jsou: [3]

- NP (Velikost populace): Parametr NP určuje celkový počet jedinců v populaci každé generace. Optimální velikost populace je klíčová; příliš malá populace může způsobit, že algoritmu bude chybět dostatek kandidátů pro efektivní křížení, zatímco příliš velká populace může proces zpomalit kvůli delší době potřebné k vygenerování nové generace.
- F (Mutační konstanta): Tato konstanta je zásadní pro určení intenzity mutací, tedy jak moc se nově vytvořené váhové diferenciální vektory liší od původních. Nízká

hodnota může vést k pomalému hledání a riziku uvíznutí v lokálním maximu, zatímco příliš vysoká hodnota může způsobit přeskočení globálního optima. S narůstající velikostí populace by měla být hodnota F upravena směrem dolů.

- CR (Práh křížení): Určuje pravděpodobnost, že se geny z mutačního vektoru projeví v novém jedinci. Jeho hodnoty se pohybují v rozmezí 0 až 1. Nulová hodnota efektivně zastavuje evoluci, protože nedochází k žádným změnám mezi generacemi, zatímco hodnota 1 znamená, že algoritmus ignoruje selektivní výhody a spoléhá se na náhodné prohledávání.
- G_{Max} (Maximální počet generací): Tento parametr nastavuje limit pro počet generací, po kterých bude algoritmus ukončen. Nastavení příliš nízkého limitu může předejít nalezení globálního extrému, protože algoritmus nemá dostatek času na prozkoumání prostoru řešení.

Každý z těchto parametrů hraje klíčovou roli v úspěšnosti diferenciální evoluce a jejich správné nastavení je nezbytné pro dosažení optimálních výsledků. V následující tabulce jsou uvedeny doporučené hodnoty parametrů pro klasickou DE. [3]

Tabulka 1 Doporučené hodnoty parametrů [3]

Parametr	**Hodnota**
NP	Problémy s nižším počtem dimenzí – 10D Problémy s vyšším počtem dimenzí – 100D
F	0,5 – 0,9 (maximálně 2,0)
CR	0,7 – 0,9
G_{Max}	Pro jednodušší problémy – 10^2 Pro složitější problémy – 10^3 - 10^6

Klíčovým parametrem, který ovlivňuje proces diferenciální evoluce, je také počet dimenzí problému, označovaný jako D. Tento parametr je přímo určen charakterem řešeného problému a jeho hodnota může být upravena uživatelem pouze prostřednictvím redefinice samotného problému. Tedy, velikost dimenze je pevně dána a reflektuje komplexnost a povahu optimalizační úlohy. [3]

Dalším faktorem, který má značný dopad na výkonnost diferenciální evoluce, je definice účelové funkce a omezení. Účelová funkce, pokud je nesprávně specifikována, může výrazně zpomalit evoluční proces nebo dokonce znemožnit nalezení řešení. Je zásadní, aby byla účelová funkce správně definována s ohledem na specifika řešeného problému. Kromě toho je nezbytné určit oblast hledání pro diferenciální evoluci, což se obvykle děje výběrem reprezentativního jedince, který určuje prostor, v němž algoritmus bude hledat globální extrém. Nedostatečná specifikace této oblasti může vést k tomu, že jednotlivci budou prozkoumávat nevhodné oblasti nebo dokonce mohou konvergovat k nekonečnu, což by mohlo celý proces evoluce výrazně narušit. [4]

1.2 Základní princip diferenciální evoluce

Diferenciální evoluce se odlišuje od ostatních evolučních algoritmů svým unikátním přístupem k procesu evoluce. Zatímco většina evolučních algoritmů v jedné generaci postupuje podle schématu: nejprve selekce, poté křížení a nakonec mutace, diferenciální evoluce tento postup obrací. Tedy, na začátku každé generace stojí mutace, po které následuje křížení a celý cyklus je zakončen selekcí. Tento netradiční přístup znamená, že diferenciální evoluce přistupuje k procesu evoluce zcela odlišně, což může přinášet výhody v efektivitě a schopnosti nalézat optimální řešení v různorodých optimalizačních problémech. [3]

Každý algoritmus diferenciální evoluce zahajuje svou činnost inicializací populace, přičemž počet jejích jedinců je určen vstupním parametrem NP. Na základě tohoto parametru se náhodně generuje odpovídající množství jedinců, které tvoří výchozí populaci. Po inicializaci následuje hlavní cyklus algoritmu, jehož jádrem jsou procesy mutace, křížení a selekce. Tyto operace jsou klíčové pro postupnou evoluci populace a jejich specifický princip se liší v závislosti na zvolené variantě algoritmu diferenciální evoluce. [3]

Hlavní cyklus je obvykle ukončen, když je splněna jedna ze stanovených ukončovacích podmínek. Nejčastěji se jedná o dosažení maximálního počtu generací, což je limit, který zamezí nekonečnému běhu algoritmu. Existuje však také možnost, že algoritmus končí, pokud se nejlepší jedinec v populaci přestane zlepšovat, což naznačuje, že další pokračování v iteracích pravděpodobně nepřinese výraznější zlepšení výsledků. [3]

Algoritmus 1: Diferenciální evoluce

inicializace populace $P = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$, $\boldsymbol{x}_i \in \Omega$

while *není dosažena ukončovací podmínka* **do**

  for $i = 1$ *to* N **do**

    generování potomka $\boldsymbol{y}_i$ reprodukcí

    if $f(\boldsymbol{y}_i) \leq f(\boldsymbol{x}_i)$ **then**

      vložení $\boldsymbol{y}_i$ do nové populace Q

    else

      vložení $\boldsymbol{x}_i$ do nové populace Q

    end

  end

  $P := Q$

end

Obrázek 1 Pseudokód základní diferenciální evoluce

1.2.1 Mutace

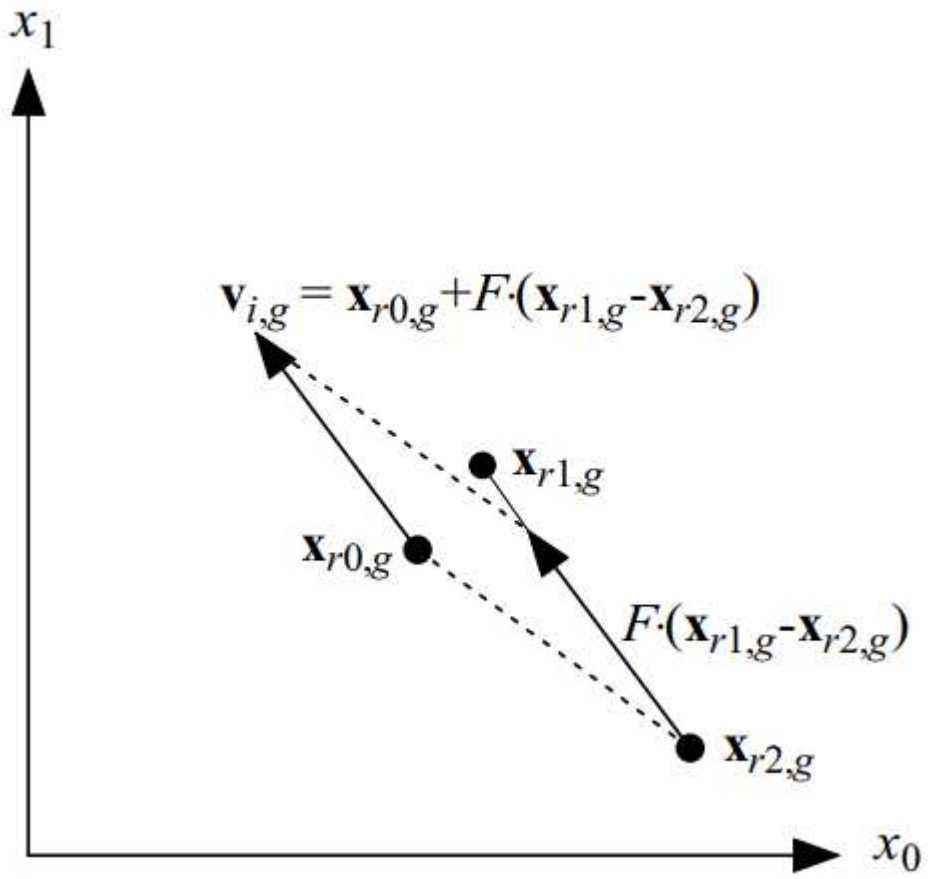
Po úvodní inicializaci populace se v diferenciální evoluci (DE) provádí mutace a křížení, aby byla vytvořena nová populace pro následující generaci. Mutace je klíčovým procesem, během kterého dochází ke změnám vlastností vybraných jedinců. Tento krok je zásadní pro udržení a zvyšování diverzity v populaci mezi generacemi. Konkrétní mutace jedince je realizována využitím mutačního vektoru, jehož výpočet se odvíjí od zvolené mutační strategie. Nejběžnější strategií používanou v DE je DE/rand/1. [1]

V rámci této strategie jsou z populace náhodně vybráni tři jedinci x_{r0}, x_{r1}, x_{r2}, kteří se navzájem liší a jsou odlišní i od jedince podléhajícího mutaci x_i. Mutační vektor v_i se vypočítá podle následujícího vztahu. [1]

$$v_{i,g} = x_{r0,g} + F \cdot (x_{r1,g} - x_{r2,g}) \tag{1}$$

Index ve vzorci g označuje generaci, ve které probíhá mutace. Tento vztah ukazuje, že mutační vektor v_i vzniká jako rozdíl mezi vektory dvou náhodně vybraných jedinců, což se nazývá diferenční vektor. Tento diferenciální vektor je pak zvětšen o faktor mutační

konstanty F a přičten k vektoru třetího jedince. Celý proces mutace ilustruje přiložený obrázek. [1]



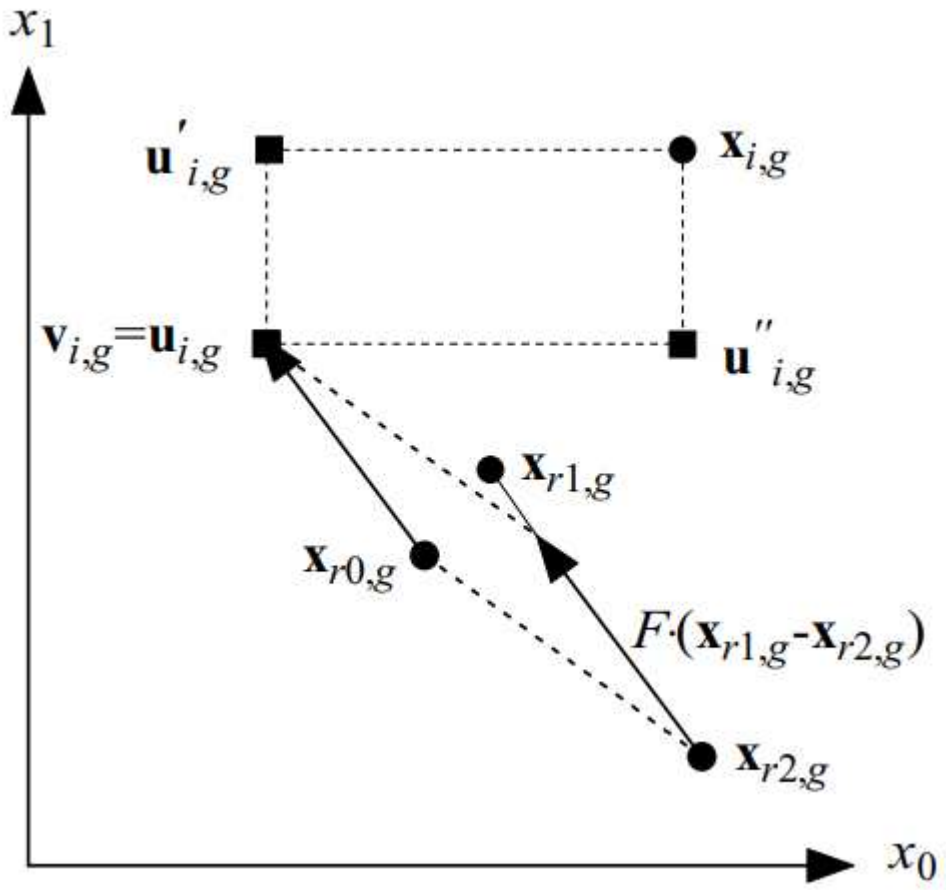
Obrázek 2 Tvorba mutačního vektoru [1]

1.2.2 Křížení

Křížení je dalším klíčovým krokem v procesu evoluce diferenciální evoluce, vedoucím k vytvoření nového jedince, který se odvozuje z kombinace vektoru rodiče a mutačního vektoru. Tento nový jedinec je často označován jako zkušební vektor. Existují různé metody křížení, přičemž binominální a exponenciální křížení jsou dvě z nejčastěji používaných. [1]

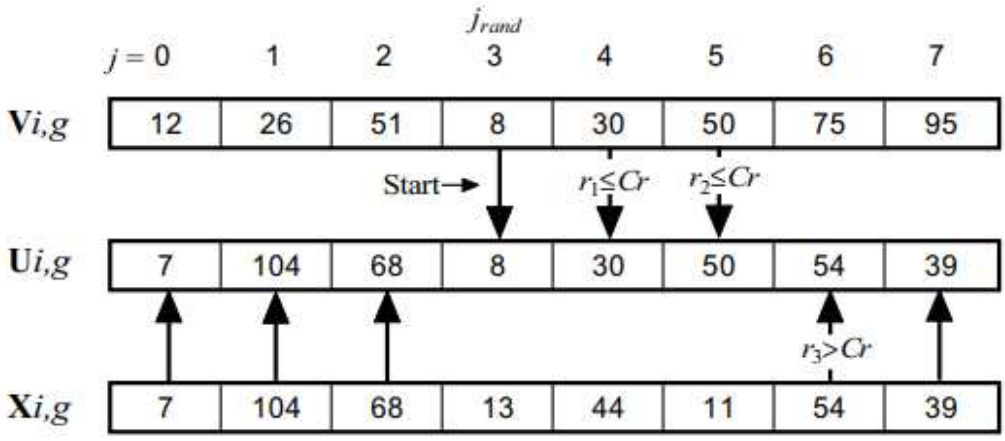
Binominální křížení tvoří potomka kombinací hodnot z rodičovského vektoru a mutačního vektoru. Zde hraje klíčovou roli práh křížení CR, který určuje poměr, v jakém jsou hodnoty přejímány z obou vektorů. Proces křížení využívá náhodný generátor čísel, který pro každou hodnotu generuje náhodné číslo *randj*(0, 1). Pokud je vygenerované číslo větší než CR, hodnota je převzata z rodičovského vektoru. V opačném případě, pokud je číslo menší nebo rovno CR, hodnota je převzata z mutačního vektoru. K zajištění diverzity a unikátnosti potomka vždy zahrnuje jeden zkušební parametr s náhodně vybraným indexem přímo z mutačního vektoru, což zabraňuje vzniku identické kopie rodiče. [1]

V následujícím obrázku je znázorněn proces a potenciální výsledky binominálního křížení.



