Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Algoritmus Particle Swarm Optimization

Martin Klabeneš

Bakalářská práce
2019

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2018/2019**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Martin Klabeneš**

Osobní číslo: **I16101**

Studijní program: **B2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Algoritmus Particle Swarm Optimization**

Zadávající katedra: **Katedra informačních technologií**

Zásady pro vypracování:

V optimalizačních úlohách lze jen zřídka najít řešení analyticky a je tedy často nutné užít algoritmů stochastických, které sice nemohou garantovat nalezení řešení v konečném počtu kroků, ale často pomohou nalézt v přijatelném čase řešení prakticky použitelné. K hledání aproximace optima bylo vyvinuto mnoho přibližných metod. Některé z nich jsou biologicky inspirovány. Cílem této práce bude prostudovat algoritmus Particle Swarm Optimization (PSO), který k nalezení řešení využívá analogie s chováním ptáků. Cílem práce bude sestavení programu umožňujícího hledání odhadu minima tzv. testovacích funkcí (např. Griewangkova funkce, Rastrigova funkce, Schwefelova funkce), které byly navrženy pro testování kvality heuristických algoritmů. V práci budou popsány veškeré aplikované metody a bude vysvětleno uživatelské prostředí aplikace.

Rozsah grafických prací: **10**

Rozsah pracovní zprávy: **35**

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

Seznam odborné literatury:

KENNEDY, J. a R. EBERHART. Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks IV., 1995, pp. 1942-1948. doi:10.1109/ICNN.1995.488968
ENGELBRECHT Andries P. Computational Intelligence: An Introduction. Chichester: John Wiley & Sons, 2007. ISBN-13: 978-0470035610

Vedoucí bakalářské práce: **Mgr. Jaroslav Marek, Ph.D.**
Katedra matematiky a fyziky

Datum zadání bakalářské práce: **31. října 2018**

Termín odevzdání bakalářské práce: **12. května 2019**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 20. března 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10.05.2019

Martin Klabeneš

PODĚKOVÁNÍ

Chtěl bych poděkovat Mgr. Jaroslavu Markovi, Ph.D. za příkladné vedení bakalářské práce, poskytnuté materiály a odborné rady ohledně algoritmu PSO.

ANOTACE

Práce je věnována problematice výpočtu globálního extrému funkce více proměnných. Problém není vždy řešitelný analyticky matematickými postupy a často se musíme spokojit pouze s přibližným řešením získaným na počítači pomocí heuristických metod. K nalezení extrému bude použit algoritmus Particle Swarm Optimization. Kroky tohoto algoritmu jsou inspirovány chováním ptáků. Hlavním cílem práce je vytvořit program pro výpočet extrému pomocí algoritmu PSO.

KLÍČOVÁ SLOVA

Particle Swarm Optimization, PSO, optimalizační algoritmus, optimalizace, extrém.

TITLE

Algorithm Particle Swarm Optimization

ANNOTATION

The thesis is focused on searching an approximate minimum of functions, for which it is not possible to find a minimal analytical path. The Particle Swarm Optimization algorithm, inspired by the analogy of bird behaviour, is used to find the minimum. The main goal of the thesis is to create a program that can be used for calculating the extreme using the PSO algorithm.

KEYWORDS

Particle Swarm Optimization, PSO, optimization algorithm, biological algorithms, optimization, extreme.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

PSO	Particle Swarm Optimization
pBest	Particle Best
gBest	Global Best

ÚVOD

Hledáním extrémů funkce se věnuje část matematické analýzy – diferenciální počet. Pomocí parciálních derivací prvního řádu můžeme najít body podezřelé z extrému a s využitím druhých parciálních můžeme rozhodnout o typu extrému. To je ale možné jen pokud derivace funkce existují a současně dokážeme vyřešit soustavu rovnic pro určení kritických bodů. Často ale takovým postupem získáme soustavu nelineárních rovnic, která nemá analytické řešení. Algebra nám nabízí jen vzorce pro výpočet kořenů polynomu třetího stupně nebo čtvrtého stupně. Francouzský matematik Galoise dokázal, že u polynomu pátého a vyššího stupně již nelze kořen analytickými postupy najít. Mnoho matematických úloh musíme tedy řešit jen s využitím numerických metod. Problémy hledání minima resp. maxima funkce se označují jako optimalizační úlohy. K řešení se používají často heuristické metody, které nám poskytnou přibližné řešení. Tyto heuristiky jsou často založeny na různých evolučních přístupech. Vzniklo mnoho metod, jejichž kroky jsou inspirovány chování různých živočichů. Ti se vyznačují a někdy lze konstatovat, že přežívají díky určitému chování. Pro přežití je nejdůležitější zejména zákon „přežití nejsilnějšího“. Slabší jedinci se pro přežití museli shlukovat do hejn, smeček nebo jiných skupin jedinců. Na principu křížení a mutace jsou založeny genetické algoritmy.

Efektivní metodou pro nalezení extrému funkce v přijatelném čase je metoda optimalizace hejnem částic neboli Particle Swarm Optimization. PSO byl poprvé popsán v roce 1995 dvěma autory Jamesem Kennedym a Russellem Eberhartem. Tato bakalářská práce vychází především z jejich práce [1].

Základní vyhledávací prvky algoritmu PSO jsou částice. Každá částice je definována svou polohou, rychlostí a doposud nejlepší nalezenou pozicí. [6]

Cílem této práce je popsat základní principy a kroky v algoritmu PSO a vytvořit software, který pomocí algoritmu PSO nalezne přibližnou hodnotu extrému u několika testovacích funkcí.

1 PARTICLE SWARM OPTIMIZATION

Optimalizační algoritmus PSO je založen především na inteligenci a chování hejna. Původním motivem pro simulaci bylo modelovat společenské chování lidí, což není na tomto principu možné. Důležitá je abstrakce. Je nutno však poukázat, že sice není možné na tomto principu společensky interagovat, ale je možné nechat se inspirovat vrstevníky v našich postojích a názorech. Hlavní rozdíl mezi lidmi a ptáky je, že dva lidé mohou mít stejné názory, ale dva ptáci nemohou být na stejném místě (problém kolize). [1]

Pro tři a méně rozměrné dimenze je nutno zabraňovat kolizím. Ve více než trojrozměrném prostoru jsou operace bez kolize. [1]

Inteligence hejna je založena na decentralizovaném chování jednotlivých jedinců, kteří na sebe vzájemně reagují. Každý jedinec se řídí jednoduchými, předem danými pravidly. I když není předem předepsáno, jak se jedinci mají chovat, vzájemná interakce mezi nimi způsobuje složité komplexní chování systému jako celku. [8]

1.1 Začátek PSO

První verze PSO pracovaly se zjednodušeným modelem společenského hejna. Ptáci byli označováni jako agenti. Původním záměrem bylo simulovat plynulý pohyb agenta, který by ale zůstal nepředvídatelným. Pro každého agenta byly důležité dvě složky: rychlost nejbližšího souseda a tzv. „šílenství“. Pozice agentů [X, Y] a jejich rychlosti v byly náhodně vybrány. Pohyby agentů jsou ovlivněny nejbližším sousedem (sousedy). Skok agenta na novou pozici je vytvořen obdobně u sousedních agentů, podobná je jejich délka skoku i směr. Bohužel tímto způsobem se hejno rychle usadilo a přestalo měnit směr a délku skoku. Proto se zavedl pojem „šílenství“. Je to přidání náhodné hodnoty pro skok (délka, směr). Po zavedení „šílenství“ je simulace mnohem živější a zajímavější. Tento jev je však vytvořen uměle a nemá skutečnou analogii v chování ptáků. [1]

Studie F. Hybnera a U. Grenandera [2] vnesla do algoritmu nový poznatek. Bylo zjištěno, že ptáci po nějaké době začnou kroužit okolo hnízda a přistávat tam. To eliminuje nutnost přidávat do algoritmu „šílenství“. Nastává však další otázka a to ta, jak je možné, že ptáci naleznou správný směr i přes to, že nevědí, kde mají hledat cíl? Příkladem jsou holuby, kteří i přes to, že nevědí, kam mají doručit zprávu, během několika hodin se zorientují a naleznou směr. Z toho vyplývá, že členové hejna pravděpodobně využívají své znalosti orientace podle okolí pro nalezení ideálního směru. [1]

1.1.1 Studie PSO

Kroky algoritmu budeme demonstrovat na hledání globálního extrému, konkrétně minima, spojité funkce dvou nezávislých proměnných. Takové zúžení problému nám umožní grafickou prezentaci výsledků.

Nejprve zavedeme mřížku bodů – pole – vnořenou do definičního oboru funkce, jejíž globální extrém budeme hledat. Z pole náhodně vybereme několik bodů, tyto body nazveme agenti. Agenti jsou tedy určeni souřadnicemi x a y.

Pro každého agenta se vyhodnotí z jeho pozice x a y funkční hodnota zkoumané funkce f, například daná vzorcem:

$$\text{Eval} = \text{f(x,y)} = \sqrt{(\text{x} - 100)^2} + \sqrt{(\text{y} - 100)^2}\ .$$

To znamená, že agent s polohou (100,100) má hodnotu 0. Souřadnice agentů budeme postupně měnit pomocí jistých pravidel, která reflektují na průběh grafu zkoumané funkce. Každý agent si bude pamatovat svou nejmenší hodnotu funkce a odpovídající nejlepší (z pohledu minimalizačního problému) pozici x a y. Tyto hodnoty byly pojmenovány $pBest[]$ (někdy označováno jako $lBest$ – „local best“) pro nejlepší hodnotu v rámci funkce, $pBest_x[]$ a $pBest_y[]$ pro nejlepší pozice x a y. Závorky nám říkají, že se jedná o pole o velikosti počtu agentů. Jakmile se agent začne pohybovat prostorem – definičním oborem, pak jsou jeho rychlost, pozice x a pozice y upravovány. Např. pokud je agent napravo od $pBest_x[]$, potom jeho rychlost X korigujeme záporně podle vzorce:

$$v_x[] = v_x[] - rand() * p_{increment}.$$

Pokud je $pBest_x[]$ vlevo, agent by skočil v kladném směru osy x podle vzorce:

$$v_x[] = v_x[] + rand() * p_{increment}.$$

Stejné to je i u $pBest_y[]$, jen se mění vertikální směr pohybu agenta nahoru nebo dolů. $v_x[]$ a $v_y[]$ jsou rychlosti pohybu. [1]

Každý agent zná nejlepší globální hodnotu hejna, kterou již našel jakýkoliv jiný agent. Toho je dosaženo pomocí přidání indexu pro nejlepší hodnotu. Index se nazývá $gBest$. Např. nejlepší hodnota pro pozici x je $pBest_x[gBest]$. Analogicky zavádíme obdobnou veličinu i pro pozici y. Informace o $gBest$ je dostupný pro všechny členy hejna. Skoky $v_x[]$ a $v_y[]$ jsou počítány následovně.[1]

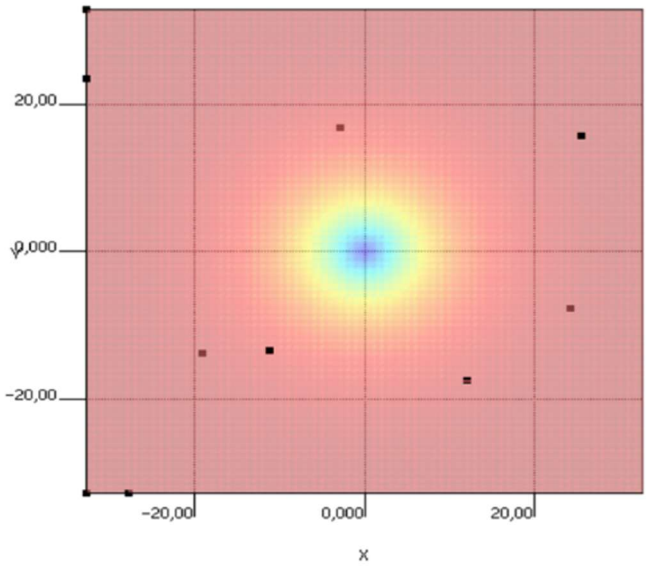
- Pokud $present_x[] > pBestx[gBest]$, potom $v_x[] = v_x[] - rand() * g_{increment}$,

- pokud $present_x[] < pBestx[gBest]$, potom $v_x[] = v_x[] + rand() * g_{increment}$,
- pokud $present_y[] > pBestx[gBest]$, potom $v_y[] = v_y[] - rand() * g_{increment}$,
- pokud $present_y[] < pBestx[gBest]$, potom $v_y[] = v_y[] + rand() * g_{increment}$.