Obrázek 3 Binominální křížení [1]

Exponenciální křížení je proces, při kterém je na začátku náhodně vybrána pozice, od které začne křížení. Hodnota na této pozici je automaticky převzata do potomka z mutačního vektoru, tedy z mutanta. Pro všechny další hodnoty v sekvenci se provádí náhodné generování čísel, která jsou porovnávána s parametrem CR. Kopírování hodnot z mutanta do potomka pokračuje tak dlouho, dokud generované náhodné číslo zůstává menší nebo rovno CR. Jakmile náhodně generované číslo přesáhne hodnotu CR, zbytek hodnot je do potomka přenesen z rodičovského vektoru. Tento postup zajišťuje, že vzniklý potomek obsahuje hodnoty z obou rodičů, ale zároveň zůstává unikátní díky pseudonáhodnému výběru bodu začátku křížení a následnému převodu hodnot. Tento proces vytváří potomky s kombinací genetické informace z obou rodičů, přičemž díky svému postupu může vést k různorodější populaci než binominální křížení. [1]



Obrázek 4 Exponenciální křížení [1]

1.2.3 Selekce

V procesu evoluce, který využívá diferenciální evoluce, následuje po kroku křížení fáze selekce. Selekce je proces, v němž dochází k výběru jedinců z aktuální populace pro začlenění do populace nové generace. Základním principem selekce je srovnání rodičovského jedince s jeho potomkem. Toto porovnání se opírá o výsledky účelové funkce, což je kritérium, podle kterého se hodnotí kvalita jedinců. [1]

V každé dvojici rodič-potomek algoritmus určí, který z nich má lepší výsledek účelové funkce. Jedinec s lepší hodnotou je poté zařazen do následující generace. Tento postup zajišťuje, že kvalita populace se v průběhu evolučního procesu buď udržuje na stejné úrovni nebo se zlepšuje. Díky selekci je také minimalizována šance na zhoršení celkové kvality populace v další generaci. [1]

Po provedení selekce je aktuální cyklus neboli generace, ukončen a algoritmus pokračuje dalším cyklem. Tento cyklus mutace, křížení a selekce se opakuje až do momentu, kdy je splněna některá z ukončovacích podmínek. Nejčastěji se jedná o dosažení maximálního počtu generací nebo nalezení globálního optima účelové funkce. [1]

Tento přístup zajišťuje, že algoritmus diferenciální evoluce se snaží neustále hledat nejlepší možná řešení, zatímco zachovává genetickou rozmanitost v populaci a zároveň předchází stagnaci vývoje. [1]

1.3 Ukončovací podmínky

Ukončovací podmínky pro DE mohou být variabilní, závislé na konkrétních cílech a znalostech o řešeném problému. Pokud je hodnota globálního extrému známá předem, může

být algoritmus konfigurován tak, aby terminoval v okamžiku, kdy hodnota účelové funkce nejlepšího jedince dosáhne hodnoty dostatečně blízké této známé optimální hodnotě. [1]

V situacích, kdy hodnota globálního extrému není předem známá, se jako nejběžnější ukončovací podmínka využívá dosažení maximálního počtu generací. Tento počet by měl být nastaven s ohledem na kompromis mezi dostatečným časem pro nalezení optimálního řešení a nutností zabránit příliš dlouhému běhu algoritmu. Nesprávně nastavený maximální počet generací může mít za následek neefektivní využití algoritmu, ať už nedostatečným časem pro nalezení globálního extrému nebo neúměrně dlouhým během bez zlepšení. [1]

Další možnou ukončovací podmínkou je omezení na dobu běhu algoritmu. Tato podmínka se uplatňuje spíše výjimečně, když jsou časové omezení kritické. Limitovaný čas běhu může přinést výzvy v podobě nedostatečného prostoru pro nalezení optimálního řešení. [1]

Ukončení algoritmu může být také podmíněno dosažením specifické statistické hodnoty v populaci, například rozdílu mezi hodnotami účelové funkce nejlepšího a nejhoršího jedince. Tato podmínka vyžaduje opatrnou kalibraci, aby nedošlo k předčasnému ukončení běhu algoritmu v důsledku stagnace v lokálním extrému. [1]

1.4 Strategie diferenciální evoluce

Diferenciální evoluce (DE) disponuje širokou škálou strategií, které určují průběh mutace a křížení. Tyto strategie mají zásadní význam pro fungování algoritmu DE, jelikož přímo ovlivňují jeho schopnost optimalizace. Přestože základní princip diferenciální evoluce zůstává neměnný, konkrétní metody mutace a křížení se liší v závislosti na zvolené strategii. [3]

Strategie DE jsou definovány pomocí zápisu DE/m/n/c, kde každý prvek má svůj specifický význam: [3]

- m označuje typ mutace, tj. metodu, podle které je vybírán aktivní jedinec pro mutační proces.
- n udává počet diferenciálních vektorů, což odpovídá počtu rozdílů mezi náhodně vybranými jedinci v populaci.
- c specifikuje typ křížení použitý ve strategii, například binominální nebo exponenciální

Výběr vhodné strategie DE má přímý dopad na efektivitu algoritmu v různých optimalizačních úlohách. Různé strategie nabízejí rozličné přístupy k mutaci a křížení, což umožňuje algoritmu lépe se adaptovat na specifika řešeného problému. [3]

V tabulce jsou vybrány některé typy strategií DE:

Tabulka 2 Strategie DE [3]

Značení	Výpočet mutačního vektoru	
DE/rand/1/bin	$v_i = x_{r0} + F \cdot (x_{r1} - x_{r2})$	(2)
DE/best/1/bin	$v_i = x_{best} + F \cdot (x_{r1} - x_{r2})$	(3)
DE/rand/1/exp	$v_i = x_0 + F \cdot (x_{r1} - x_{r2})$	(4)
DE/best/1/exp	$v_i = x_{best} + F \cdot (x_{r1} - x_{r2})$	(5)
DE/rand/2/bin	$v_i = x_{r0} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$	(6)
DE/best/2/bin	$v_i = x_{best} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$	(7)
DE/rand/2/exp	$v_i = x_{r0} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$	(8)
DE/rand/2/exp	$v_i = x_{best} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$	(9)
DE/rand-to-best/1/bin	$v_i = x_i + \lambda \cdot (x_{best} - x_i) + F \cdot (x_{r1} - x_{r2})$	(10)
DE/rand-to-best/1/exp	$v_i = x_i + \lambda \cdot (x_{best} - x_i) + F \cdot (x_{r1} - x_{r2})$	(11)

1.5 Typy diferenciální evoluce

Efektivita diferenciální evoluce (DE) výrazně závisí na pečlivém výběru vstupních parametrů a na adaptaci vhodné strategie specifické pro daný problém. Optimalizace těchto nastavení se liší v závislosti na charakteru řešené úlohy, což vyžaduje individuální přístup k nastavování parametrů F, CR a volbě strategie pro každý nový problém. Uživatelé mají možnost volby z široké škály strategií a mohou libovolně upravovat parametry F a CR v rámci určených rozsahů, což poskytuje obrovskou variabilitu v konfiguraci algoritmu. Avšak s takovým množstvím možností nastavení se proces kalibrace stává časově náročným. [5]

Právě kvůli této výzvě byly vyvinuty různé typy adaptivní diferenciální evoluce. Adaptivní verze DE jsou schopné dynamicky upravovat parametry F a CR, strategii, a dokonce i velikost populace během svého běhu. Díky schopnosti "učení" z výkonu zvolených nastavení v reálném čase se tyto adaptivní algoritmy stávají postupně efektivnějšími. Tato vlastnost adaptivní DE značně usnadňuje proces optimalizace tím, že redukuje čas a úsilí potřebné pro manuální kalibraci algoritmu, čímž umožňuje rychlejší a přesnější nalezení optimálních řešení pro různorodé problémy. [5]

1.5.1 jDE

V algoritmu jDE, což je adaptivní varianta diferenciální evoluce, jsou pro každého jedince individuálně definovány parametry F a CR, které z počátku nabývají náhodně generovaných hodnot. Tento inovativní přístup umožňuje, že během procesu mutace mají tyto parametry možnost dynamického přizpůsobení dle určené pravděpodobnosti. Následná selekce poté hodnotí, zda adaptace těchto parametrů přinesla lepší výsledky než původní nastavení rodičovského jedince. [5]

Pokud adaptované hodnoty F a CR vykazují lepší výkon – měřeno hodnotou účelové funkce – jsou tyto hodnoty přijaty a aplikovány na jedince v další generaci. V případě, že nové hodnoty nevedou ke zlepšení, je preferováno zachování původních parametrů rodiče pro nadcházející generaci. Tento adaptivní mechanismus nejenže zvyšuje efektivitu algoritmu, ale také umožňuje flexibilní reakci na různorodé optimalizační výzvy bez potřeby předběžného ručního ladění parametrů. [5]

Algoritmus jDE tak reprezentuje pokročilý nástroj v oblasti optimalizačních algoritmů, jehož hlavní předností je schopnost automatické adaptace a učení se z průběhu evoluce. Tato vlastnost významně snižuje čas a úsilí potřebné pro přípravu algoritmu k efektivní práci na širokém spektru problémů. Díky adaptivním vlastnostem jDE mohou být výsledky optimalizace výrazně lepší a rychleji dosažitelné, což otevírá cestu k řešení i těch nejsložitějších optimalizačních úkolů. [5]

Tato adaptace a učení se v průběhu procesu optimalizace jsou klíčové pro dosažení optimálních výsledků v neustále se měnícím a výzvami nabitém prostředí optimalizačních problémů, čímž algoritmus jDE stojí v popředí současných trendů v oblasti evolučních výpočetních technik. [5]

1.5.2 EPSDE

Algoritmus EPSDE (Enhanced Parameter Selection Differential Evolution) představuje pokročilou formu diferenciální evoluce, která je navržena tak, aby umožnila efektivnější a cílenější prohledávání prostoru řešení optimalizačních úloh. Tento algoritmus se odlišuje od tradiční diferenciální evoluce zejména svým přístupem k selekci a adaptaci parametrů F a CR, jakož i k volbě strategie mutace. [5]

Na začátku je v EPSDE uživatelem definována množina potenciálních hodnot pro parametry F a CR, společně s výběrem možných strategií mutace, které mohou být v průběhu

evolučního procesu využity. Mezi oblíbené strategie patří například DE/Best/2/bin, DE/rand/1/bin a DE/current-to-rand/1/bin, které se liší v mechanismu výběru jedinců pro generování nových potomků a aplikaci mutací. [5]

V rámci inicializační fáze je každému jedinci přidělena jedna z předem definovaných strategií, stejně jako náhodně zvolené hodnoty pro F a CR. Tato jedinečná kombinace strategie a parametrů je následně přenášena z rodiče na potomka. Během selekce, pokud se ukáže, že potomek (s jeho specifickými hodnotami parametrů a zvolenou strategií) předčí svého rodiče v hodnotě účelové funkce, je potomek vybrán pro začlenění do nové generace, přičemž si zachová své parametry a strategii pro další cyklus. [5]

Pokud je však během selekce rozhodnuto o zachování rodiče pro další generaci, dochází v dalším cyklu k náhodnému přeřazení strategie a novým hodnotám pro F a CR. Tímto způsobem algoritmus EPSDE podporuje diverzitu v populaci a zároveň umožňuje adaptaci na nejúčinnější kombinace strategií a parametrů v závislosti na aktuálních podmínkách a potřebách optimalizačního procesu. [5]

Je důležité poznamenat, že EPSDE byl specificky navržen pro zlepšení schopnosti algoritmu DE přizpůsobit se různým typům optimalizačních problémů, zlepšení konvergence a zvýšení pravděpodobnosti nalezení globálního optima. Tato adaptivní schopnost činí EPSDE zejména vhodným pro složité optimalizační úlohy, kde tradiční metody mohou selhat nebo kde je potřeba značně snížit počet hodnocení účelové funkce. [5]

1.5.3 JADE

Verze JADE (Adaptive Differential Evolution with Optional External Archive) byla navržena s novou mutační strategií DE/current-to-pbest, která představuje modifikaci mutační strategie DE/current-to-best. Tato strategie se liší tím, že nevyužívá globálně nejlepšího jedince, ale náhodně vybírá z 100p % nejlepších jedinců v populaci, čímž zvyšuje diverzitu a zabraňuje předčasné konvergenci k lokálním extrémům. Kromě toho JADE využívá archiv starších jedinců, do kterého se ukládají jedinci, jež byli nahrazeni svými potomky. Jedinci z archivu jsou poté využíváni při tvorbě mutačního vektoru, čímž se dále zvyšuje diverzita a zlepšuje směr hledání optimálního řešení. Archiv má omezenou kapacitu rovnou velikosti populace a při jeho naplnění jsou nově přidaní jedinci nahrazeni náhodně vybranými jedinci z archivu. [5]

Součástí JADE je také adaptace parametrů F a CR, kde základní adaptace probíhá generováním těchto parametrů náhodně pro každého jedince v každé generaci s využitím

Cauchyho a normálního rozdělení. Inovativním aspektem je komplexní adaptační mechanismus, který využívá historický záznam úspěšných hodnot F a CR, umožňující postupné přizpůsobování těchto parametrů pro zvýšení efektivity hledání. Tento adaptivní přístup značně zlepšuje schopnost algoritmu dynamicky reagovat na různé optimalizační výzvy bez potřeby předběžného ručního ladění parametrů. [6]

JADE byl testován na souboru klasických benchmarkových funkcí a ukázal lepší nebo alespoň srovnatelný výkon optimalizace ve srovnání s jinými klasickými nebo adaptivními algoritmy DE a odvozeným algoritmem PSO, přičemž JADE prokázal významnou převahu při optimalizaci problémů s vysokou dimenzionalitou. [6]

1.5.4 SaDE

Algoritmus SaDE (Self-adaptive Differential Evolution) využívá čtyři základní strategie mutace: DE/rand/1/bin, DE/rand/2/bin, DE/rand-to-best/2/bin a DE/current-to-rand/1. Dynamická selekce a adaptace těchto strategií na základě úspěšnosti generovaných řešení umožňuje algoritmu identifikovat a upřednostnit strategii, která je pro daný problém nejúčinnější. Tento proces adaptace zajišťuje, že SaDE může postupně optimalizovat svůj přístup k řešení problému, čímž zvyšuje pravděpodobnost dosažení globálního optima. [5]

Adaptace parametrů F a CR je dalším klíčovým prvkem SaDE. Zatímco parametr F je upravován individuálně pro každého jedince, což umožňuje personalizovanou míru mutace, parametr CR je upravován na základě zvolené strategie, což vede k jednotnosti při křížení uvnitř této strategie. Tato specifická adaptace parametrů podporuje efektivnější prozkoumávání a využívání prostoru řešení s ohledem na aktuální kontext optimalizace. [6]

SaDE tedy představuje pokročilou metodiku v rámci rodiny algoritmů diferenciální evoluce, která se vyznačuje vysokou adaptabilitou a schopností efektivně řešit široké spektrum optimalizačních problémů. Její výkon a adaptabilita jsou dále podporovány použitím archivu starších jedinců a adaptivního nastavení klíčových parametrů, což SaDE činí výkonným nástrojem pro moderní optimalizační výzvy. [5][6]

1.5.5 SHADE

Varianta SHADE (Success-History Based Parameter Adaptation for Differential Evolution) je vylepšením algoritmu JADE a zachovává si jeho klíčové prvky, jako jsou mutační strategie, externí archiv a adaptaci parametrů F a CR. V této variantě je proces adaptace parametrů dále rozšířen. Hodnoty F a CR se uchovávají a postupně aktualizují směrem k

úspěšným středním hodnotám získaným v předchozích generacích. Algoritmus SHADE navíc ukládá střední hodnoty pro každou generaci, což umožňuje sofistikovanější adaptaci parametrů v průběhu času. Díky této metodě, pokud jsou do archivu přidány hodnoty z méně úspěšné generace, původní, efektivnější hodnoty zůstávají nedotčeny, což zvyšuje robustnost algoritmu SHADE ve srovnání s JADE. Tato vlastnost zajišťuje, že SHADE je odolnější proti negativním vlivům neúspěšných generací a lépe se přizpůsobuje výzvám optimalizačních problémů. [2]

2 ALGORITMUS SHADE

Diferenciální evoluce (DE) je považována za jednoduchou, avšak efektivní metodu pro numerickou optimalizaci, což potvrzuje i její široké uplatnění v mnoha praktických problémech. Výkonnost algoritmu DE značně závisí na správném nastavení jeho kontrolních parametrů, což vedlo k vývoji mechanismů pro jejich samoadaptaci. Mezi nejznámější a nejúčinnější varianty DE patří JADE, který představuje adaptaci kontrolních parametrů a používá inovativní mutační strategii current-to-pbest/1 spolu s externím archivem pro ukládání dříve generovaných jedinců. Na základě těchto přístupů byl navržen SHADE (Success-History Adapted Differential Evolution), který rozšiřuje JADE o historicky založený mechanismus adaptace parametrů. Tento mechanismus ukládá úspěšná nastavení kontrolních parametrů do paměti a tyto historické údaje následně využívá pro generování nových hodnot parametrů, čímž zvyšuje efektivitu vyhledávání optimálních řešení. [2]

SHADE demonstruje lepší výkonnost než jeho předchůdci, včetně JADE, a byl úspěšně otestován na širokém spektru benchmarků. Tato významná vylepšení jsou důsledkem jeho schopnosti dynamicky se přizpůsobit a optimalizovat parametry F a CR na základě historických údajů o úspěchu, což umožňuje algoritmu lépe reagovat na různorodé optimalizační úkoly a zvyšuje jeho robustnost a spolehlivost. [2]

Navíc, SHADE zavádí nový přístup k nastavení parametru *p*, který umožňuje nastavit míru "chamtivosti" mutační strategie. Tento adaptivní přístup, spolu s použitím externího archivu pro udržování diverzity populace, činí SHADE výkonným nástrojem pro řešení široké škály optimalizačních problémů, od jednoduchých unimodálních funkcí až po složité multimodální a kompozitní funkce. [2]

SHADE tak představuje významný krok vpřed v oblasti diferenciální evoluce, což otevírá nové možnosti pro další výzkum a aplikace v numerické optimalizaci. Jeho schopnost adaptovat kontrolní parametry na základě úspěšné historie a jeho výkonnost na širokém spektru benchmarkových testů ukazují na potenciál SHADE jako robustního, efektivního a adaptivního nástroje pro globální optimalizaci. [2]

2.1 Princip

Princip algoritmu SHADE je velmi obdobný principu klasické DE. Největším rozdílem je zavedení paměti, kam se ukládají dříve použité hodnoty parametrů F a CR. Stejně jako u klasické DE, je počáteční populace inicializována náhodně pomocí pseudonáhodného

generátoru čísel s rovnoměrným rozdělením. Změny v chování přichází až v dalších krocích algoritmu. [7]

2.1.1 Mutace

Mutace v SHADE je založena na strategii "current-to-pbest/1", což je vylepšení oproti klasickému DE, kde je běžně používána strategie "rand/1". Mutační vektor se ve strategii "current-to-pbest/1" vypočítá podle následujícího vzorce: [7]

$$v_i = x_i + F_i(x_{pbest} - x_i) + F_i(x_{r1} - x_{r2}) \tag{12}$$

V SHADE se kombinují čtyři vzájemně odlišné vektory: aktuální vektor x_i, náhodně vybraný vektor x_{pbest} z NP × p nejlepších jedinců v populaci, a dva další náhodně vybrané vektory x_{r1} a x_{r2}. Hodnota p je generována náhodně pro každou mutaci s rovnoměrným rozdělením z rozsahu [p_{min}, 0.2], kde p_{min} = 2/NP. Vektor x_{r1} je vybrán náhodně z aktuální populace, zatímco vektor x_{r2} je vybírán náhodně ze sjednocení aktuální populace a externího archivu. Mutační konstanta F_i pro každou mutaci je generován z Cauchyho rozdělení s vybranou hodnotou z historické paměti M_F a mutační konstantou 0.1. Pokud generovaná hodnota F_i přesáhne 1, je omezena na 1. Pokud je F_i menší nebo rovno 0, je znovu generována. [7]

Tento proces mutace umožňuje SHADE lépe adaptovat se na dynamiku řešeného problému tím, že dynamicky upravuje hodnoty mutační konstanty a prahu křížení na základě historie úspěchu. To pomáhá algoritmu efektivněji prozkoumávat prostor řešení a předcházet předčasné konvergenci, což je obzvláště důležité ve vyšších dimenzích vyhledávacího prostoru. [7]

Inovace zavedená v SHADE tedy spočívá ve způsobu, jakým jsou generovány mutované vektory, a v adaptivním nastavování klíčových kontrolních parametrů na základě historie úspěchu, což představuje zásadní rozdíl oproti klasickému DE a je příkladem pokračujícího vývoje v oblasti evolučních algoritmů. [7]

Algorithm 1: SHADE

```
   // Initialization phase
1  G = 0;
2  Initialize population P_0 = (x_{1,0}, ..., x_{N,0}) randomly;
3  Set all values in M_CR, M_F to 0.5;
4  Archive A = ∅;
5  Index counter k = 1;
   // Main loop
6  while The termination criteria are not met do
7      S_CR = ∅, S_F = ∅;
8      for i = 1 to N do
9          r_i = Select from [1, H] randomly;
10         CR_{i,G} = randn_i(M_{CR,r_i}, 0.1);
11         F_{i,G} = randc_i(M_{F,r_i}, 0.1);
12         p_{i,G} = rand[p_min, 0.2];
13         Generate trial vector u_{i,G} by current-to-pbest/1/bin;
14     end
15     for i = 1 to N do
16         if f(u_{i,G}) ≤ f(x_{i,G}) then
17             x_{i,G+1} = u_{i,G};
18         else
19             x_{i,G+1} = x_{i,G};
20         end
21         if f(u_{i,G}) < f(x_{i,G}) then
22             x_{i,G} → A;
23             CR_{i,G} → S_CR, F_{i,G} → S_F;
24         end
25     end
26     Whenever the size of the archive exceeds |A|, randomly
       selected individuals are deleted so that |A| ≤ |P|;
27     if S_CR ≠ ∅ and S_F ≠ ∅ then
28         Update M_{CR,k}, M_{F,k} based on S_CR, S_F;
29         k++;
30         If k > H, k is set to 1;
31     end
32 end
```

Obrázek 5 Pseudokód algoritmu SHADE [2]

2.1.2 Křížení

V kroku křížení algoritmu SHADE se vytváří zkušební vektor *u* z mutačního vektoru *v* a původního vektoru *x*. Pro každou složku vektoru se pomocí pseudonáhodného generátoru čísel s rovnoměrným rozdělením generuje náhodná hodnota. Pokud je tato náhodná hodnota menší nebo rovna dané hodnotě prahu křížení CR_i, bude aktuální složka vektoru převzata ze zkušebního vektoru, v opačném případě bude převzata z původního vektoru. Tento proces také zahrnuje bezpečnostní opatření, které zajišťuje, že alespoň jedna složka vektoru bude převzata ze zkušebního vektoru. To je dáno náhodně generovaným indexem složky j_{rand}. [7]

Hodnota prahu křížení CR_i je generována z Gaussova rozdělení s průměrnou hodnotou parametru M_{CRr}, vybranou z historické paměti prahu křížení M_{CR} podle stejného indexu *r* jako v případě mutační konstanty, a hodnotou standardní odchylky 0.1. Pokud je generovaná

hodnota CR_i menší než 0, je nahrazena nulou, a pokud je větší než 1, je nahrazena jedničkou. [7]

Takto upravený krok křížení umožňuje SHADE efektivně kombinovat informace z více vektorů a přizpůsobit se tak různým charakteristikám optimalizačního problému. Zavedení adaptivního prahu křížení CR na základě historické paměti umožňuje algoritmu flexibilně reagovat na různé fáze hledání optimálního řešení, což je významný rozdíl oproti klasické diferenciální evoluci, kde je hodnota prahu křížení obvykle pevně nastavena. [7]

2.1.3 Selekce

Selekce jedinců do další generace je velmi jednoduchá. Algoritmus vybírá do další generace pouze ty jedince, jejichž hodnota účelové funkce je lepší nebo alespoň stejná jako hodnota jedinců v současné populaci. [7]

2.1.4 Paměť

Historické paměti M_F a M_{CR} jsou inicializovány na začátku běhu algoritmu na hodnoty 0.5, ale během evoluce se jejich složky mění. Tyto paměti slouží k uchování úspěšných hodnot F a CR použitých ve fázích mutace a křížení, které byly úspěšné ve smyslu vytvoření zkušebního jedince, který je lepší než původní jedinec. V každé generaci jsou tyto úspěšné hodnoty ukládány do příslušných polí S_F a S_{CR}. Po každé generaci je aktualizována jedna buňka pamětí M_F a M_{CR}. Tato buňka je určena indexem *k*, který začíná na 1 a po každé generaci se zvýší o 1. Když přeteče velikost paměti H, je resetována na 1. Nové hodnoty k-té buňky pro M_F a M_{CR} jsou vypočítány podle Lehmerova váženého průměru. [7]

Pokud pole S_F není prázdné, vypočítá se pro M_F vážený Lehmerův průměr hodnot v S_F. V opačném případě zůstává M_F nezměněno. Stejně tak pokud pole S_{CR} není prázdné, vypočítá se pro M_{CR} vážený Lehmerův průměr hodnot v S_{CR}. Pokud je pole S_{CR} prázdné, zůstává M_{CR} nezměněno. Vážený Lehmerův průměr se vypočítá jako vážený průměr, kde váhy *w* jsou založeny na zlepšení hodnoty cílové funkce mezi zkušebními a původními jedinci v aktuální generaci G. Tímto způsobem SHADE upravuje historické paměti, aby odrážely nejúspěšnější hodnoty F a CR, což umožňuje lepší přizpůsobení algoritmu pro budoucí generace. [7]

2.2 Varianty SHADE

Algoritmus SHADE existuje v mnoha variantách, které se odlišují svými přístupy k adaptaci parametrů nebo dokonce rozdílnými mutačními strategiemi. V této kapitole budou popsány některé z těchto variant algoritmu SHADE.