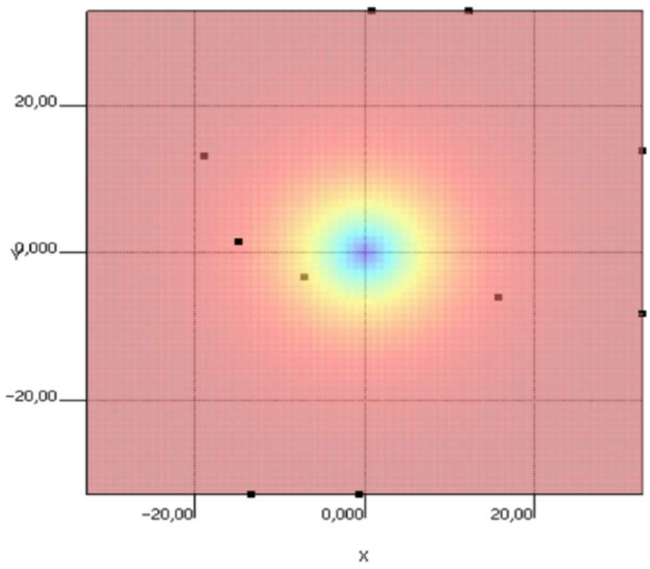
kde $g_{increment}$ je akcelerační koeficient (následně budou tyto konstanty označovány jako c_1 a c_2).

Pozorovatel mohl sledovat, jak se agenti pohybují, dokud nenajdou námi hledaný bod (představující např. potravu). Výsledky byly překvapivé. Pokud jsou parametry $g_{increment}$ a $p_{increment}$ velké, hejno se velmi rychle dostávalo k hledanému bodu v několika málo iteracích. Celé hejno (15 – 30 jedinců) obklopilo námi hledaný bod a kroužilo okolo něj. S nízkými parametry se hejno dostane k cíli s určitou rytmikou a nakonec přistává v námi hledaném bodě. [1]

1. Vysoký první koeficient tj. $c_1 = 2$. Koeficient c_2 zůstává nezměněn.

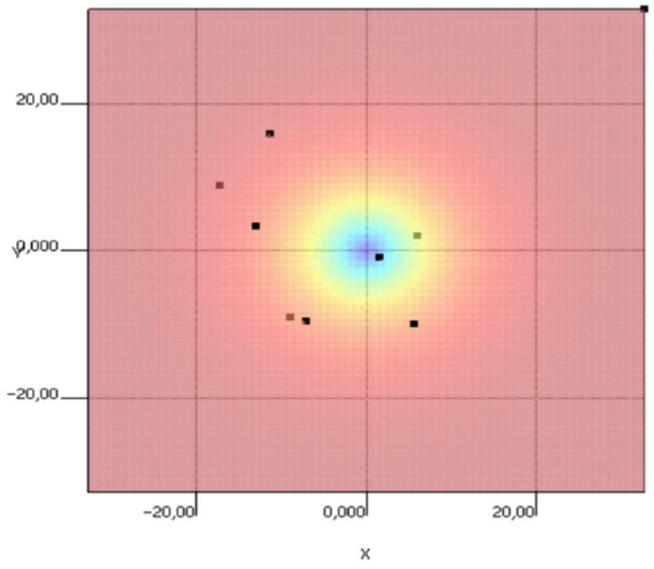


Obrázek 1: 1.iterace s vysokým c_1

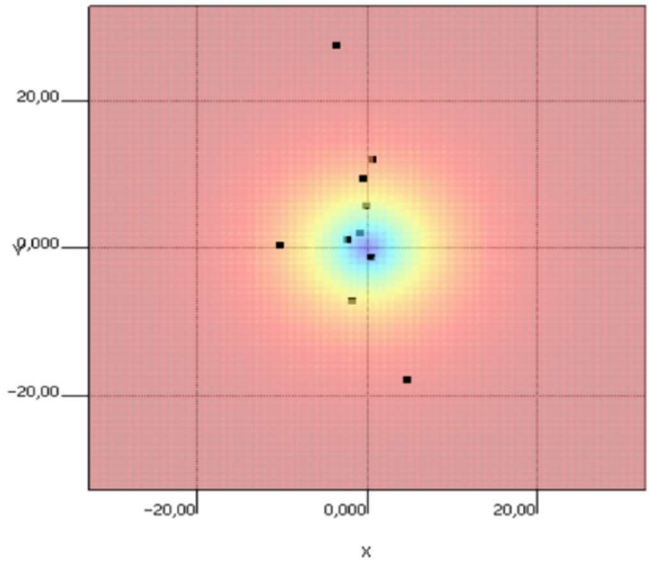


Obrázek 2: 20.iterace s vysokým c_1

2. Vysoký druhý koeficient, tj. $c_2 = 2$. Koeficient c_2 zůstává nezměněn.



Obrázek 3: 1.iterace s vysokým c_2



Obrázek 4: 20.iterace s vysokým c_2

1.2 Pomocné proměnné

1.2.1 Odstranění pomocných proměnných

Autoři algoritmu zjistili, že v algoritmu není nutné mít proměnnou „šílenství“. Algoritmus sice ztratil živost, kterou tato proměnná dávala, ale na průběh optimalizace funkce to nemělo vliv. [1]

Dále bylo zjištěno, že k optimu iterujeme o něco rychleji, pokud je odstraněna rychlost nejbližšího souseda. To změní celkový visuální efekt. Jednotlivci hejna se pohybují náhodně, ale na výslednou schopnost konvergence k cíli to nemá vliv. [1]

1.2.2 Důležité proměnné

1. Proměnná $pBest$ – proměnnou $pBest$ můžeme nazvat jako paměť agenta, jelikož každý agent si pamatuje svou vlastní pozici. Tato pozice vychází ze zkušeností konkrétního jedince. Jedinec má tendenci vracet se na místo, které ho v minulosti zaujmulo – v našem případě pozice $pBest$.

2. Proměnná $gBest$ – proměnnou $gBest$ můžeme nazvat jako globální paměť agentů. Zároveň je to hodnota, kterou se snaží jedinec dosáhnout.

3. Hodnota $p_{increment}$ – první akcelerační koeficient, který se v následných pracích zapisuje jako c_1. Vzorec akceleračního koeficientu je dle [1]:

$$0{,}5 + log(2) \cong 1{,}193.$$

4. Hodnota $g_{increment}$ – druhý akcelerační koeficient, který se v následujících pracích označuje jako c_2. Vzorec akceleračního koeficientu je podle [1]:

$$\frac{1}{2\log(2)} \cong 0{,}721.$$

Hodnoty $p_{increment}$ a $g_{increment}$ jsou velice důležité v rychlosti konvergence a nalezení optima. Viz explorace a exploitace [1].

1.3 Úprava rychlosti

Úprava rychlosti je založena pouze na porovnávání – pokud je $present_x > Best_x$, zmenši $present_x$. Pokud je $present_x < Best_x$, zvětši $present_x$. Experimenty dále odhalili, že pro zlepšení výkonu algoritmu je lepší měnit rychlost podle pozice od cíle.

$$v_{xy}[][] = v_{xy}[][] + rand() * p_{increment} * (pBest_x[][] - present_x[][]).$$

Parametry v_x a present_x mají dvě závorky, protože to jsou matice agentů podle rozměrů. [1]

1.4 Zjednodušené PSO

Vzhledem k tomu, že hodnoty akceleračních konstant $p_{increment}$ nebo $g_{increment}$ byly zjišťovány experimentálně a jejich přesná hodnota není ke správně funkci algoritmu třeba, byla snaha o eliminaci těchto konstant ze vzorce. Proto tyto konstanty byly z algoritmu vyškrtnuty a nahrazeny vynásobením dvěma. Původní algoritmus byl zjednodušen a nyní vypadá takto:

$$v_x[][] = v_x[][] + 2 * rand() * (pBest_x[][] - present_x[][]) + 2 * rand() * (pBest_x[][gBest] - present_x[][]).$$

Dále byly testovány další typy algoritmu, ale výsledek se již nezlepšil. Např. metoda osadníků a průzkumníků (podobné, jako heslo „rozděl a panuj“). Průzkumníci prohledávali celou oblast a cílem osadníků byl prohledat mikroregiony. Tato metoda však nezaznamenala zlepšení oproti původnímu algoritmu. [1]

Další modifikace algoritmu spočívala v odstranění původní rychlosti a změnu vzorce na úpravu rychlosti na:

$$v_x[][] = 2 * rand() * (pBest_x[][] - present_x[][]) + 2 * rand() * (pBest_x[][gBest] - present_x[][]).$$

Tato verze se však oproti původnímu velice zhoršila a byla téměř nepoužitelná při hledání globálního optima. [1]

1.5 Model inteligence hejna

M.M. Millonas [3] vyvinul model pro inteligenci hejna. Tento model má 5 základních principů:

- populace by měla být schopna provádět časové a prostorové výpočty. (princip blízkosti),
- populace by měla reagovat na kvalitu životního prostředí (princip kvality),
- populace by neměla provádět své úkony v příliš malém okolí (princip rozmanitosti),
- populace by se měla chovat stejně při každé změně prostředí (princip stability),
- populace by měla být schopna se adaptovat na jiné prostředí, pokud to má smysl (princip adaptace).

Přičemž body 4 a 5 jsou vzájemně opačné, ale nevylučují se. [1]

Zdá se, že algoritmus dodržuje všech pět bodů. Následný termín „částice“ byl vybrán jako kompromis, protože se jedná o poměrně velké množství agentů, kteří se pohybují prostorem z části náhodně a relativně nesynchronizovaně. Tento algoritmus je extrémně jednoduchý ale však velice efektivní při hledání optima pro velké množství funkcí. Jde o biologický algoritmus založený na inteligenci hejna. Je velmi závislý na stochastických procesech. [1]

1.6 Vícedimenzionální prohledávání

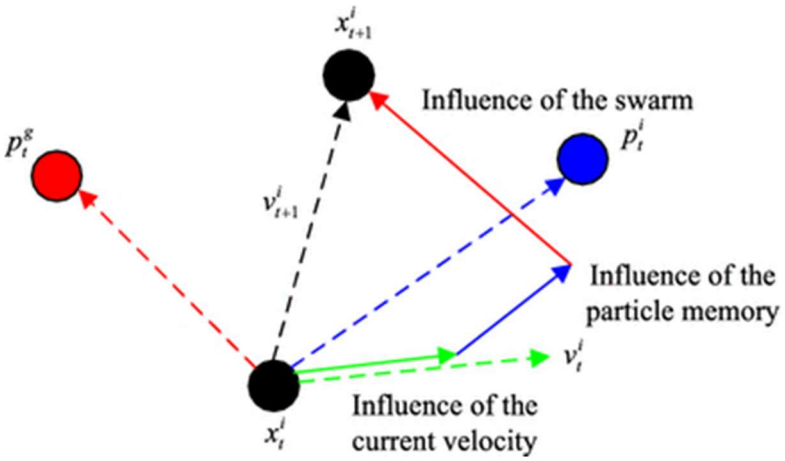
Jedním z cílů je vytvořit algoritmus pro vyhledávání ve více než dvourozměrných systémech a zároveň docílit nekolizního stavu (aby nevznikla kolize). Oproti původnímu algoritmu se změní $present_x$ a $present_y$ (samozřejmě se také změní $v_x[]$ a $v_y[]$). Pro všechny tyto hodnoty se vytvoří matice z původního jednodimenzionálního pole na D x N, přičemž D je počet dimenzí a N je počet agentů v každé dimenzi. [1]

Pro simulaci nelineárních a vícedimenzionálních polí nastal problém, bylo potřeba vytvořit vícevrstvou neuronovou sít, kterou je třeba trénovat. Byl vytvořen pokus pro třináctirozměrnou sít s uspokojivým výsledkem, neuronová sít XOR bylo natrénována během 31 iterací, dvaceti agenty. [1]

1.7 První algoritmus PSO

Každý jedinec je definován především svou pozicí a svou rychlostí. Každý jedinec si pamatuje své lokální optimum a zná také nejlepší nalezené optimum celého hejna. [9]

Pohyb jedinců, který je popsán níže, vypadá přibližně dle [10] takto:



Obrázek 5: Znázornění výpočtu skoku

celkový pohyb každého jedince je zajištěn dle vzorce:

$$x_i(t+1) = x_i(t) + v(t+1),$$

kde

- $x_i(\mathrm{t})$ je aktuální pozice agenta,
- $v(t+1)$ je pohyb agenta v následující iteraci,
- $x_i(\mathrm{t}+1)$je pozice agenta v následující iteraci.