2.2.1 LSHADE

V LSHADE, což je rozšíření standardního algoritmu SHADE, došlo ke zvýšení efektivity vyhledávání díky zavedení postupu pro lineární snižování počtu jedinců v populaci. Tento přístup vedl ke zrychlení procesu konvergence a k redukci celkové složitosti algoritmu. Efektivita LSHADE byla ověřena prostřednictvím testů na souboru funkcí IEEE CEC 2017, přičemž bylo dosaženo srovnání s nejmodernějšími variantami DE a CMA-ES. Výsledky testů ukázaly, že LSHADE je buď nadřazený nebo srovnatelný s těmito pokročilými algoritmy ve smyslu výkonnosti. [5]

2.2.2 iL-SHADE

Ve variantě iL-SHADE se zachovávají základní prvky algoritmu L-SHADE, jako jsou mutační strategie, využití externího archivu a postup pro snižování velikosti populace. Avšak dochází k drobným úpravám v nastavení: hodnoty pro historickou paměť CR jsou na začátku nastaveny na vyšší hodnotu 0.8 namísto původních 0.5, což naznačuje změnu v přístupu k adaptivnímu nastavení těchto parametrů. Kromě toho, historická paměť obsahuje specifické hodnoty pro F a CR, konkrétně dvojici s hodnotou 0.9, aby se povolilo použití vyšších hodnot pro obě proměnné současně. Tento přístup se ukázal být klíčový ve fázi, kdy algoritmus začíná prozkoumávat prostor řešení. [5]

Změna nastavení měla za cíl optimalizovat používání vysokých hodnot CR společně s nízkými hodnotami F v rané fázi hledání. Adaptace historických hodnot pro následující generace probíhala pomocí vážených Lehmerových průměrů, kde každé hodnotě z aktuální generace byla přiřazena stejná váha. [5]

Další novinkou v iL-SHADE je aktualizace hodnoty p, která určuje míru "chamtivosti" algoritmu, na základě nově definovaného vzorce. Tato inovace byla empiricky ověřena pomocí sady testovacích funkcí v různých dimenzích, kde iL-SHADE prokázal svou schopnost konkurovat ostatním pokročilým algoritmům. [5]

2.2.3 LSHADE-EpSin

LSHADE-EpSin představuje pokročilou verzi algoritmu L-SHADE, která využívá inovativní přístup k automatické adaptaci mutační konstanty prostřednictvím sad sinusoidálních metod. Tento soubor zahrnuje jak nereaktivní metodu s postupně klesajícím efektem, tak adaptivní metodu, která se opírá o historické údaje a zvyšuje úpravu podle sinusoidální funkce. Tato kombinace metod umožňuje algoritmu najít optimální rovnováhu mezi průzkumem prostoru řešení a efektivním využíváním nalezených informací. Navíc, pro zlepšení schopnosti efektivně vyhledávat v pozdějších fázích vývoje byla implementována technika lokálního prohledávání založená na Gaussově rozdělení. LSHADE-EpSin byl testován na sadě testovacích funkcí CEC 2014, a jeho výkonnost byla srovnána s LSHADE a dalšími vedoucími optimalizačními algoritmy. Výsledky testů ukázaly vylepšenou efektivitu a robustnost LSHADE-EpSin, což potvrzuje jeho přidanou hodnotu v oblasti evolučního výpočtu. [5]

2.2.4 Db SHADE

Db-SHADE rozšiřuje možnosti algoritmu SHADE řešením problémů s předčasnou konvergencí, které se u této rodiny algoritmů mohou vyskytovat. Tato verze zavádí mechanismus adaptace parametrů založený na vzdálenosti, což umožňuje lepší průzkum prostoru řešení v případech s vysokou dimenzí, přičemž si zachovává efektivitu v oblasti výpočetní náročnosti. Hodnoty mutace a křížení se upravují podle euklidovské vzdálenosti mezi aktuálním a mutovaným vektorem, což poskytuje větší příležitost pro průzkum těm jedincům, kteří se ve vyhledávacím prostoru vzdálili nejdále. Testování ukázalo, že Db-SHADE dosahuje lepších výsledků v porovnání s jinými variantami SHADE, což dokládá jeho efektivitu a přínos pro proces vyhledávání optimálního řešení. [5]

2.2.5 COLSHADE

COLSHADE rozšiřuje možnosti původního algoritmu L-SHADE tím, že zahrnuje adaptivní Lévyho lety pro efektivnější globální průzkum a dynamickou toleranci k udržení diverzity mezi řešeními. Díky zavedení nové mutační strategie zvané levy/1/bin, která se kombinuje s lokálními vyhledávacími metodami na základě strategie current-to-pbest, COLSHADE poskytuje sofistikovanější nástroj pro optimalizaci. Tento algoritmus se vyznačuje schopností adaptovat použití různých strategií mutace v závislosti na aktuálním stadiu hledání optimálního řešení, což mu umožňuje flexibilně reagovat na různorodé výzvy optimalizačního procesu. [5]

3 VÁHOVÉ MODELY

Váhové modely určují, podle jakého klíče se budou adaptovat parametry F a CR. Existuje mnoho různých druhů váhových modelů. Ve své podstatě si každý uživatel může vytvořit vlastní váhový model, který následně použije.

3.1 Váhový model založený na zlepšení fitness

Tento model přiděluje váhy jedincům na základě míry zlepšení jejich fitness. Váha každého jedince je přímo úměrná absolutní hodnotě zlepšení jeho kondice ve vztahu k celkovému absolutnímu zlepšení všech jedinců. Tímto způsobem je jedincům s větším zlepšením přidělena vyšší váha, což znamená, že větší zlepšení je považováno za významnější.

3.2 Váhový model na bázi inverzního zlepšení fitness

Tento model pracuje na opačném principu než předchozí. Jedinci s menším zlepšením dostávají vyšší váhy, což znamená, že menší zlepšení je považováno za významnější. K tomu je použit inverzní vztah, kde menší hodnoty zlepšení dostávají po přidání velmi malé konstanty (aby se předešlo dělení nulou) větší relativní váhu.

3.3 Kvartilové váhové modely

Tento přístup rozděluje zlepšení do čtyř skupin (kvartilů) podle jejich velikosti. Jedincům v nižších kvartilech, tedy s menším zlepšením, jsou přiděleny vyšší váhy. Model tedy preferuje menší zlepšení. Opačný model, inverzní kvartilový váhový model, funguje naopak tak, že vyšší kvartily (větší zlepšení) dostávají vyšší váhy.

3.4 Model vah založený na vzdálenosti od mediánu

Tento model váží jedince na základě vzdálenosti jejich zlepšení od referenční hodnoty (mediánu). Váhy jsou přiděleny inverzně vzdálenostem, tedy jedinci blíže k referenční hodnotě dostávají vyšší váhu. Tento model preferuje jedince s výsledky blízkými mediánovému zlepšení. Model s inverzními vzdálenostmi naopak přiděluje vyšší váhy těm, kteří jsou od referenční hodnoty vzdálenější.

3.5 Lineárně klesající váhové modely

V tomto modelu jsou jedinci seřazeni podle velikosti zlepšení a přiděleny jim váhy podle lineárně klesající funkce. Nejmenší zlepšení dostává nejvyšší váhu a váhy se snižují s

rostoucím zlepšením. Inverzní model lineárně klesajících vah funguje opačně, kde největší zlepšení získá nejvyšší váhu a váhy se snižují s klesajícím zlepšením. Obě varianty normalizují váhy tak, aby jejich součet byl roven 1, což zajišťuje konzistentní využití v rámci modelování nebo algoritmických procesů.

II. PRAKTICKÁ ČÁST

4 TESTOVÁNÍ VLIVU VAH

Pro testování, jaký vliv mají váhy na adaptivní proces diferenciální evoluce, jsem zvolil algoritmus SHADE. Algoritmus je blíže popsán v teoretické části. Algoritmus, na kterém bylo testování prováděno, byl vytvořen na základě pseudokódu, taktéž přiloženého v teoretické části práce. Algoritmus je napsán v jazyce Python, protože tento jazyk je na tento typ úloh jeden z nejvhodnějších. V kódu bylo využito osmi váhových modelů, přičemž každý model se liší způsobem výpočtu vah a samotným přiřazováním vah k jedincům.

```
def shade(population_size, dimensions, func_num, max_iterations, H, weight_model):
    objective_function = cec2022_func(func_num)

    MCR_history = []
    MF_history = []

    G = 0
    P = np.random.rand(population_size, dimensions) * 200 - 100  # Assuming bounds [-100, 100] for CEC functions
    fitness = np.array([objective_function.values(P[i:i + 1].T).ObjFunc for i in range(population_size)]).flatten()
    MCR = np.full(H, 0.5)
    MF = np.full(H, 0.5)
    A = []
    k = 0
    best_fitness_over_time = []

    while G < max_iterations:
        SCR = []
        SF = []
        fitness_improvements = []
        for i in range(population_size):
            ri = np.random.randint(H)
            CRi = np.random.normal(MCR[ri], 0.1)
            Fi = max(0.1, min(1, np.random.normal(MF[ri], 0.1)))
            pi = np.random.uniform(0.05, 0.2)

            ui = generate_trial_vector(P, i, pi, Fi, CRi)
            ui_fitness = objective_function.values(ui.reshape(1, -1).T).ObjFunc[0]
            if ui_fitness < fitness[i]:
                # Calculate improvement before updating fitness[i]
                improvement = fitness[i] - ui_fitness
                # Update fitness[i]
                fitness[i] = ui_fitness
                P[i] = ui
```

Obrázek 6 Ukázka implementace SHADE

4.1 Použité váhové modely

Pro testování bylo využito celkově osmi váhových modelů. Základem jsou čtyři modely, které se liší výpočtem vah. Druhá čtveřice modelů je inverzní k původním čtyřem a liší se tedy způsobem přiřazování vah. Například v modelu, který počítá váhy na základě zlepšení hodnoty fitness, se nejvyšší váhy přiřazují jedincům, kteří dosáhli největšího zlepšení. Inverzní model naopak přiděluje nejvyšší váhy jedincům s nejmenším zlepšením.

Použité modely:

- Výpočet vah na základě zlepšení fitness (Fitness)

- Výpočet vah na základě kvartilů (Quartile)
- Výpočet vah na základě vzdálenosti fitness od referenční hodnoty (Distance)
- Lineárně klesající váhy (Linear descending)
- Inverzní varianty těchto modelů

Jednotlivé váhové modely jsou podrobněji popsány v teoretické části. Všechny váhové modely jsou vytvořeny jako samostatné funkce, které jsou volány v běhu algoritmu. Vstupním parametrem je hodnota fitness, na základě, které se vypočítá a následné přiřadí váha pro každého jedince. V ukázce kódu níže je zobrazeno, jakým způsobem je váhový model volán a jak probíhá adaptace parametrů F a CR.

```
if SCR and SF:
    weights = weight_model(fitness_improvements)
    mean_CR = sum(w * cr for w, cr in zip(weights, SCR)) / sum(weights)
    mean_F = sum(w * f ** 2 for w, f in zip(weights, SF)) / sum(w * f for w, f in zip(weights, SF))

    # Update MCR and MF
    MCR[k] = mean_CR
    MF[k] = mean_F
```

Obrázek 7 Implementace váhového modelu

4.2 Testovací sada

Pro testování byla využita testovací sada CEC2022. Tato sada obsahuje 12 různých funkcí. Pět funkcí jsou základní funkce, které jsou různě posunuty a otočeny. Zároveň testovací sada má přednastavené hranice prohledávaného prostoru hodnot na (-100; 100).

1. Zakharovova funkce:

$$f(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5x_i\right)^4 \quad (13)$$

   Minimální hodnota: 300

2. Rosenbrockova funkce:

$$f(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_{i+1} - 1)^2) \quad (14)$$

   Minimální hodnota: 400

3. Schafferova F7 funkce:

$$\mathrm{f16(x)} = [\frac{1}{\mathrm{D}-1}\sum_{i=1}^{D-1}(\sqrt{s_i}(\sin(50.0s_i^{0.2})+1))\,]^2\,, s_i = \sqrt{x_i^2 + x_{i+1}^2} \tag{15}$$

   Minimální hodnota: 600

4. Rastriginova funkce:

$$\mathrm{f(x)} = \sum_{i=1}^{D}(\,x_i^2 - 10\cos(2\pi x_i) + 10) \tag{16}$$

   Minimální hodnota: 800

5. Levyho funkce:

$$\mathrm{f(x)} = \sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i - 1)^2[1 + 10\sin^2(\pi w_i - 1)] + (W_D - 1)^2\left[1 + \sin^2(2\pi w_{D)}\right]$$

$$\text{Kde } w_i = 1 + \frac{x_i - 1}{4}, \forall i = 1, \ldots, D \tag{17}$$

   Minimální hodnota: 900

Další tři funkce jsou hybridní. Skládají se z více funkcí a různých parametrů. Tyto funkce mají co nejvíce simulovat reálné optimalizační problémy.

6. Hybridní funkce hf02:

   Skládá se z:

   - Bent cigar funkce
   - HGBat funkce
   - Rastriginovy funkce

   Minimální hodnota: 1800

7. Hybridní funkce hf06:

   Skládá se z:

   - HGBat funkce
   - Katsuura funkce
   - Ackleyho funkce
   - Rastriginovy funkce

- Modifikované Schwfelovy funkce
- Schafferovy funkce

Minimální hodnota: 2200

8. Hybridní funkce hf10:

   Skládá se z:

   - Katsuura funkce
   - HappyCat funkce
   - Rozšířené Griewankovy funkce plus Rosebrockovy funkce
   - Modifikované Schwefelovy funkce

   Minimální hodnota: 2000

Poslední čtyři funkce jsou složené funkce, které lépe slučují vlastnosti jednotlivých funkcí a zachovávají spojitost v oblasti optima.

9. Složená funkce cf01:

   Skládá se z:

   - Rotované Rosenbrockovy funkce
   - Vysoce podmíněné eliptické funkce
   - Rotované Bent Cigar funkce
   - Rotované Discus funkce

   Minimální hodnota: 2300

10. Složená funkce cf02:

    Skládá se z:

    - Rotované Schwefelovy funce
    - Rotované Rastriginovy funkce
    - HGBat funce

    Minimální hodnota: 2400

11. Složená funkce cf06:

    Skládá se z:

    - Rozšířené Schafferovy funkce
    - Modifikované Schwefelovy funkce
    - Griewankovy funkce
    - Rosenbrockvy funkce
    - Rastriginovy funkce

    Minimální hodnota: 2600

12. Složená funkce cf07:

    Skládá se z:

    - HGBat funkce
    - Rastriginovy funkce
    - Modifikované Schwefelovy funkce
    - Bent Cigar funkce
    - Vysoce podmíněné eliptické funkce
    - Rozšířené Schafferovy funkce

    Minimální hodnota: 2700

4.3 Vstupní parametry

Použitý SHADE algoritmus má několik vstupních parametrů. Konkrétně to jsou velikost populace, maximální počet generací, dimenze, velikost archivu, optimalizační problém a váhový model. Nastavení parametrů pro testování bylo následující:

- Velikost populace: 100
- Počet generací: 2500
- Dimenze: 10
- Velikost archivu: 20

Pomocí vnořených for cyklů byly vkládány výše uvedené testovací funkce a váhové modely. Tento přístup zaručil, že všechny váhové modely byly otestovány na všech funkcích. Zároveň byly všechny kombinace funkcí a váhových modelů spuštěny 30krát, což umožnilo získat dostatečný počet dat pro vyhodnocení.

5 VÝSLEKDY TESTOVÁNÍ

V této kapitole budou představeny dosažené výsledky testování. Výsledky jsou prezentovány ve formě tabulek a konvergenčních grafů.

Tabulky obsahují statistické údaje a skóre získané provedením Mann-Whitneyova U-testu. Při testování byly porovnávány nejlepší dosažené hodnoty fitness ve 30 bězích. Testování probíhalo ve formě každého s každým po jednotlivých dvojicích. Vzájemně se tedy porovnaly všechny výsledky pro jednotlivé funkce. Vítězná varianta získala 1 bod, poražené byl odebrán 1 bod a při remíze obě testované varianty zůstaly na 0.

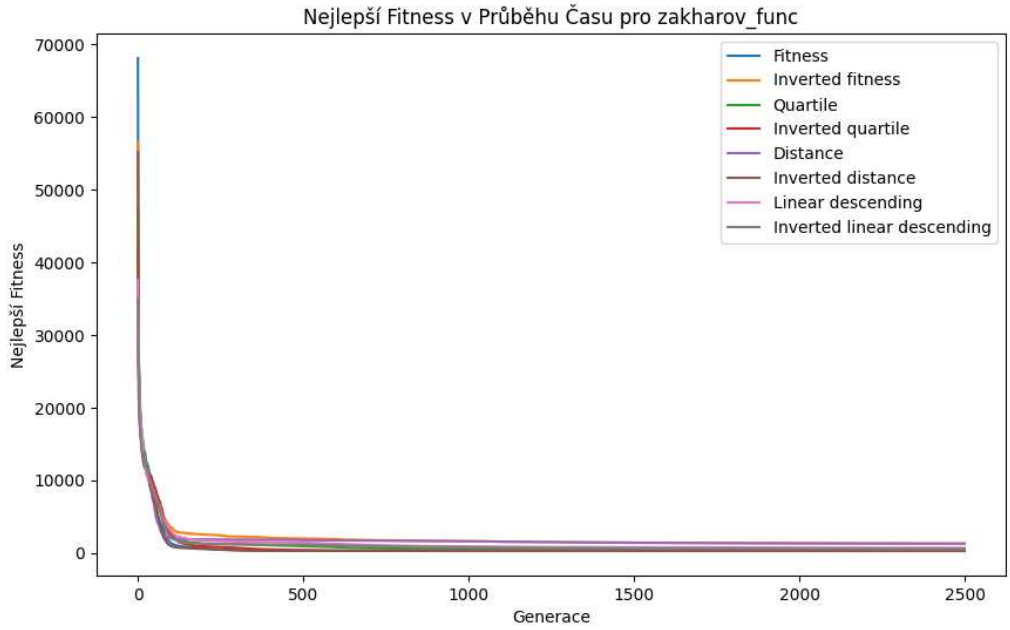
Konvergenční grafy jsou vykresleny z průměru průběžných hodnot fitness ze všech 30 běhů.

5.1 Zakharovova funkce

Tabulka 3 Výsledky pro Zakharovovu funkci

Minimální hodnota funkce:			300			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	300.0	300.0	300.0	300.0	0.0	6
Inv. fitness	303.9767	4833.6885	1316.1325	838.4838	1220.4197	-6
Quartile	300.0	852.7667	386.9121	317.1857	144.4699	1
Inv. quartile	300.0	300.0001	300.0	300.0	0.0	3
Distrance	300.02	4827.5957	1306.1227	881.2191	1001.63	-6
Inv. distance	300.0	300.0	300.0	300.0	0.0	6
Linear descendig	300.0	2716.5847	723.2214	409.9246	639.0574	-2
Inv. linear descending	300.0	2344.6016	577.9355	385.6564	487.5716	-2

Z výsledků je zřejmé, že pro Zakharovovu funkci nejlépe fungují váhové modely fitness a inv. distance. Na druhé straně, modely distance a inv. fitness se ukázaly jako nejméně účinné. Rozdíl mezi těmito modely je signifikantní. Podobně slabé výsledky vykazuje i model inv. fitness. Lepších výsledků dosáhl model inv. quartile, avšak rozdíly ve výsledcích oproti vítězným modelům nejsou tak markantní. Quartile model sice zaostával za třemi nejlepšími, ale přiblížil se k nim. Modely založené na lineárním snižování vah sice nevykázaly nejhorší výsledky, ale ze statistik vyplývá, že měly obtíže dosáhnout globálního minima.



Obrázek 8 Konvergenční graf pro Zakharovovu funkci

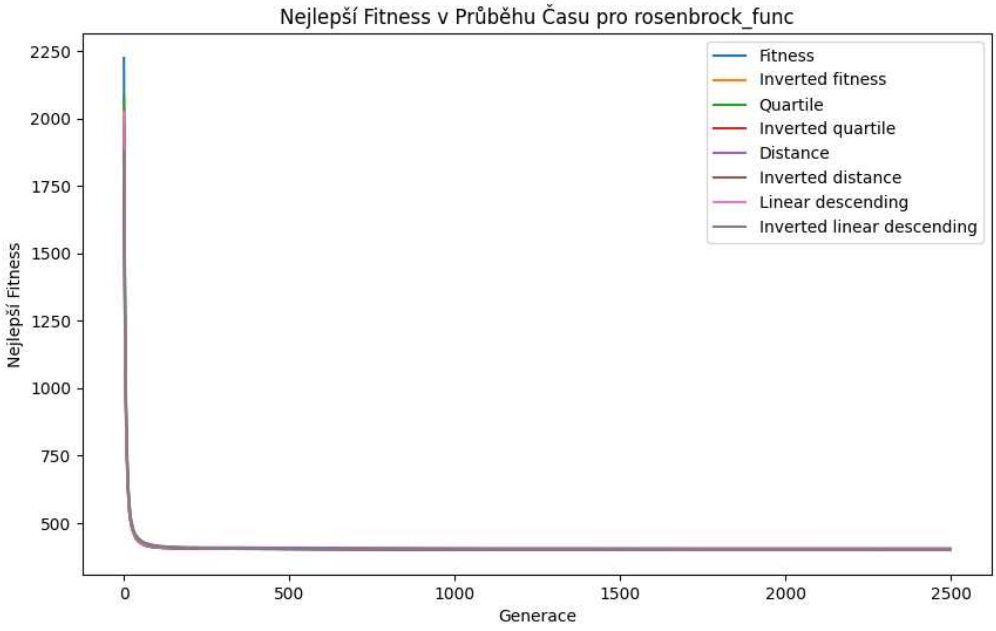
Z konvergenčního grafu je patrné, že modely se dostaly ke globálnímu minimu během cca 100 prvních iterací algoritmu. Žádný z modelů neměl výraznější obtíže se přiblížit z prvotních hodnot ke globálnímu minimu. Ovšem ze statistik výše je patrné, že finální nalezení globálního minima bylo pro některé modely problematické.

5.2 Rosenbrockova funkce

Tabulka 4 Výsledky pro Rosenbrovkovu funkci

Minimální hodnota funkce:			400			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	400.0	403.9866	400.2658	400.0	0.9944	6
Inv. fitness	400.0001	409.586	402.8401	400.5373	3.4719	-2
Quartile	400.0001	404.9182	402.3633	402.8795	1.7315	-2
Inv. quartile	400.0	403.9866	400.2017	400.0135	0.721	3
Distrance	400.0004	461.2598	406.6616	406.0766	10.6256	-7
Inv. distance	400.0	403.9866	400.3987	400.0	1.196	6
Linear descendig	400.0	406.5349	401.9437	401.2015	2.0267	-2
Inv. linear descending	400.0	405.4819	402.1679	2.085	402.8085	-2

U Rosenbrockovy funkce jsou výsledky mezi jednotlivými modely mnohem vyrovannější než u Zakharovovy funkce. Přesto podle výsledků Mann-Whitneyova U-testu jsou opět nejvhodnějšími modely fitness a inv. distance. Naopak nejméně vhodným se ukázal distance model. Podle statistik se všechny modely dokázaly bez větších problémů dostat ke globálnímu minimu. Všechny ostatní zkoumané modely vykazují velmi podobné statistiky. A neliší se ani ve výsledcích testování, ve kterém získaly shodné skóre.



Obrázek 9 Konvergenční graf pro Rosenbrockovu funkci

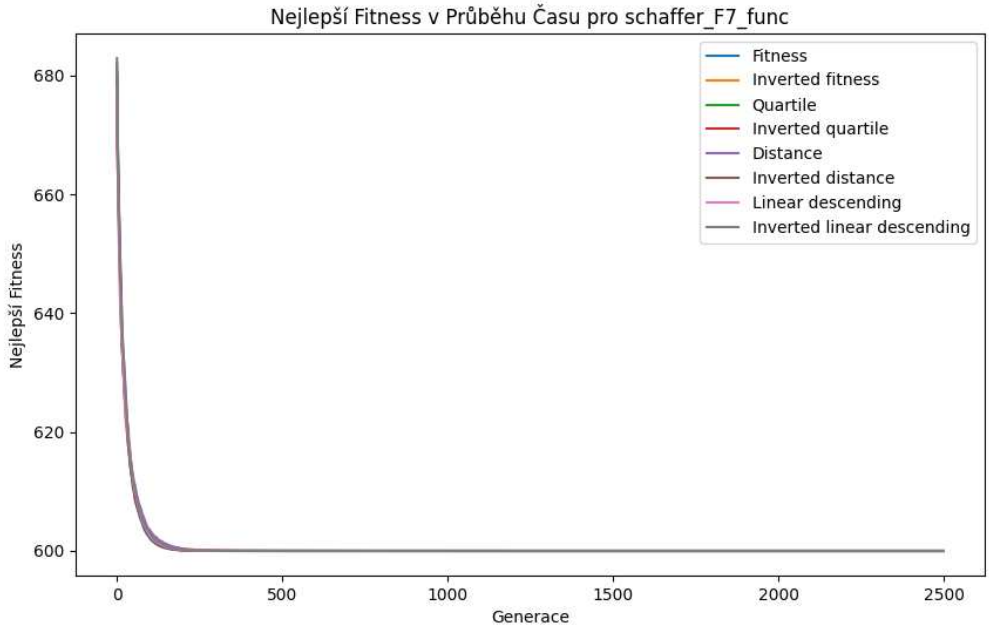
Velice těsné výsledky lze vypozorovat i z konvergenčního grafu. Žádný z testovaných modelů neměl problém se dostat ke globálnímu minimu. Všechny modely se dostaly k minimální hodnotě během prvních iterací algoritmu. To poukazuje na fakt, že žádný z testovaných modelů není vyloženě nevodný pro tuto funkci.

5.3 Schafferova F7 funkce

Tabulka 5 Výslekdy pro Schafferovu F7 funkci

Minimální hodnota funkce:			600			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	600.0	600.0	600.0	600.0	0.0	1
Inv. fitness	600.0	600.0	600.0	600.0	0.0	-7
Quartile	600.0	600.0	600.0	600.0	0.0	1
Inv. quartile	600.0	600.0	600.0	600.0	0.0	1
Distrance	600.0	600.0	600.0	600.0	0.0	1
Inv. distance	600.0	600.0	600.0	600.0	0.0	1
Linear descendig	600.0	600.0	600.0	600.0	0.0	1
Inv. linear descending	600.0	600.0	600.0	600.0	0.0	1

Testování na Schafferovy F7 funkci přineslo velice těsné výsledky mezi všemi váhovými modely. Ze statistik lze vyčíst, že žádný z modelů nijak nezaostával a všechny dosáhly rovnocenných výsledků. Výsledky Mann-Whitneyova U-testu ale poukazují, že model inverzní fitness má nedostatky oproti ostatním modelům. Zatímco všechny ostatní modely se ve všech běžích dokázaly dostat přesně na globální minimum, tento model se v některých případech nedostal na přesné minimum. Ale přiblížil se natolik, že rozdíly v hodnotách fitness nastávaly až na marginálních desetinných místech.



Obrázek 10 Konvergenční graf pro Schafferovu F7 funkci

I na konvergenčním grafu je zcela patrné, že výsledky byl velice těsné a žádný z modelů nezaostával za ostatními. Všechny modely se ke globálnímu minimu přiblížily zhruba za stejný počet iterací.

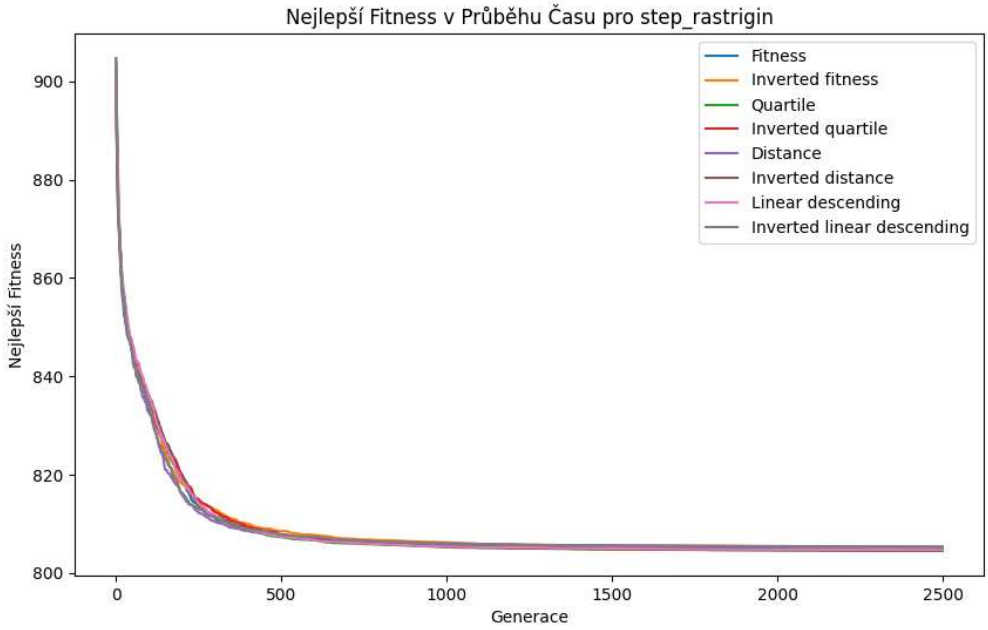
5.4 Rastriginova funkce

Tabulka 6 Výsledky pro Rastriginovu funkci

Minimální hodnota funkce:			800			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	801.2315	808.0756	804.8421	804.2775	1.6553	0
Inv. fitness	801.1768	808.2288	805.2805	805.1862	1.3208	-2
Quartile	801.2488	807.2176	804.4927	804.5894	1.6408	2
Inv. quartile	801.1149	808.1289	804.6143	804.3584	1.4837	2
Distrance	802.0834	808.1582	805.3761	805.1561	1.3217	-2
Inv. distance	801.1903	807.1749	804.6853	804.6469	1.5177	0
Linear descendig	801.1095	808.1487	804.6918	805.0465	1.4202	0
Inv. linear descending	803.1344	808.0797	805.2581	805.0719	1.3264	0

Pro Rastriginovu funkci dosahují nejlepších výsledků modely quartile a inv. quartile, což potvrzuje Mann-Whitneyův U-test. Naopak modely inv. fitness a distance vykazují nejhorší

výsledky. Podle statistik však situace není tak jednoznačná. Všechny modely mají velmi podobné statistické ukazatele a nelze jednoznačně určit, který model je pro danou funkci nejvhodnější. To potvrzují i výsledky testování, které jsou velmi těsné a většina modelů z nich odnesla 0 bodů. Navíc se žádný model nedokázal dostat přímo ke globálnímu minimu, ale všechny modely se nacházely v jeho velmi blízkém okolí.