Rychlost je následně vypočítána podle vzorce:

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1(t)\left[pBest_{ij}(t) - x_{ij}(t)\right] + c_2 r_2(t)\left[gBest_{ij}(t) - x_{ij}(t)\right],$$

kde

- v_{ij} je rychlost jedince,
- x_{ij} je aktuální pozice jedince,
- $t+1$ je následující iterace,
- t je aktuální iterace,
- c_1, c_2 jsou akcelerační koeficienty,
- r_1, r_2 jsou náhodně vygenerovaná čísla v intervalu $\langle 0, 1 \rangle$,
- $pBest_{ij}$ je dosud nejlepší nalezená pozice konkrétního jedince,
- $gBest_{ij}$ je dosud nejlepší nalezená pozice celého hejna. [9]

1.8 Popis vyhledávání pomocí PSO

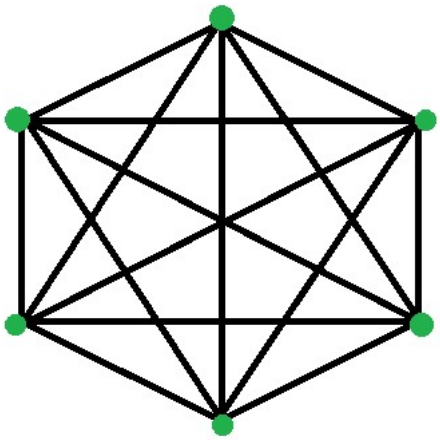
1. Inicializace pole částic. Pozice každé částice je zcela náhodná. Kontrolována je pouze pozice ve vyhledávacím prostoru.
2. Pro každou částici je vypočítána její hodnota na její aktuální pozici.
3. Porovnání aktuální hodnoty každé částice s $pBest$. Pokud je současná hodnota lepší než původní hodnota, současná hodnota je uložena do $pBest$ a zároveň je uložena i její pozice.
4. Aktualizace rychlosti.
5. Opakování kroku 2 – 5, dokud nejsou splněny podmínky ukončení.[8]

1.9 Varianty PSO

Z důvodu pokusu o zrychlení konvergence existují modifikace algoritmu PSO. Hlavní dvě modifikace se nazývají GBEST Model a PBEST Model. [8]

1.9.1 GBEST Model

Model GBEST je založen na globálním nejlepším nalezeným řešením celého hejna. Všechny částice jsou přitahovány tímto nejlepším nalezeným řešením. GBEST Model si lze představit jako zcela propojenou síť, kde všechny částice mají přístup ke všem informacím. U tohoto modelu je vysoká rychlost konvergence. Problém je, že s vysokou rychlostí konvergence je vysoká možnost uváznutí v lokálním řešení. [8]

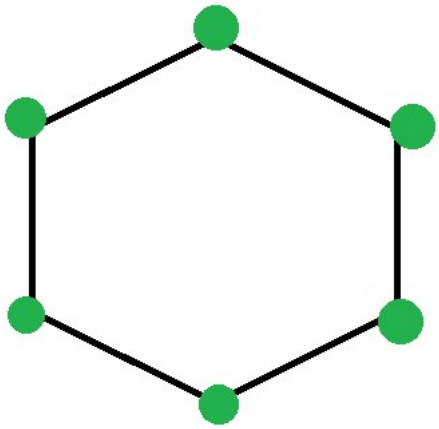


Obrázek 6: GBEST Model

1.9.2 PBEST Model

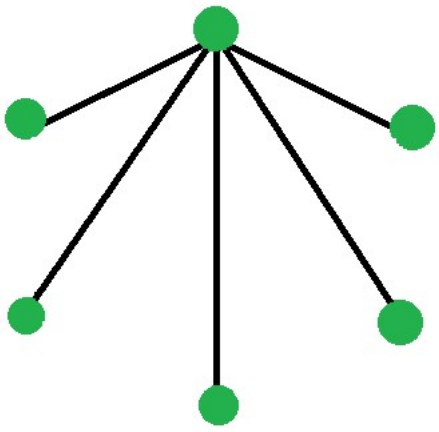
Model PBEST je založen na nejlepším nalezeném řešení pouze nejbližších jedinců. Pro toto řešení existují dvě varianty. Tento model je co do konvergence pomalejší než GBEST Model, ale je nižší pravděpodobnost, že simulace sklouzne do lokálního optima. [8]

1. Kruhovitá topologie – každá částice je propojena s dvěma nejbližšími sousedy, kteří s danou částicí sdílejí své informace o nejlepší nalezené pozici. [8]



Obrázek 7: PBEST Model - kruhovitá topologie

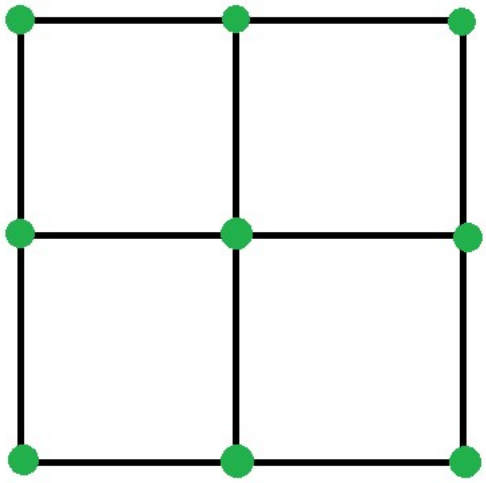
2. Centralizovaná topologie – částice jsou odděleny od ostatních. Všechny informace o nalezených hodnotách a pozicích jsou soustředěny na hlavního jedince. [8]



Obrázek 8: PBEST Model - centralizovaná topologie

1.9.3 Von Neumannův model

U Von Neumannova modelu se předpokládá relativně rychlá konvergence a zároveň snižuje šance na uváznutí v lokálním optimu. Tento model je založen na čtvercové mřížce. Každý jedinec komunikuje se čtyřmi okolními sousedy. [8]



Obrázek 9: Von Neumannův Model

1.10 Určení parametrů

Při určování parametrů pro PSO je třeba dosažení konvergence ale zároveň zajistit, aby nedošlo k tzv. explozi roje. To znamená, že částice kvůli velké délce skoku přeskakují řešení, které by bylo jinak vhodné. Mezi základní předpoklady pro správou funkci PSO patří[8]:

1. maximální rychlost částic,
2. správné nastavení konstant,
3. správné nastavení setrvačnosti.

1.10.1 Rychlost částice

Vzhledem k tomu, že rychlost částice je generována v každém kroku náhodně, může nabývat jakýchkoli hodnot. Z tohoto důvodu se částice může pohybovat zcela chaoticky. Proto je vhodné nastavit dolní a horní mez. Tyto meze je nutno měnit v závislosti na mezích řešené funkce. v případě velké rychlosti budou částice optimum přeskakovat. v případě malé rychlosti částice budou konvergovat velice pomalu a mohou skončit v lokálním optimu. Podle výzkumu je nejlepší řešení (V_{max}) podle vzorce: [8]

$$V_{max} = (X_{max} - X_{min})/N,$$

kde N je počet intervalů funkce. Rozměry X_{max} a X_{min} jsou minimální a maximální hodnoty pozice aktuálně nalezených částicemi. [8]

1.10.2 Akcelerační konstanty

Akcelerační konstanty zajišťují pohyb částic k nejlepší dosavadně nalezené pozici (ke $gBest$). V případě vysoké hodnoty konstant je nebezpečí, že částice budou divergovat. V případě nízké hodnoty konstant jsou částice velice omezeny v pohybu a mohly by sklouznout do lokálního optima. Z důvodu zjištění optimální velikosti konstant byl proveden experiment. V jednom prostoru byla experimentálně určena akcelerační konstanta φ. Pro jednorozměrný prostor je konstanta jen jedna, protože nejlepší pozice pro optimum je také jen jedna. Hodnota φ je rovna $c_1 + c_2$. Pro nízké hodnoty konstanty φ vypadala trajektorie jako sinusoida. Při vysokých hodnotách se dráha částice blížila nekonečnu. Ideální velikost konstanty byla stanovena na hodnotu 4. Z toho vyplývá, že c_1 a zároveň c_2 mají vhodnou velikost 2. Velikost konstant nemusí být stejná, záleží na optimalizačním problému. [8]

Postupem času se však ukázalo, že i když jsou konstanty i maximální rychlost správně nastaveny, stále může dojít k explozi roje. V současné době jsou dvě metody, jak lze tento problém řešit. [8]

1. Vynásobení pravé strany rovnice omezujícím faktorem χ. Omezující faktor je konstanta, která se používá při situaci, kdy $c_1 + c_2 > 4$. Výpočet omezujícího faktoru vypadá následovně. [8]

$$\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}.$$

Rovnice pro výpočet rychlosti se při této modifikací zapíše následovně:

$$v_i(t) = \chi\{v_i(t-1) + c_1 \cdot rand_1 \cdot (p_i - x_i(t-1)) + c_2 \cdot rand_2 \cdot (p - x_i(t-1))\}.$$

Konvergence je lepší s omezujícím faktorem kvůli snížení oscilace částic. To však platí ve chvíli, kdy jsou částice již zaměřeny na oblast, kde se nachází optimum. Je zde ale riziko, že částice nebudou konvergovat nikdy. To nastává, pokud je jejich nejlepší pozice ($pBest$) příliš vzdálena od nejlepší dosažené hodnoty ($gBest$). [8]

2. Vynásobení setrvačnosti. Parametr pro konstantu setrvačnosti se značí ϕ_{ic}. Tímto parametrem vynásobíme pouze setrvačnost $v_i(t - 1)$, ne celou pravou stranu. Rovnice pro výpočet rychlosti se v touto modifikací zapíše následovně[8]:

$$v_i(t) = \phi_{ic} v_i(t-1) + c_1 \cdot rand_1 \cdot (p_i - x_i(t-1)) + c_2 \cdot rand_2 \cdot (p_g - x_i(t-1)).$$

Konstanta setrvačnosti může být po celou dobu optimalizace konstantní nebo se může měnit v čase. Tímto parametrem se kontroluje detailnost prohledávání. Při spuštění algoritmu je vhodné mít parametr nastavený na relativně vysokou hodnotu (např. 0,9). V takovém případě se částice pohybuje poměrně rychle. Pokud dojde k nalezení oblasti s optimem, je vhodné parametr snížit (např. 0,4). Tím se sníží velikost skoku a zároveň zvýší detailní prohledávání prostoru, což urychluje nalezení optima. Problém však může nastat z důvodu uváznutí v lokálním optimu. [8]

1.11 Kombinace PSO a genetického algoritmu

Kombinace algoritmu PSO a genetických algoritmů je algoritmus, který kombinuje mechanismus přirozeného výběru a výhody inteligence hejna. Tím se zvyšuje počet částic s dobrou pozicí a tím i rychlost konvergence. Algoritmus v každém kroku snižuje počet částic, které jsou špatně ohodnocené. Výhody tohoto algoritmu jsou změny prohledávané oblasti díky $gBest$ a $pBest$ parametrům. Jednou z možností, jak lze PSO vylepšit je aplikování reprodukce, což znamená, že u částic, které si náhodně vybereme, změníme pozici a rychlost. Např.

$$child_1(x) = p \cdot parent_1(x) + (1-p)\, parent_2(x),$$

$$child_2(x) = p \cdot parent_2(x) + (1-p)\, parent_1(x),$$

$$child_1(v) = (parent_1(v) + parent_2(v)) \cdot \frac{|parent_1(v)|}{|paren_{\ 1}(v) + parent_2(v)|},$$

$$child_2(v) = (parent_1(v) + parent_2(v)) \cdot \frac{|parent_2(v)|}{|parent_1(v) + pare_{\ \ 2}(v)|},$$

kde

- p je náhodné číslo ⟨0,1⟩,
- $paretn_{1,2}(x)$ je pozice náhodně zvolené částice,
- $paretn_{1,2}(v)$ je rychlost částice,
- $child_{1,2}(x)$ a $child_{1,2}(v)$ jsou potomci.