Obrázek 11 Konvergenční graf pro Rasriginovu funkci

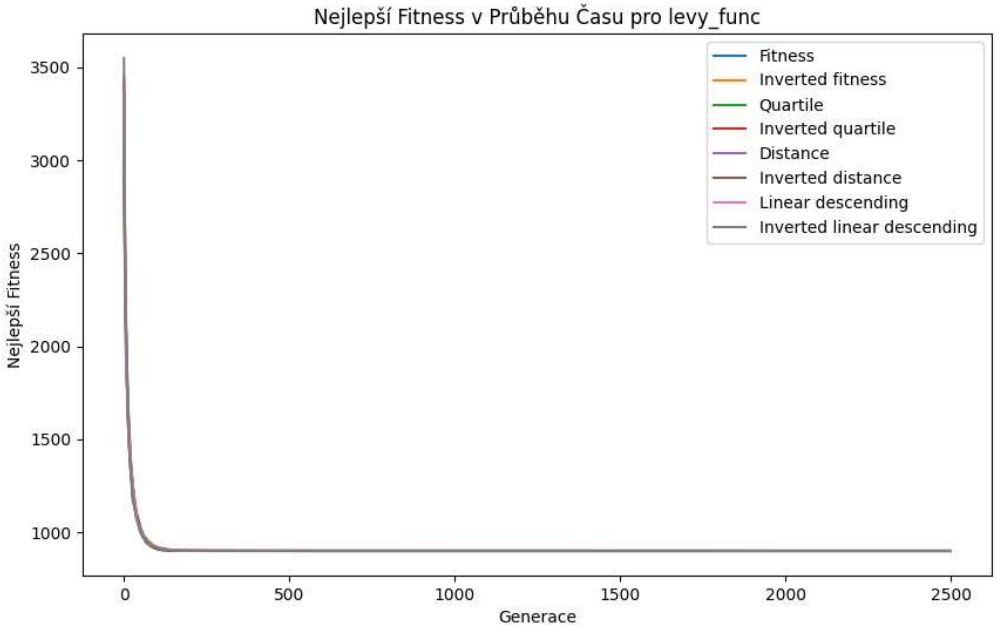
Konvergenční graf opět ukazuje vyrovnanost váhových modelů. Lze ale pozorovat, že u této funkce bylo pro váhové modely obtížnější se přiblížit ke globálnímu minimu. Vzhledem k tomu, že tento problém měly všechny váhové modely, je zde předpoklad, že to je způsobeno komplexností funkce.

5.5 Levyho funkce

Tabulka 7 Výsledky pro Levyho funci

Minimální hodnota funkce:			900			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	900.0	900.0	900.0	900.0	0.0	5
Inv. fitness	900.0	900.0253	900.167	900.0036	0.4085	-7
Quartile	900.0	900.0933	900.0032	900	0.00167	1
Inv. quartile	900.0	900.0	900.0	900.0	0.0	5
Distrance	900.0	901.4724	900.1717	900.0004	0.3892	-4
Inv. distance	900.0	900.0	900.0	900.0	0.0	5
Linear descendig	900.0	900.6603	900.0306	900.0	0.1214	-2
Inv. linear descending	900.0	900.2337	900.0132	900.0002	0.0444	-3

Výsledky pro Levyho funkci jsou velmi vyrovnané. Všechny váhové modely se dokázaly dostat až k samotnému globálnímu minimu. Ve všech 30 běžích dosáhl fitness model, inv. quartile model a inv. distance model lokálního minima. Statistiky těchto modelů jsou zcela vyrovnané. Ostatní modely však výrazně nezaostávají a mají průměr velmi blízký globálnímu minimu. Ale značné rozdíly ve výsledcích Mann-Whitneyho U-testu poukazují na to, že inv. fitness model je nejméně vhodný pro tuto funkci.



Obrázek 12 Kovergenční graf pro Levyho funkci

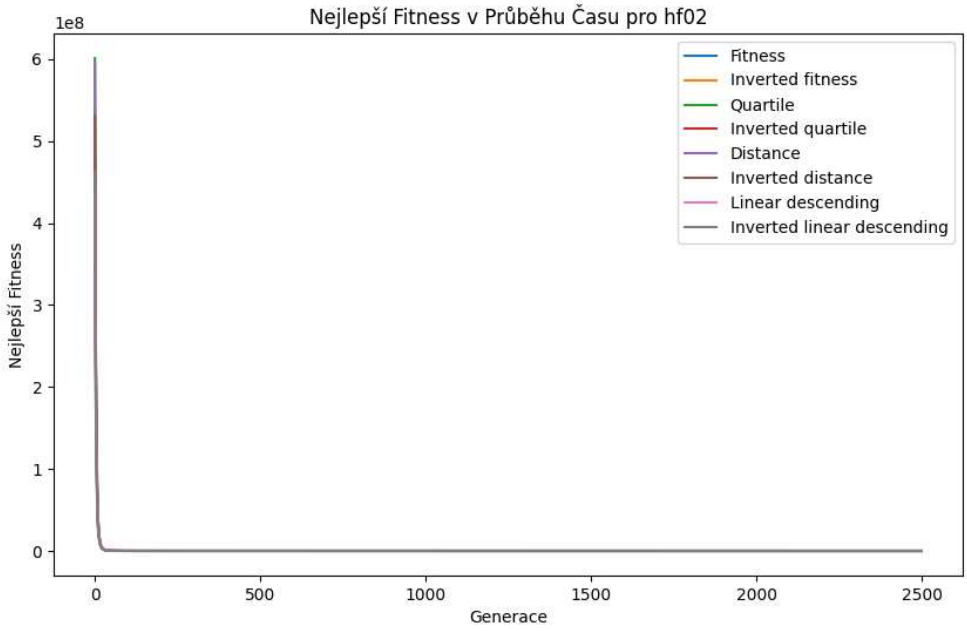
Konvergenční graf opět ukazuje vyrovnanost všech modelů. Všechny modely se dokázaly ke globálnímu minimu dostat velice rychle a konvergenční křivky pro jednotlivé modely se prakticky překrývají.

5.6 Hybridní funkce hf02

Tabulka 8 Výsledky pro hybridní funkci hf02

Minimální hodnota funkce:			1800			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	1800.0608	1802.2248	1800.7269	1800.4701	0.5711	5
Inv. fitness	1801.0644	1869.9274	1815.7146	1810.8433	14.2077	-4
Quartile	1800.089	1910.9925	1812.2222	1801.5704	27.6196	2
Inv. quartile	1800.0334	1809.0681	1801.9345	1801.1304	2.4589	4
Distrance	1800.2018	5553.4757	2052.6409	1867.1911	684.6549	-7
Inv. distance	1800.0516	1805.1405	1800.8798	1800.4237	1.2291	5
Linear descendig	1800.292	2328.8216	1833.1618	1807.0228	96.9409	-3
Inv. linear descending	1800.1551	1997.6252	1820.2251	1806.0744	46.6506	-2

Statisticky nejlépe u této funkce opět vypadá fitness model. Podobně dobré výsledky má i model inv. distance a inv. quartile. Naopak jednoznačně nejhorší výsledky má distance model. Sice se dokázal přiblížit ke globálnímu minimu, ale jeho nejhorší výsledek je od něj velmi vzdálen. To, že tento model měl konstantně problémy s dosažením globálního minima, dokládá i jednoznačně nejvyšší průměrná hodnota fitness. Ostatní modely neměly zdaleka takový problém se dostat ke globálnímu minimu. Linear descending model sice má nejhorší výsledek výrazně vzdálený od globálního minima, ale podle ostatních statistik lze usuzovat, že šlo spíše o ojedinělé uváznutí v lokálním minimu.



Obrázek 13 Konvergenční graf pro hybridní funkci hf02

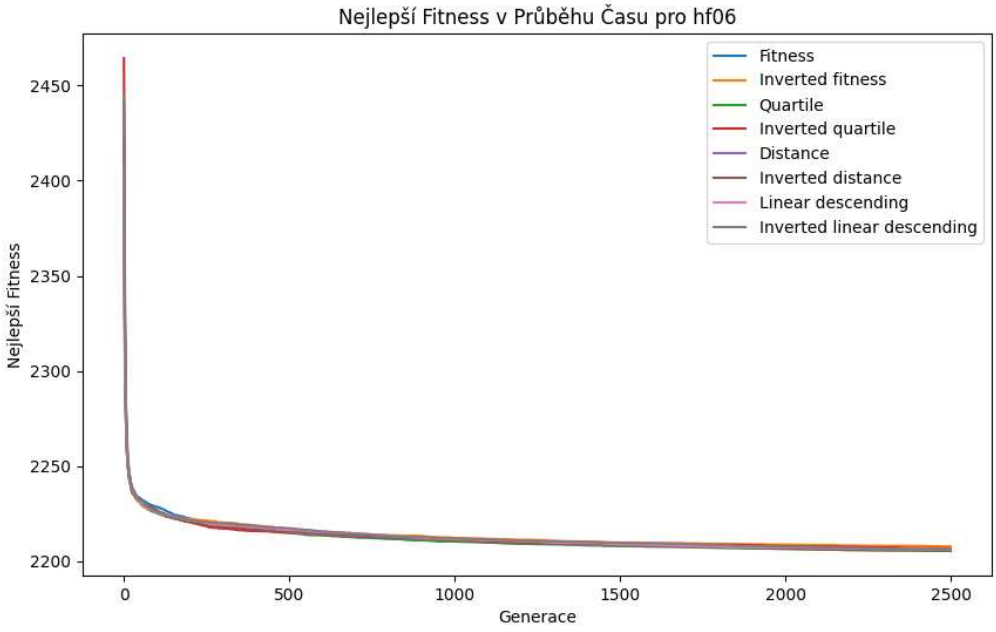
Z grafu je patrné, že tato funkce nabývá vysokých počátečních hodnot. Modely se sice velice rychle dokázaly dostat k výrazně nižším hodnotám ale nic jiného z toho grafu nelze s určitostí vyčíst.

5.7 Hybridní funkce hf06

Tabulka 9 Výsledky pro hybridní funkci hf06

Minimální hodnota funkce:			2200			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	2201.5248	2214.8133	2205.914	2205.448	3.2307	0
Inv. fitness	2201.842	2220.6708	2207.901	2205.5098	5.7498	0
Quartile	2202.2079	2214.8678	2205.5221	2204.6549	2.8594	0
Inv. quartile	2201.6773	2220.9882	2206.7915	2204.9406	5.0152	0
Distrance	2201.7059	2220.5031	2206.3104	2205.2344	4.4127	0
Inv. distance	2201.4899	2213.959	2205.5676	2205.42	2.9404	0
Linear descendig	2202.4265	2220.6568	2206.3123	2205.1516	4.6387	0
Inv. linear descending	2201.5757	2221.0161	2206.0782	2204.6617	4.7426	0

U hybridní funkce hf06 je možné, že při provádění Mann-Whitneyova U-testu došlo k nějaké chybě, ale test byl prováděn několikrát, pokaždé se stejným výsledkem. Nicméně statistiky opět ukazují na vyrovnanost všech váhových modelů. Všechny modely se dokázaly dostat blízko ke globálnímu minimu, a maximální hodnota ze všech běhů je u všech modelů velmi podobná. Lze tedy předpokládat, že žádný model nemá u této funkce výhodu.



Obrázek 14 Konvergenční graf pro hybridní funkci hf06

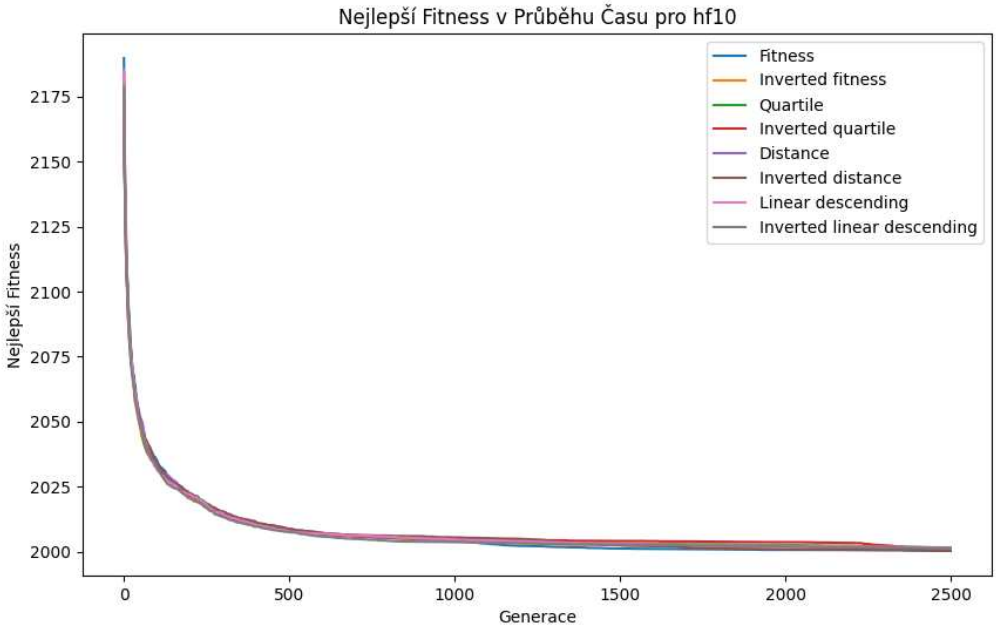
U tohoto konvergenčního grafu lze opět pozorovat trochu pomalejší finální přibližování ke globálnímu minimu. Opět žádný z modelů neměl výrazně navrch nebo naopak.

5.8 Hybridní funkce hf10

Tabulka 10 Výslekdy pro hybridní funkci hf10

Minimální hodnota funkce:			2000			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	2000.0	2004.1727	2000.3387	2000.0057	0.9718	6
Inv. fitness	2000.1189	2020.0755	2001.5685	2000.8374	3.5082	-3
Quartile	2000.0031	2004.6495	2001.4935	2001.3963	1.2231	-3
Inv. quartile	2000.0017	2003.5357	2000.5464	2000.0626	1.0561	3
Distrance	2000.207	2005.7969	2001.4058	2001.0197	1.1778	-3
Inv. distance	2000.0	2004.1976	2000.5105	2000.0032	1.1601	6
Linear descendig	2000.0044	2005.7846	2001.4746	2000.9519	1.463	-3
Inv. linear descending	2000.0118	2003.949	2001.4372	2001.1495	1.0216	-3

Výsledky pro hybridní funkci hf10 opět ukazují vyrovnanost váhových modelů. Všechny modely se dokázaly velmi přiblížit globálnímu minimu. Fitness model a inv. distance model se dokázaly dostat dokonce přesně na globální minimum. Tyto dva modely získaly nejlepší skóre v Mann-Whitneyově U-testu. Kladné skóre si z testu odnesl i model inv. quartile. Ostatní modely si odnesly záporné skóre. Každopádně podle výsledků testování je zcela zřejmé, že nejúčinnějšími modely jsou fitness a inv. distance.



Obrázek 15 Konvergenční graf pro hybridní funkci hf10

Opět lze vyčíst pomalejší finální přibližování se ke globálnímu minimu. Nicméně je možné pozorovat i drobné rozdíly v jednotlivých konvergenčních křivkách. Je třeba patrné, že model inverted quartile se dostával k finálním hodnotám nejdéle. Ale rozdíl není natolik markantní, aby byl více významný.

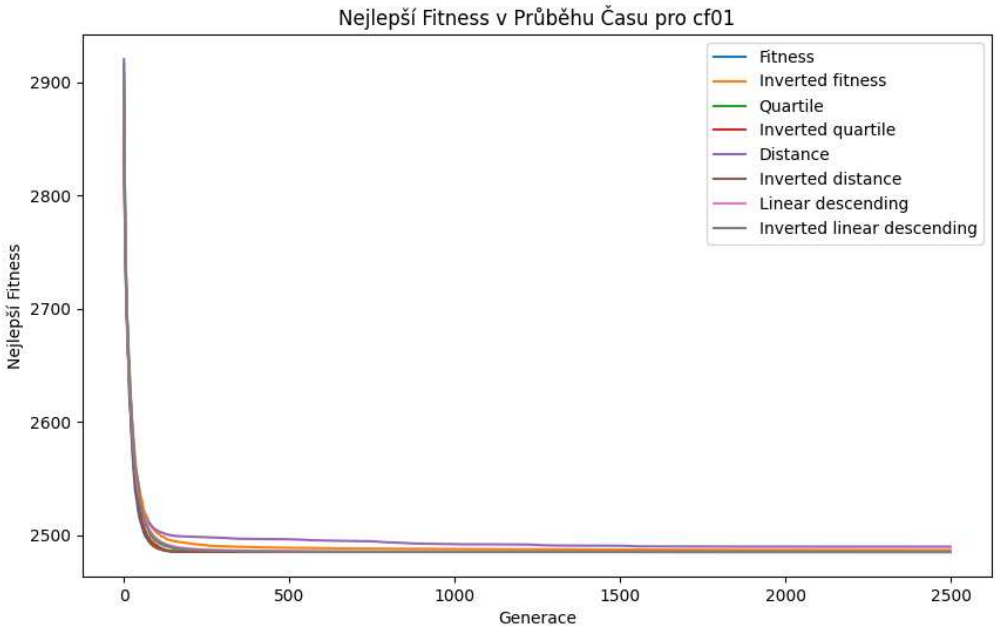
5.9 Složená funkce cf01

Tabulka 11 Výsledky pro složenou funkci cf01

Minimální hodnota funkce:			2300			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	2485.5017	2485.5017	2485.5017	2485.5017	0.0	5
Inv. fitness	2485.5017	2497.2425	2487.1482	2485.8104	2.6993	-6
Quartile	2485.5017	2485.5017	2485.5017	2485.5017	0.0	1
Inv. quartile	2485.5017	2485.5017	2485.5017	2485.5017	0.0	5
Distrance	2485.5017	2518.8323	2490.097	2486.0982	7.3767	-6
Inv. distance	2485.5017	2485.5017	2485.5017	2485.5017	0.0	5
Linear descendig	2485.5017	2485.5017	2485.5017	2485.5017	0.0	-2
Inv. linear descending	2485.5017	2485.5017	2485.5017	2485.5017	0.0	-2

V dokumentaci k testovací sadě je uvedeno, že minimální hodnota funkce je 2300. Nicméně ze statistik vyplývá, že žádný váhový model nedosáhl hodnoty menší než 2485.5017. Vzhledem k počtu modelů a k tomu, kolikrát byl algoritmus prováděn, lze předpokládat, že skutečná minimální hodnota je skutečně 2485.5017, protože pravděpodobnost uváznutí v lokálním minimu je velmi malá.

Všechny váhové modely dosáhly stejných výsledků, kromě modelů inv. fitness a distance, které měly problémy dosáhnout globálního minima ve všech 30 bězích. Rozdíly se nachází ve výsledcích Mann-Whitneyho U-testu. Nejlepší skóre získaly modely fitness, inv. quartile a inv. distance. Tyto modely jsou si zcela rovny. Nejhorší skóre získaly modely inv. fitness a distance. Tyto modely výrazně zaostaly v testování za prvními dvěma modely.



Obrázek 16 Konvergenční graf pro složenou funkci cf01

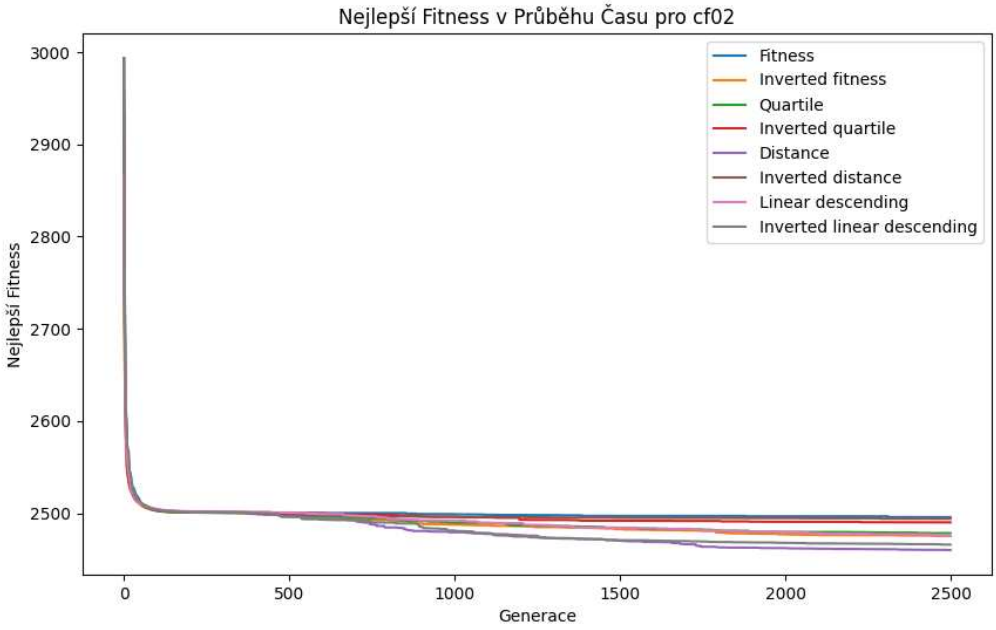
U této funkce se opět všechny modely dostaly k hodnotám blízkým globálnímu minimu během prvních iterací algoritmu. Je ovšem patrné, že distance model se přibližoval ze všech modelů nejdéle. Stejně tak model inv. fitness se k finálním hodnotám přibližoval pomaleji než ostatní. Ale opět rozdíly ve křivkách nejsou nikterak výrazné.

5.10 Složená funkce cf02

Tabulka 12 Výsledky pro složenou funkci cf02

Minimální hodnota funkce:			2400			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	2415.7808	2500.3794	2495.9162	2500.2777	17.064	1
Inv. fitness	2404.518	2501.3769	2475.3877	2500.3269	36.8136	0
Quartile	2004.1829	2500.8247	2478.5551	2500.3157	33.0456	0
Inv. quartile	2411.2373	2511.4934	2490.3124	2500.3391	25.804	-1
Distrance	2404.0919	2511.4961	2460.2551	2468.797	41.1085	0
Inv. distance	2404.3487	2501.2239	2494.1905	2500.2968	23.0068	0
Linear descendig	2406.6119	2501.575	2475.6671	2500.3692	34.7662	0
Inv. linear descending	2401.6103	2501.1031	2466.1761	2500.3338	41.7593	0

Opět byly zaznamenány velmi vyrovnané výsledky, tentokrát u složené funkce cf02. Fitness model opět vynikl v Mann-Whitneyově U-testu. Nicméně statistiky ukazují, že nejblíže ke globálnímu minimu se dostal model inv. distance. Konzistentně nejlepší výsledky podle průměru však podávaly modely distance a inv. linear descending, u kterého se váhy lineárně snižují a nejvyšší váha je přidělena jedinci s nejhorší hodnotou fitness. Statisticky nejhorší průměr byl dosažen modelem fitness, i když tento model vykázal nejnižší směrodatnou odchylku a nejlepší výsledek testování ze všech. Naopak nejvyšší směrodatnou odchylku měl model inv. linear descending. Ani jeden model se však nedostal až na globální minimum, což vyvolává otázku, zda je to způsobeno složitostí funkce, nebo tím, že ani jeden z vybraných modelů není zcela vhodný pro tuto funkci.



Obrázek 17 Konvergenční graf pro složenou funkci cf02

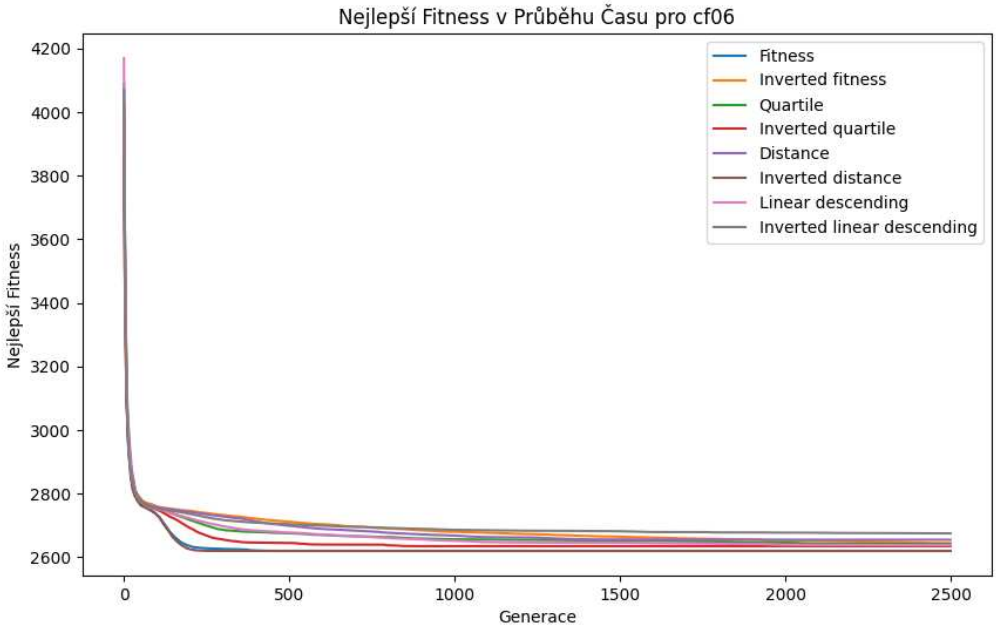
Tento konvergenční graf nabízí zajímavý pohled na rozdíly mezi jednotlivými modely. Ze začátku byly všechny modely vyrovnané, ale po přibližně 500 iteracích se začaly lišit. Zatímco distance model a inv. linear descending model se postupně začaly přibližovat ke globálnímu minimu, ostatní modely buď pokračovaly ve stagnaci v nějakém lokálním minimu nebo se přibližovaly ke globálnímu minimu pomaleji. Je tedy zřejmé, že modely distance a inv. linear descending jsou pro tuto funkci vhodnější než ostatní.

5.11 Složená funkce cf06

Tabulka 13 Výsledky pro složenou funkci f06

Minimální hodnota funkce:			2600			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	2600.0	2750.4272	2620.057	2600.0	51.1354	5
Inv. fitness	2600.0	2751.077	2644.7507	2611.1734	57.4318	-5
Quartile	2600.0	2750.8734	2641.3537	2600.0	64.5283	1
Inv. quartile	2600.0	2750.4272	2635.0997	2600.0	63.6236	3
Distrance	2600.0	2751.9113	2655.189	2601.0115	69.4498	-5
Inv. distance	2600.0	2751.0751	2620.0786	2600.0	51.1906	5
Linear descendig	2600.0	2750.5256	2638.0913	2600.0	63.8677	0
Inv. linear descending	2600.0	2751.7525	2675.1424	2687.5504	71.9303	-4

U této funkce se všechny váhové modely dokázaly dostat až na globální minimum. Vzhledem ke komplexnosti funkce však všechny modely občas uvízly v lokálním minimu. Z Mann-Whitneyova U-testu si nejlepší skóre odnesly fitness model a inv. distance model, které mají velmi podobné statistiky a jsou tedy pro tuto funkci podobně účinné. Naopak nejhorší výsledky vykazují modely inverzní k těmto dvěma, které se ale mírně liší ve statistikách. Zatímco model inv. fitness má lepší statistiky průměru a směrodatné odchylky, model distance má lepší medián. Zbylé statistiky jsou opět velmi podobné. Špatné výsledky má i model inv. linear descending, který sice v testování získal mírně lepší skóre, ale ve všech zásadních statistikách zaostává. Tyto tři modely můžeme považovat za neúčinné.