Potomci následně nahradí rodiče s nízkou hodnotou funkce. Rodiče jsou z algoritmu odstraněni. [8]

1.12 Explorace a exploitace

V celém algoritmu PSO je nezbytné mít správný poměr mezi Explorací a exploitací. Jsou funkce, kdy je důležitější pro nalezení optima více explorace než exploitace. Tím se zvyšuje

šance najít optimum, ale zároveň se také zvyšuje doba konvergence. Explorace a exploitace jsou nezbytné pro správné fungování algoritmu PSO. [9]

Explorace je v rámci PSO znamená prohledávání velké části grafu. V důsledku zachování rozmanitosti a zároveň neuváznout v lokálním optimu je udržení explorace v PSO velice důležité. Graf sice není prohledáván příliš do hloubky, ale celé hejno získá povědomí o velkém prostoru, na němž se nachází. V rámci explorace částice prohledávají od sebe vzdálená místa. [9]

Exploitace je opakem explorace. Při exploitaci PSO je prohledávána vybraná oblast, která je již relativně malá. Tato oblast je prohledávána převážně do hloubky. ve správné oblasti, ve které se nachází i hledané optimum. s exploitací také souvisí konvergence algoritmu. Částice v této fázi se již nepohybují chaoticky po celém prostoru grafu, ale pouze po určité prohledávané části. Tato část je vybrána jako nejlepší v rámci explorace. [9]

2 TESTOVACÍ FUNKCE

Testovací funkce jsou funkce, pro které je složité nalézt globální minimum. Funkce jsou vykresleny v programu Matlab 9.4.0.813654 (R2018a). [4][5]

Existuje mnoho funkcí, u nichž problém minimalizace nelze řešit analyticky nebo to není vhodné. Důvody pro řešení jiným, než analytickým způsobem jsou:

- vysoká výpočetní náročnost,
- vůbec neexistuje analytické řešení.

2.1 Ackleyho funkce

Ackleyho funkce je dána předpisem:

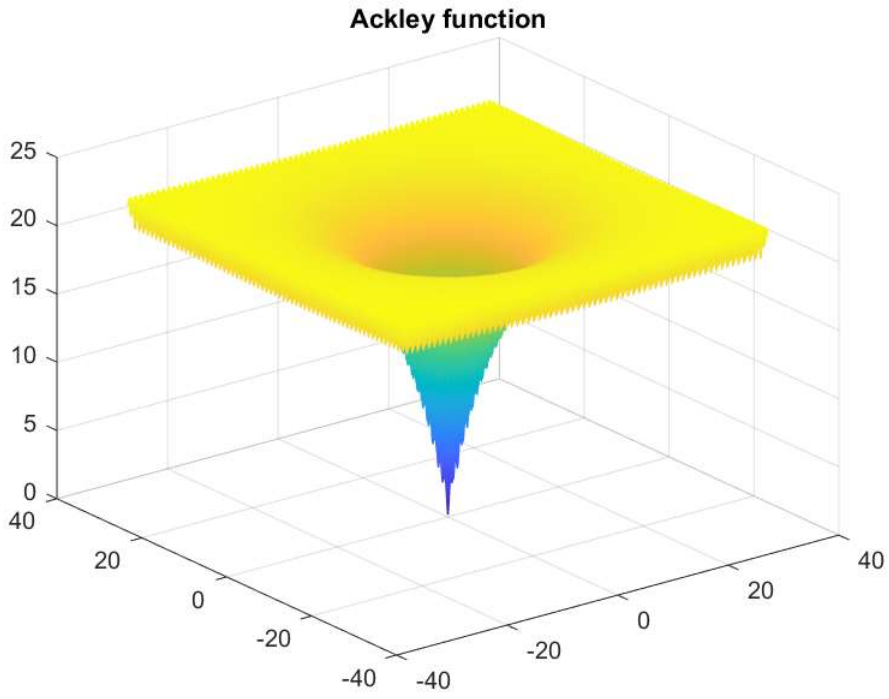
$$f(x) = -a\,exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right) + a + exp(1), \text{ kde}$$

globální minimum je:

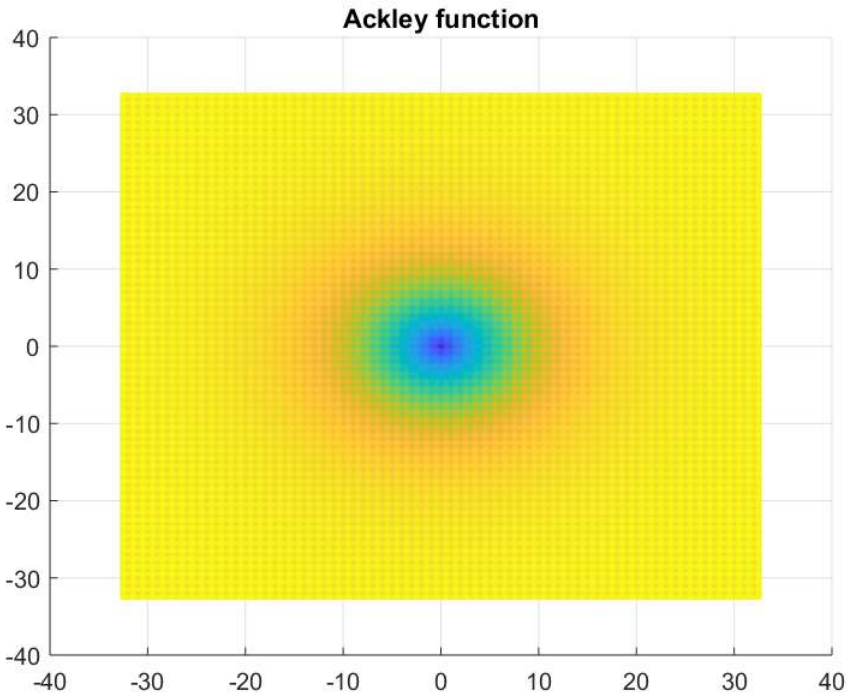
- $f(x^*) = 0$,
- $x^* = (0, \ldots, 0)$.

Interval funkce je:

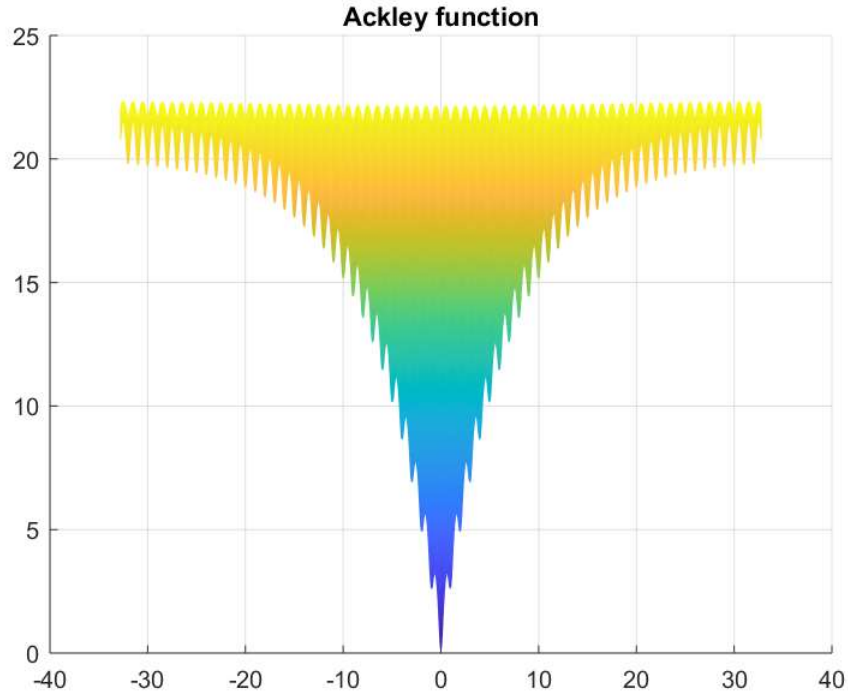
- ⟨-32.768, 32.768⟩.



Obrázek 10: Ackleyho funkce



Obrázek 11: Ackleyho funkce 2D



Obrázek 12: Ackleyho funkce profil

2.2 Drop-Wave funkce

Drop-Wave funkce je dána předpisem:

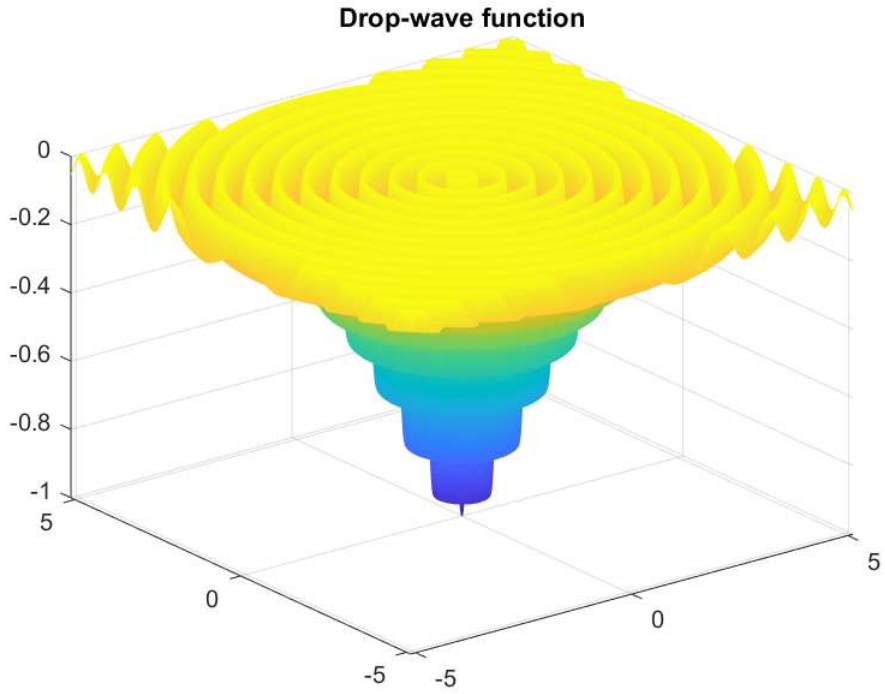
$$f(x) = -\frac{1+cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5\left(x_1^2+x_2^2\right)+2}, \text{ kde}$$

globální minimum je:

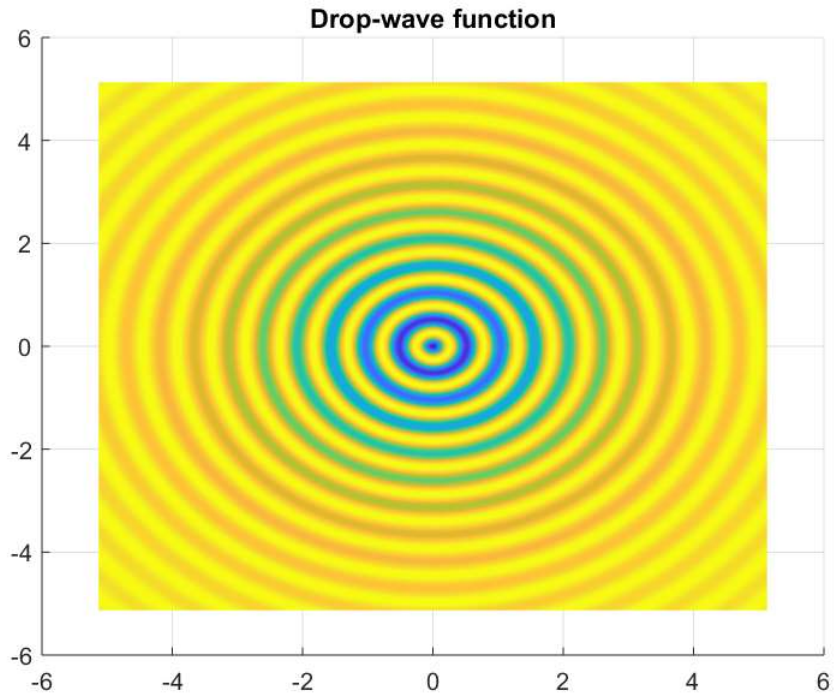
- $f(x^*) = -1$,
- $x^* = (0,0)$.

Interval funkce je:

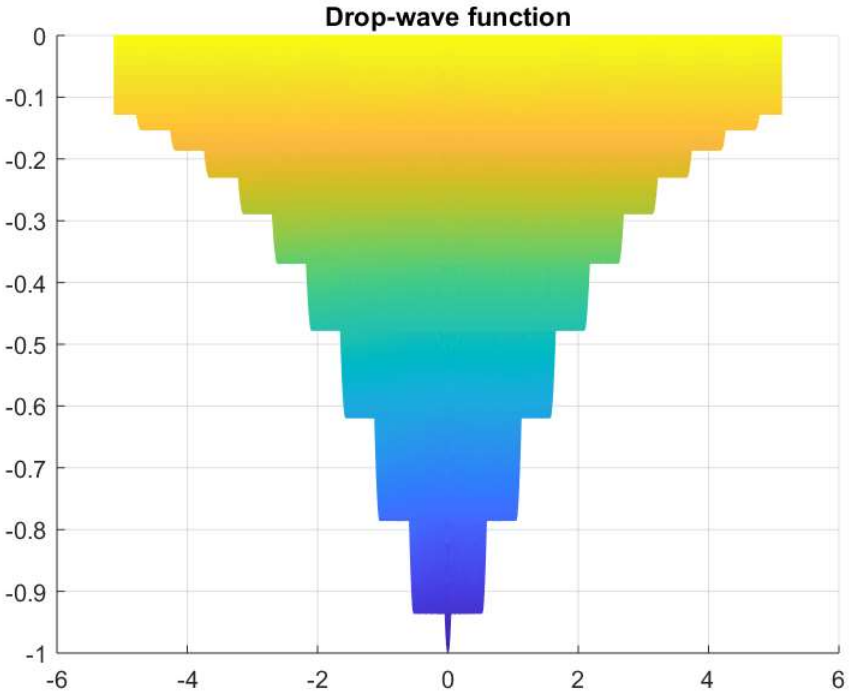
- ⟨- 5.12, 5.12⟩.



Obrázek 13: Drop-Wave funkce



Obrázek 14: Drop-Wave 2D



Obrázek 15: Drop-Wave funkce profil

2.3 Griewangkova funkce

Griewangkova funkce je dána předpisem:

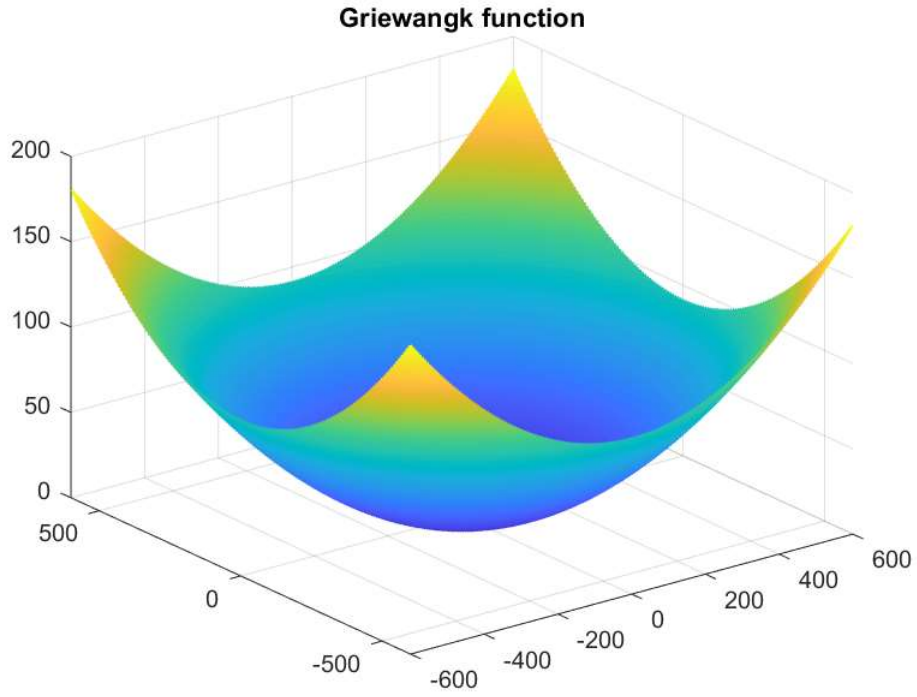
$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \text{ kde}$$

globální minimum je:

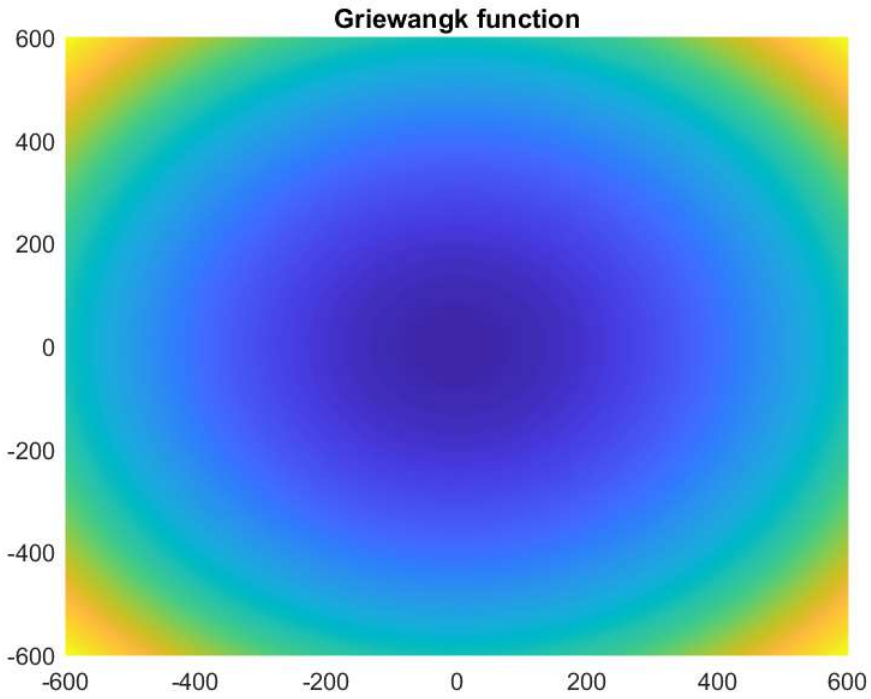
- $f(x^*) = 0,$
- $x^* = (0, \ldots, 0).$

Interval funkce je:

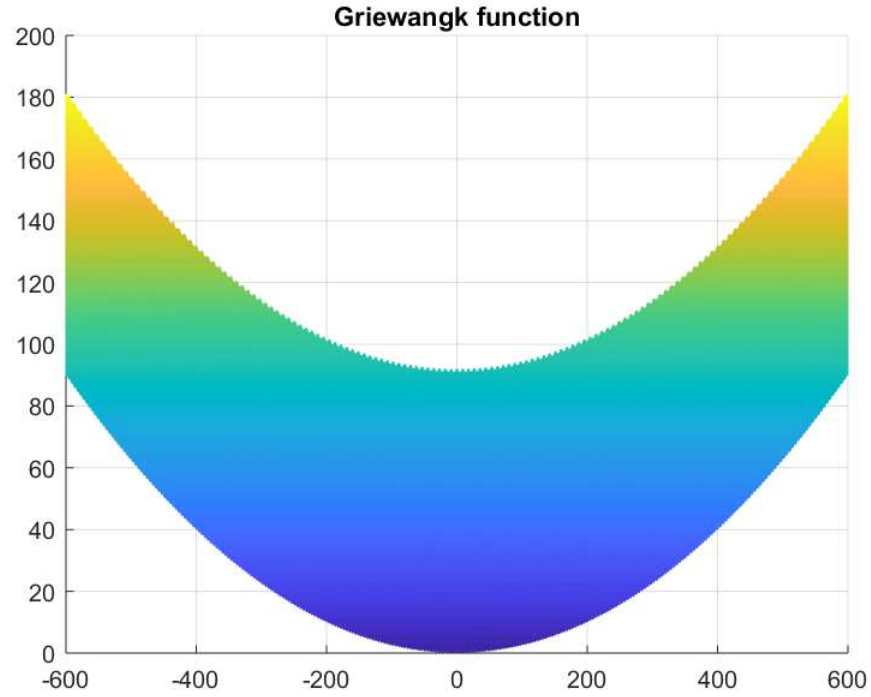
- ⟨- 600, 600⟩.



Obrázek 16: Griewangkova funkce



Obrázek 17: Griewangkova funkce 2D



Obrázek 18: Griewangkova funkce profil

2.4 Eggholderova funkce

Eggholderova funkce je dána předpisem:

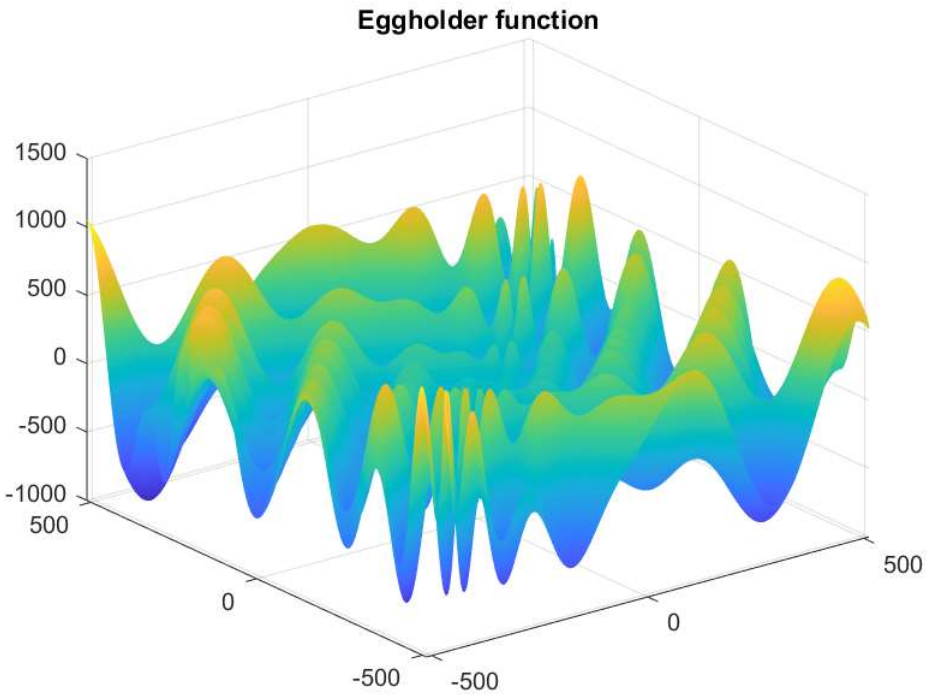
$$f(x) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right), \text{ kde}$$

globální minimum je:

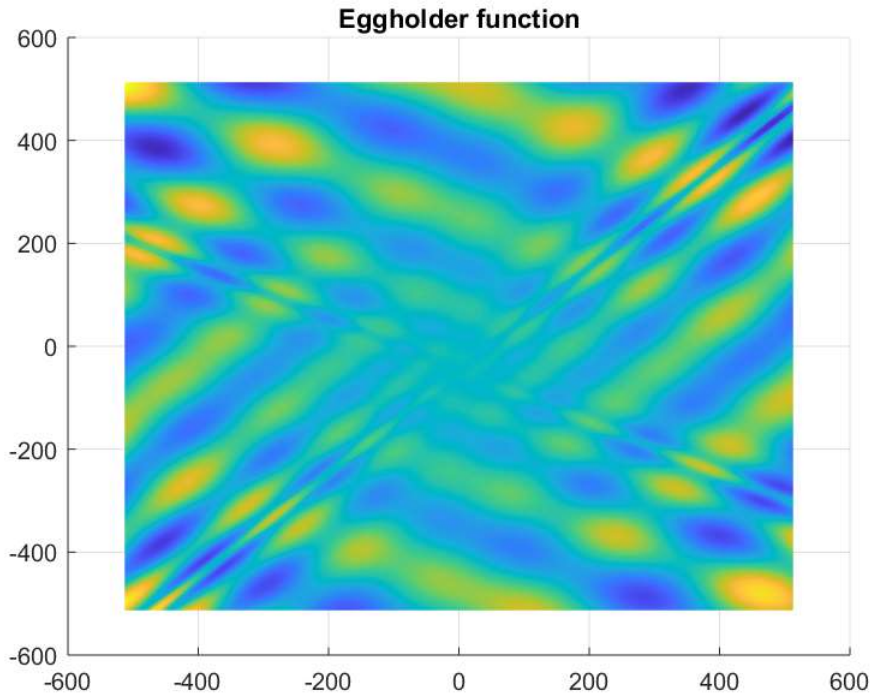
- $f(x^*) = -959.6407,$
- $x^* = (512, 404.2319).$

Interval funkce je:

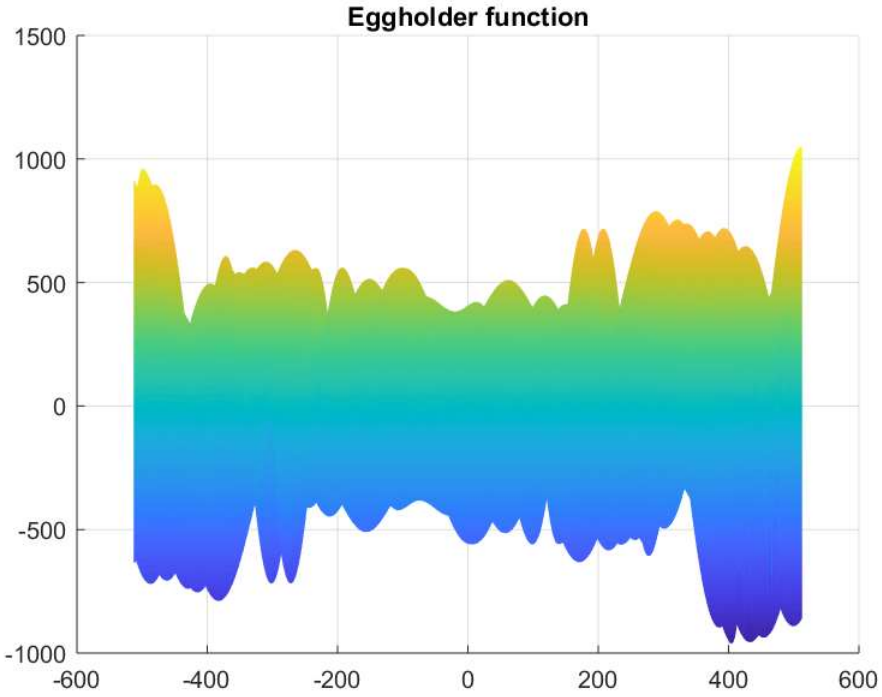
- $\langle -512, 512 \rangle.$



Obrázek 19: Eggholderova funkce



Obrázek 20: Eggholderova funkce 2D



Obrázek 21: Eggholderova funkce profil

2.5 Levyho funkce

Levyho funkce je dána předpisem:

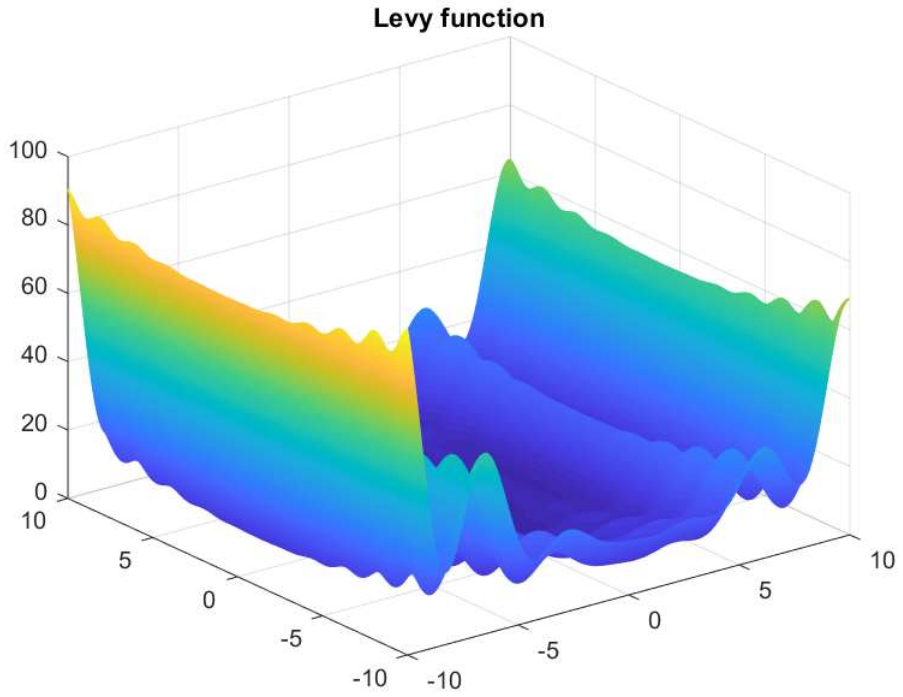
$$f(x) = sin^2(\pi\omega_1) + \sum_{i=1}^{d-1}(\omega_i - 1)^2[1 + 10sin^2(\pi\omega_i + 1)] + (\omega_d - 1)^2[1 + sin^2(2\pi\omega_d)],$$
$$kde\ \omega_i = 1 + \frac{x_i - 1}{4}, pro\ všechny\ i = 1, \dots, d.$$

Globální minimum je:

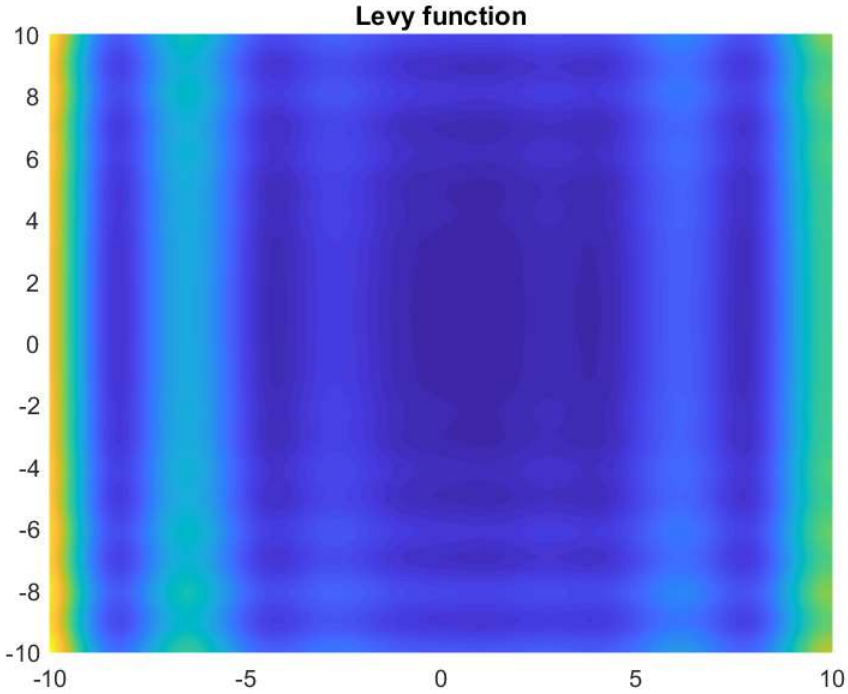
- $f(x^*) = 0,$
- $x^* = (1, \dots, 1).$

Interval funkce je:

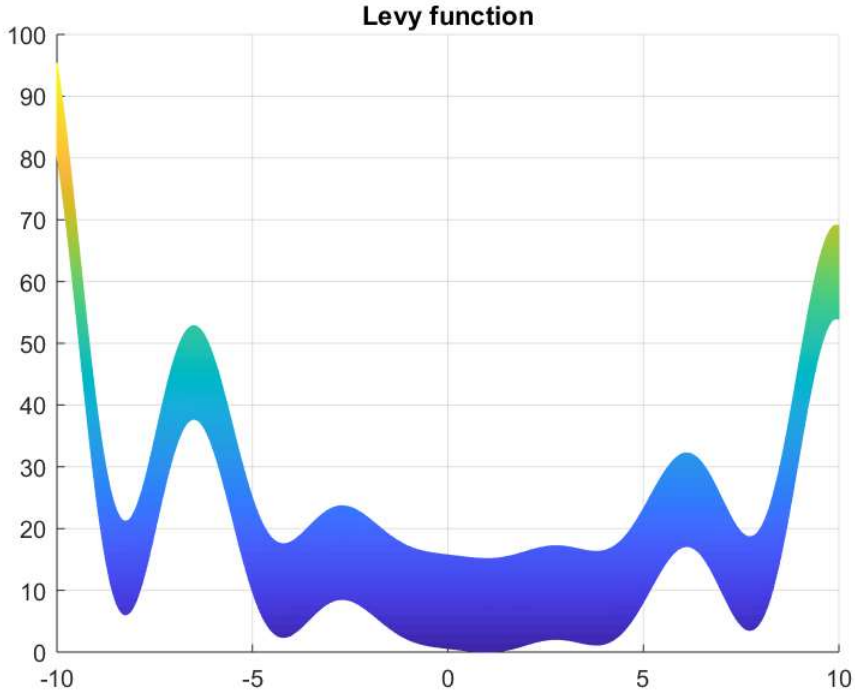
- $\langle$- 10, 10$\rangle$.



Obrázek 22: Levyho funkce



Obrázek 23: Levyho funkce 2D



Obrázek 24: Levyho funkce profil

2.6 Rastriginova funkce

Rastriginova funkce je dána předpisem:

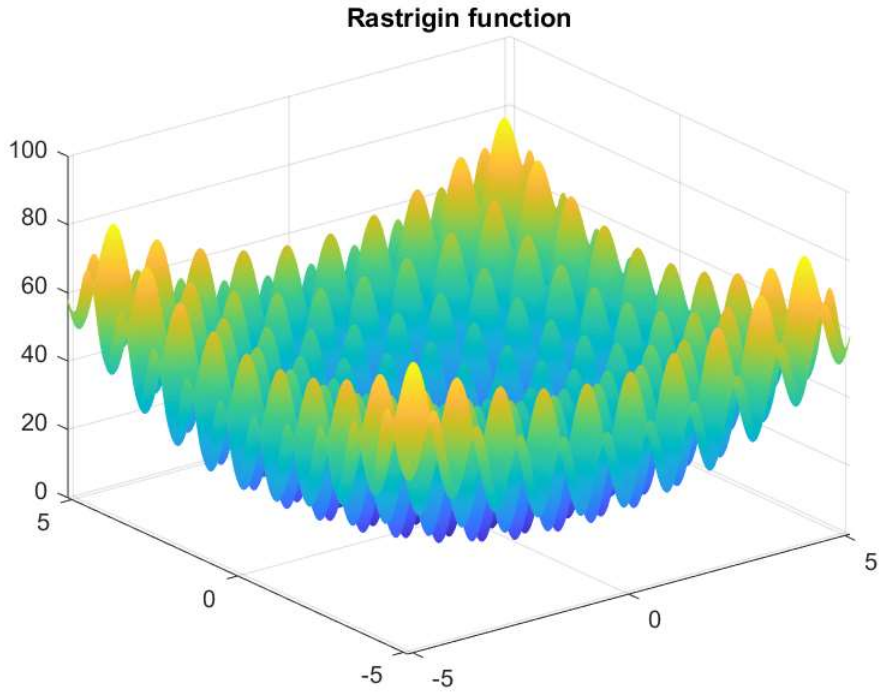
$$f(x) = 10\mathrm{d} + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)], \text{ kde}$$

globální minimum je:

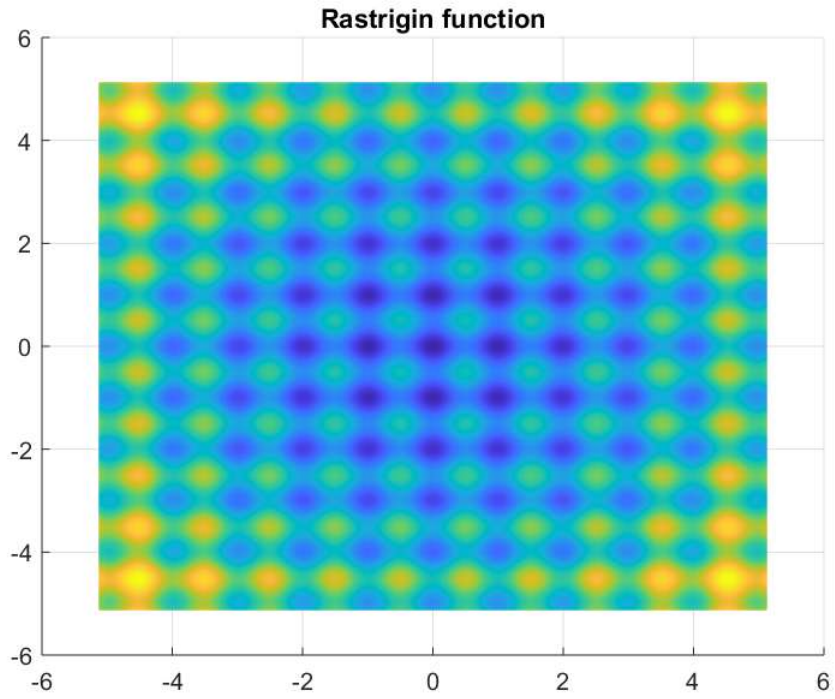
- $f(x^*) = 0$,
- $x^* = (0, \ldots, 0)$.

Interval funkce je:

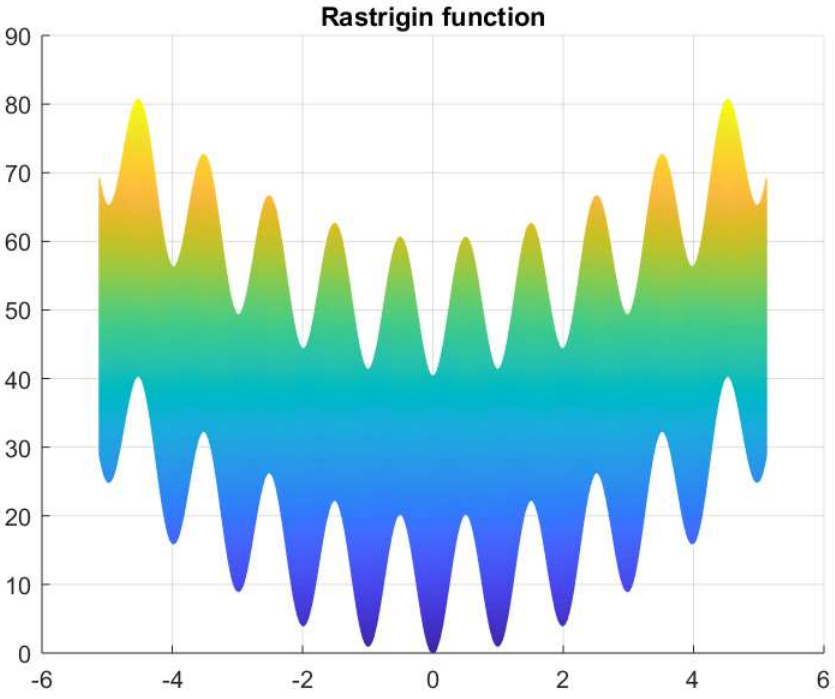
- ⟨- 5.12, 5.12⟩.



Obrázek 25: Rastriginova funkce



Obrázek 26: Rastriginova funkce 2D



Obrázek 27: Rastriginova funkce profil

2.7 Schwefelova funkce

Schwefelova funkce je dána předpisem:

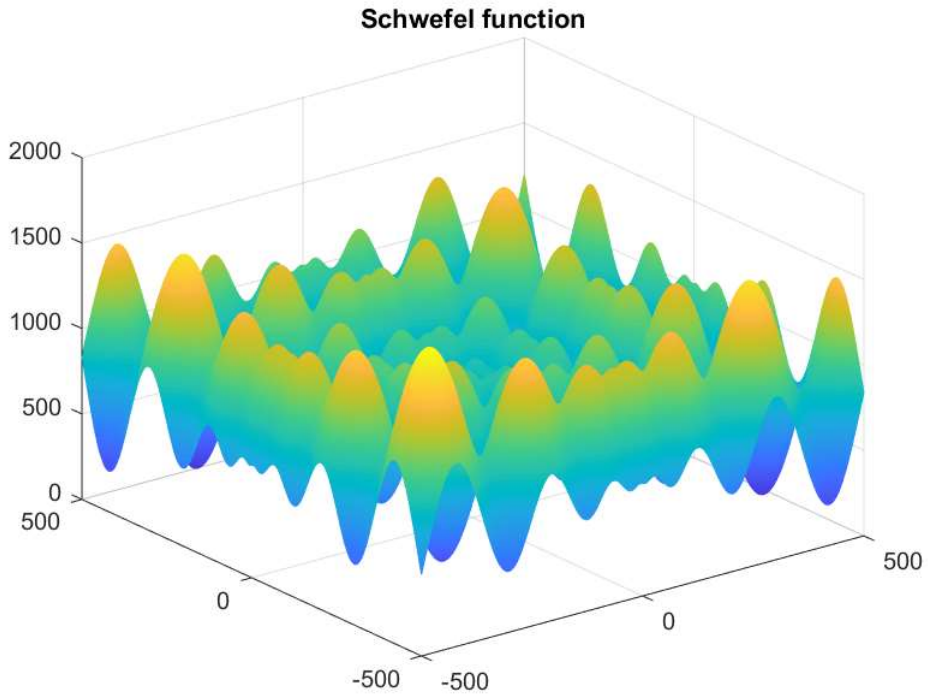
$$f(x) = 418.9829\mathrm{d} - \sum_{i=1}^{d} x_i \sin\left(\sqrt{|x_i|}\right), \text{ kde}$$

globální minimum je:

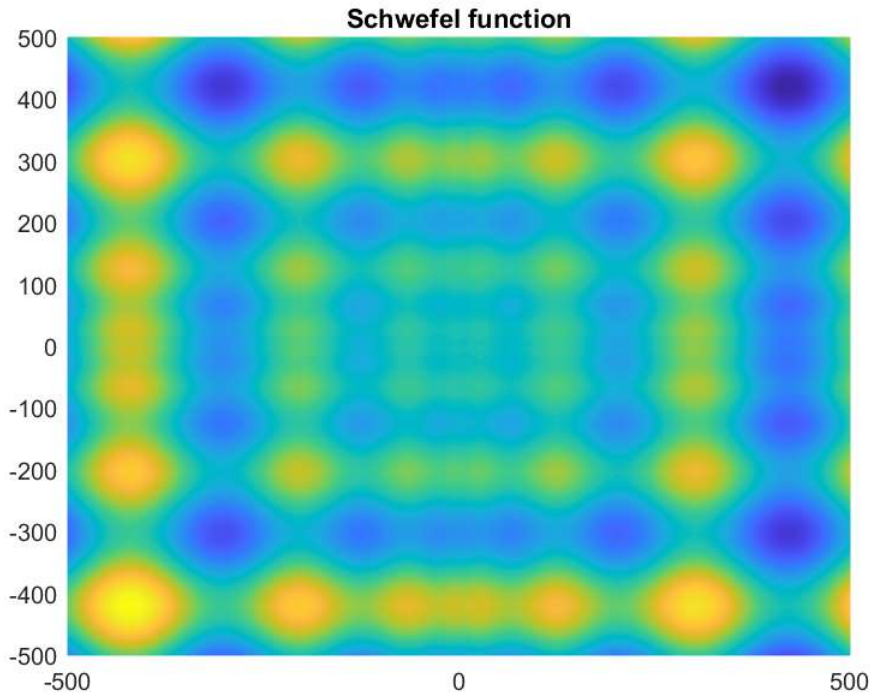
- $f(x^*) = 0$,
- $x^* = (420.9687, \ldots, 420.9687)$.

Interval funkce je:

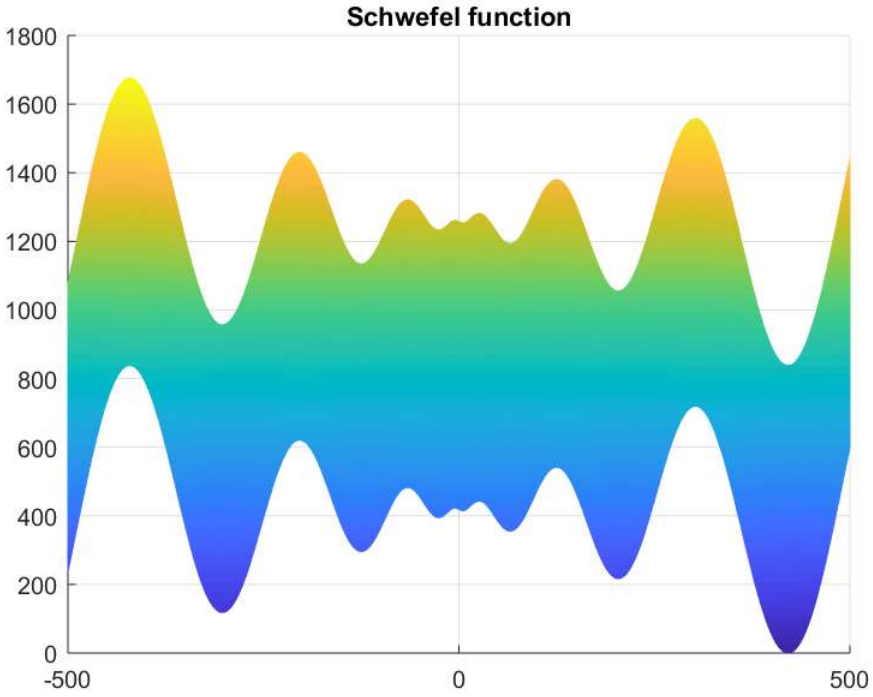
- ⟨- 500, 500⟩.



Obrázek 28: Schwefelova funkce



Obrázek 29: Schwefelova funkce 2D



Obrázek 30: Schwefelova funkce profil

2.8 Rosenbrockova funkce

Rosenbrockova funkce je dána předpisem:

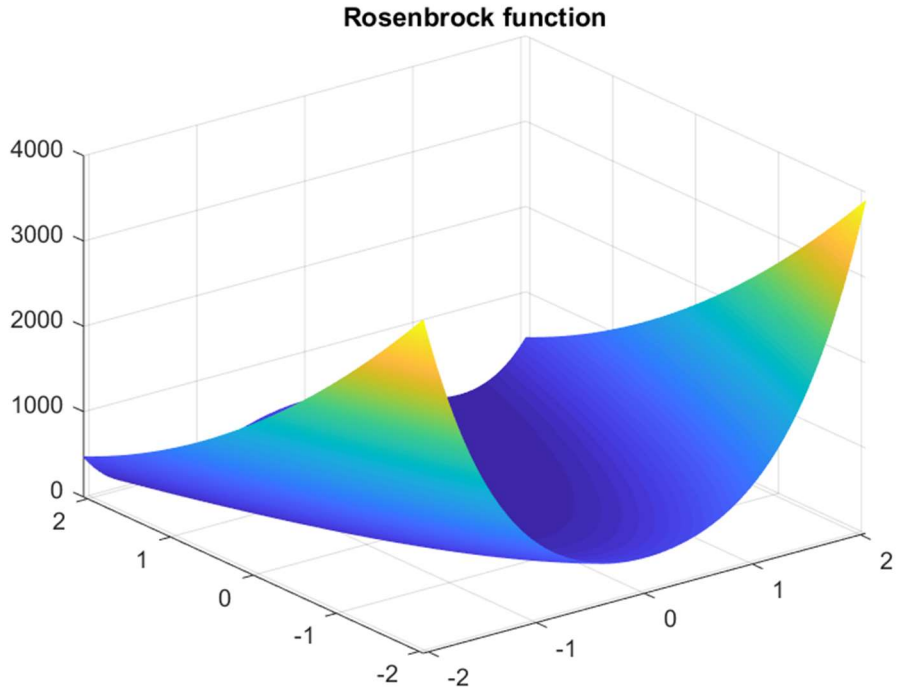
$$f(x) = \sum_{i=1}^{d-1}[100(x_{i-1} - x_i^2)^2 + (x_i - 1)^2], \text{ kde}$$

globální minimum je:

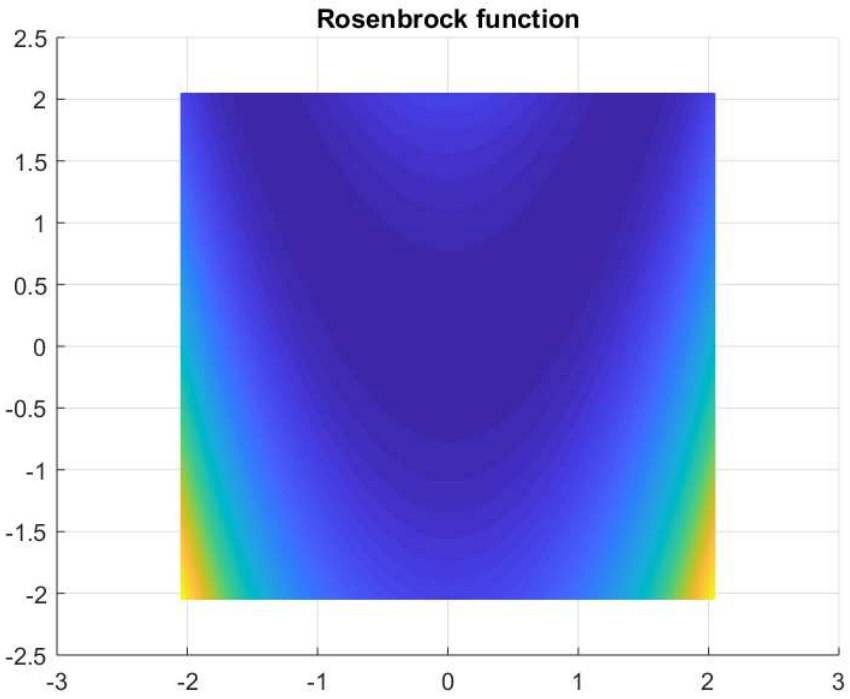
- $f(x^*) = 0$,
- $x^* = (1, \dots, 1)$.

Interval funkce je:

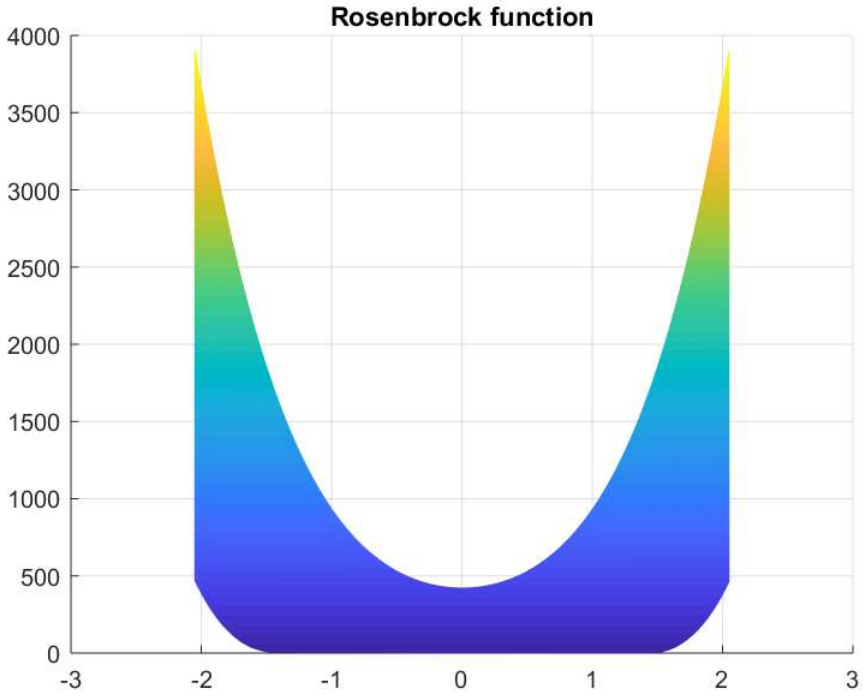
- ⟨-2.048, 2.048⟩.



Obrázek 31: Rosenbrockova funkce



Obrázek 32: Rosenbrockova funkce 2D



Obrázek 33: Rosenbrockova funkce profil

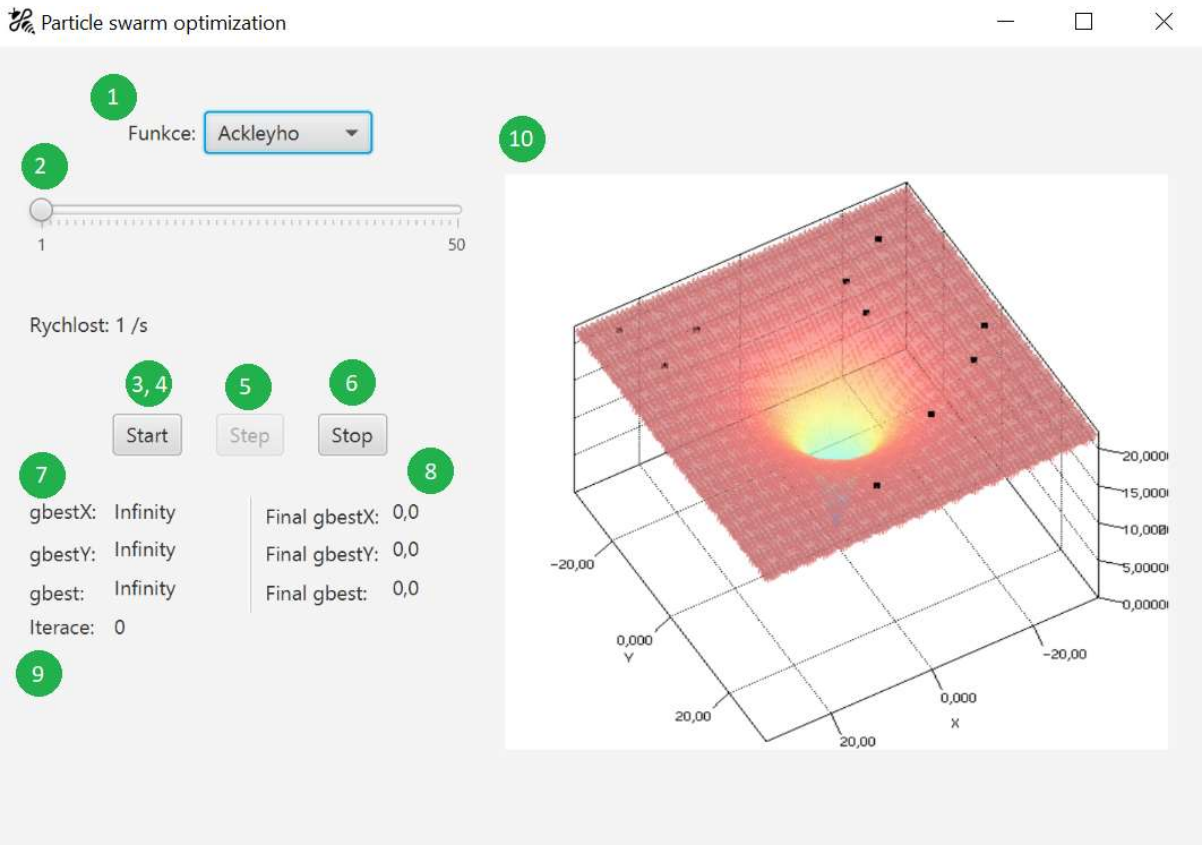
3 POPIS PROGRAMU

Program je postaven na vzorci pro vyhledávání optima s násobením setrvační konstantou podle:

$$v_i(t) = \phi_{ic} v_i(t-1) + c_1 \cdot rand_1 \cdot (p_i - x_i(t-1)) + c_2 \cdot rand_2 \cdot (p_g - x_i(t-1)).$$

Aplikace je vytvořena ve vývojovém prostředí NetBeans IDE 8.2 (Build 201610071157) v jazyce JavaFX.

Na Obr. 34 jsou v zeleně očíslovány prvky v grafickém rozhraní aplikace, které dále podrobně popíši.



Obrázek 34: Popis aplikace

1. Funkce: „název funkce“ – pro vykreslení požadované funkce je nutné vybrat funkci z kombinovaného pole. Zde jsou na výběr funkce, které jsou implementovány. Při spuštění programu je kombinované pole inicializováno na funkci „Ackleyho“.
2. Slider počtu iterací za sekundu. Lze měnit libovolně i za běhu programu. Při maximální rychlosti nemusí počet iterací za sekundu odpovídat z důvodu výpočetní náročnosti algoritmu.

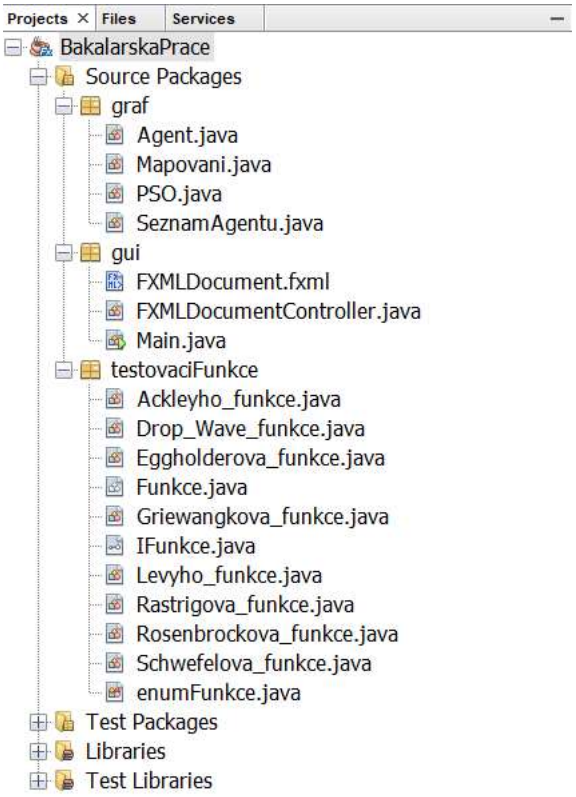
3. Tlačítko „Start“ – spustí simulaci hledání optima. Zároveň se spustí animace v grafu (10). Mění se pohyb částic v grafu pro lepší znázornění práce algoritmu. Pokud je simulace spuštěna, nápis na tlačítku se změní na „Pause“.
4. Tlačítko „Pause“ – pozastaví aktuálně běžící simulaci. Výsledné hodnoty nejsou změněny. Pokud je simulace pozastavena, nápis na tlačítku se změní na „Start“.
5. Tlačítko „Step“ – pokud