Obrázek 18 Konvergenční grap pro složenou funkci cf06

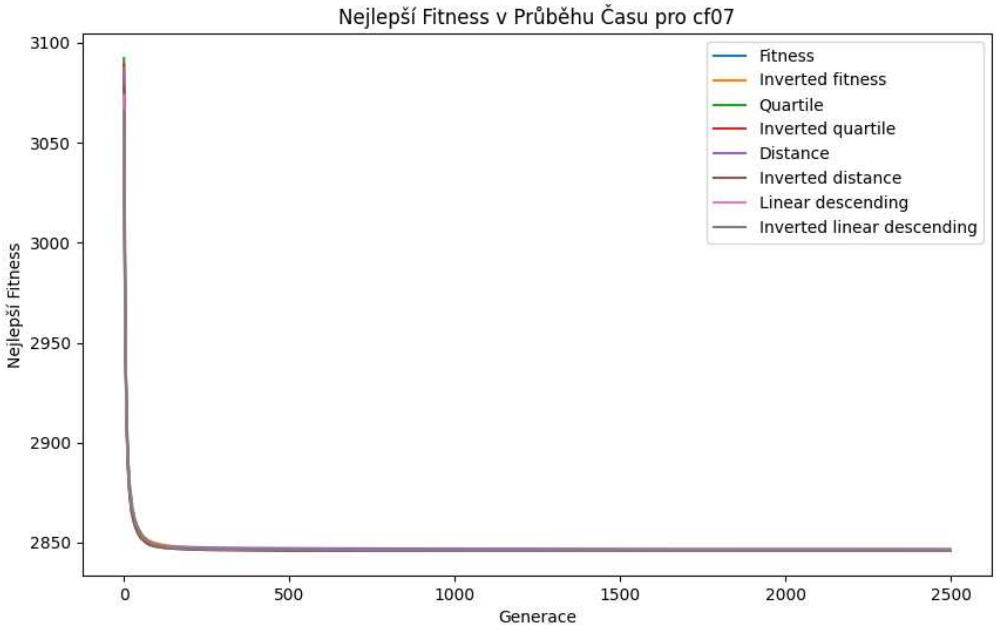
Konvergenční graf opět nabízí zajímavý pohled na vývoj průměru nejlepších fitness hodnot během všech 30 běhů. Všechny modely se na několik iterací zasekly v lokálním minimu, ale postupně se z něj dokázaly dostat. Nejlépe si v tom vedly modely fitness a inv. distance, které zároveň dosáhly nejlepších výsledků, jak je patrné z tabulky výše. Ostatní modely se z daného lokálního extrému dostávaly výrazně pomaleji. Zároveň je možné pozorovat, že model inv. linear descending za ostatními modely zaostával ve finálních iteracích.

5.12 Složená funkce cf07

Tabulka 14 Výsledky pro složenou funkci cf07

Minimální hodnota funkce:			2700			
Váhový model	Min	Max	Průměr	Medián	Směrodatná odchylka	Skóre
Fitness	2845.2954	2849.6357	2845.9443	2845.8824	0.7505	5
Inv. fitness	2845.4969	2848.3175	2846.3939	2846.1104	0.6908	-2
Quartile	2845.4536	2847.3183	2846.0487	2846.052	0.348	-1
Inv. quartile	2845.2954	2846.7075	2845.8283	2845.8061	0.3471	5
Distrance	2845.7869	2852.345	2846.7942	2846.4066	1.2382	-7
Inv. distance	2845.2954	2846.5138	2845.905	2845.9777	0.3533	2
Linear descendig	2845.2972	2848.5068	2846.1357	2846.0583	0.6493	-1
Inv. linear descending	2845.4531	2848.056	2846.1111	2846.0134	0.5351	-1

Vzhledem k minimálním hodnotám výsledků lze předpokládat, že v dokumentaci k testovací sadě je nesprávně uvedeno globální minimum. Všechny váhové modely dospěly k velmi podobným výsledkům s minimálními statistickými rozdíly. Z Mann-Whitneyova U-testu vyšly nejlépe modely fitness a inv. quartile. Naopak nejhorší výsledek měl distance model, který měl navíc i nejhorší statistiky. Lze předpokládat, že tento model je pro tuto funkci nejméně účinný. Ostatní modely dosáhly velice podobných výsledků a rozdíly mezi nimi jsou minimální. Zároveň jsou tyto výsledky velice blízké modelům s nejlepším skóre.



Obrázek 19 Konvergenční graf pro složenou funkci cf07

Graf opět poukazuje na vyrovnanost všech modelů. Žádný z modelů nevykazuje problémy s přiblížením ke globálnímu minimu a křivky všech modelů se téměř překrývají.

6 VYHODNOCENÍ VÝSLEDKŮ

Tabulka 15 Celkové skóre z Mann-Whitneyho U-testu

Model	Skóre
Fitness	45
Inv. distance	41
Inv. quartile	33
Quartile	3
Linear descending	-14
Inv. linear descending	-18
Inv. fitness	-30
Distance	-46

Obecně byly výsledky, až na výjimky, na první pohled velmi vyrovnané a těsné. Zdá se, že všechny váhové modely byly vzájemně srovnatelné. Výraznější rozdíly se začaly projevovat až u složitějších složených funkcí, u kterých měly některé modely problémy. Toto lze vyčíst jak z tabulek statistických údajů, tak i z konvergenčních grafů. Statistické výsledky pro jednotlivé váhové modely se u většiny funkcí lišily jen minimálně. Nebylo tedy možné s jistotou určit, který váhový model je lepší nebo horší.

Výsledky Mann-Whitneyova U-testu byly mnohem více vypovídající. Podle těchto výsledků bylo možné určit, který váhový model je u jednotlivých funkcí lepší. Pro závěrečné vyhodnocení výsledků jsem vytvořil tabulku, která obsahuje celkové skóre jednotlivých modelů. Z této tabulky je zřejmé, že nejlepší výsledky podával fitness model, který je založen na zlepšení fitness. Naopak nejhorší výsledky podával distance model, založený na vzdálenosti jedinců od referenční hodnoty fitness, kde menší vzdálenost vždy znamenala vyšší váhu.

Inverzní distance model byl druhý nejlepší. Tento model přiděloval vyšší váhu jedincům, kteří byli nejvzdálenější. Díky tomu byl algoritmus schopný prozkoumat větší oblast, a tak lépe najít globální minimum. Z teoretického hlediska je to velice zajímavý model, který může mít velký potenciál pro další využití.

Kvartilový model dosáhl prakticky neutrálních výsledků. Tento model rozdělil jedince do čtyř kvartilů podle zlepšení fitness. Jedinci, kteří se dostali do kvartilu s nejmenším zlepšením, získali nejvyšší váhu. Byl zde předpoklad, že jsou blízko ke globálnímu minimu. Vzhledem k celkovému skóre je možné konstatovat, že tento model není vhodný, ale zároveň není zcela neúčinný pro optimalizaci.

Inverzní kvartilový model dosáhl třetího nejlepšího výsledku. Nejvyšší váhu v tomto případě dostali jedinci, jejichž zlepšení bylo největší. Tento model je tak vhodnější než většina ostatních testovaných modelů a může představovat zajímavou variantu pro použití při optimalizaci pomocí diferenciální evoluce. Vzhledem k trendu, že nejúspěšnější modely přiřazují váhy největšímu zlepšení hodnoty fitness, není překvapením, že inverzní model fitness má druhé nejhorší skóre. Tento model tak není příliš vhodný pro další použití. Modely založené na lineárním snižování vah skončily oba se záporným skóre, což ukazuje, že ani jeden z těchto modelů není zcela vhodný pro optimalizaci.

Celkově, jak již bylo zmíněno, byly výsledky testování na první pohled velmi vyrovnané. U většiny funkcí, na kterých byly modely testovány, byl nejzásadnější rozdíl ve skóre získaném z Mann-Whitneyova U-testu. Vzhledem k tomu, že ale i sebemenší rozdíly ve výsledcích mohou mít obrovský význam, je tato statistika nejzásadnější. Ostatní statistiky se vzájemně nelišily natolik, aby z nich bylo možné s jistotou něco vyčíst. Testování však hodnotilo výsledky experimentů i na základě drobností, které většinou rozhodly o výsledném skóre. Pro hlubší pochopení toho, jak různé nastavení vah ovlivňuje chování adaptivního algoritmu diferenciální evoluce, je nutné provést další testy s různými testovacími funkcemi a modely vah. Zároveň by bylo vhodné provádět testy na více dimenzích. Nicméně na základě mého testování a poznatků jsem usoudil, že způsob výpočtu vah může mít velmi zásadní vliv na výsledky, obzvláště v situacích, kde může hrát významnou roli i zlepšení řádově o tisíciny nebo i o ještě menší řády. Způsob výpočtu a přidělování vah může opravdu zásadně ovlivnit výsledky optimalizace. Samozřejmě zde existuje mnoho jiných faktorů, které celý proces optimalizace ovlivňují.

ZÁVĚR

Diplomová práce se zaměřuje na analýzu vlivu adaptivních vah v procesu diferenciální evoluce s cílem zjistit, jak různé modely těchto vah ovlivňují efektivitu hledání globálních extrémů v optimalizačních problémech. Práce je strukturována do několika klíčových částí, které zahrnují teoretický úvod, metodologický rámec, praktické experimenty a vyhodnocení výsledků, přičemž každá část přispívá k celkovému porozumění zkoumané problematiky.

V teoretické části je poskytnut přehled základních principů diferenciální evoluce, včetně detailního popisu jejích klíčových operací, jako jsou mutace, křížení a selekce. Dále jsou zde představeny adaptivní strategie jako SHADE (Success-History Adaptation for Differential Evolution) a jDE (Self-Adaptive Differential Evolution), které umožňují dynamickou adaptaci kontrolních parametrů algoritmu v průběhu evolučního procesu. Tyto metody jsou zásadní pro pochopení, jak lze využít historii výkonu k efektivnějšímu prozkoumání a využití vyhledávacího prostoru.

Metodologická část popisuje proces vývoje a implementace různých modelů adaptivních vah, jejich integraci do algoritmu diferenciální evoluce a nastavení experimentálního testování. Byly zvoleny různé optimalizační funkce pro testování efektivity navržených adaptivních modelů, což umožnilo objektivní srovnání jejich výkonnosti.

V praktické části bylo provedeno několik experimentů, jejichž cílem bylo otestovat účinnost adaptivních váhových modelů v diferenciální evoluci. Výsledky ukázaly, že některé modely vah dokážou výrazně zlepšit rychlost konvergence algoritmu a celkovou kvalitu nalezených řešení, zatímco jiné modely přinesly pouze marginální zlepšení. Bylo zjištěno, že úspěšnost konkrétního váhového modelu může být výrazně ovlivněna specifickými charakteristikami testované optimalizační funkce, což naznačuje potřebu dalšího výzkumu pro lepší pochopení a optimalizaci těchto modelů.

Z výsledků testování vyplynulo, že způsob výpočtu vah může mít zásadní vliv na proces diferenciální evoluce. Z prvotního pohledu na statistiky se může zdát, že rozdíly ve výsledcích jsou minimální a nemají větší význam. Ovšem i na základě zdánlivě drobných rozdílů dokázal Mann-Whitneyho U-test lépe osvětlit výsledky. Toto testování posuzovalo výsledky jednotlivých modelů a přiřazovalo finální skóre. Na základě tohoto testování lze konstatovat, že nejlépe si vedl fitness model. Tento model přiřadil nejvyšší váhu jedincům, kteří měli nejlepší zlepšení hodnoty fitness. Naopak nejhorší výsledek testování měl model

distance. Tento model přiřazoval nevyšší váhy jedincům, jejich hodnota fitness byla nejblíže k referenční hodnotě. Výsledky jsou podrobně popsány v závěrečné kapitole.

V závěrečné části práce je zároveň diskutováno o důležitosti adaptivních vah v kontextu zvyšování efektivity evolučních algoritmů. Je zde zdůrazněna potřeba dalšího výzkumu v této oblasti, zejména prohloubení poznatků o dynamice adaptivních vah a jejich aplikace v širším spektru optimalizačních scénářů. Dále bylo navrženo, aby případné další testování se zaměřilo na více váhových modelů a testovacích funkcí. Zároveň by bylo vhodné provádět testování na problémech s více dimenzemi. Takové testování by přineslo mnohem více směrodatných výsledků.

SEZNAM POUŽITÉ LITERATURY

[1] PRICE, Kenneth, Rainer M STORN a Jouni A LAMPINEN, 2005. Differential Evolution: A Practical Approach to Global Optimization [online]. Berlin, Heidelberg: Springer Berlin Heidelberg [cit. 2024-02-26]. Natural Computing Series. ISBN 978-3-540-31306-9.

[2] TANABE, Ryoji a Alex FUKUNAGA, 2013. Success-history based parameter adaptation for Differential Evolution. 2013 IEEE Congress on Evolutionary Computation [online]. IEEE, 71-78 [cit. 2024-02-26]. ISBN 978-1-4799-0454-9. Dostupné z: doi:10.1109/CEC.2013.6557555.

[3] ZELINKA, Ivan, 2009. Evoluční výpočetní techniky: principy a aplikace. Praha: BEN - technická literatura. ISBN 978-80-7300-218-3.

[4] ZELINKA, Ivan, 2002. Umělá inteligence: v problémech globální optimalizace. Praha: BEN - technická literatura. ISBN 80-730-0069-5.

[5] Chakraborty, S., Saha, A.K., Ezugwu, A.E. et al. Differential Evolution and Its Applications in Image Processing Problems: A Comprehensive Review. Arch Computat Methods Eng 30, 985–1040 (2023). https://doi.org/10.1007/s11831-022-09825-5

[6] Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. IEEE Transactions on evolutionary computation, 13(5), 945-958.

[7] VIKTORIN, Adam; SENKERIK, Roman; PLUHACEK, Michal; KADAVY, Tomas a ZAMUDA, Ales. Distance based parameter adaptation for Success-History based Differential Evolution. Online. Swarm and Evolutionary Computation. 2019, roč. 2019, č. 50, article 100462. ISSN 2210-6502. Dostupné z: https://doi.org/10.1016/j.swevo.2018.10.013.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

DE	Diferenciální evoluce
F	Mutační konstanta
CR	Práh křížení
NP	Velikost populace
G_{Max}	Maximální počet generací
D	Počet argumentů/dimenzí
jDE	DE s propagací řídících parametrů
EPSDE	DE se sadou řídicích parametrů a strategií
JADE	Adaptive differential evolution with optional external archive
PSO	Particle swarm optimization
SaDE	Self-adaptive Differential Evolution
SHADE	Success-History Based Parameter Adaptation for Differential Evolution

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH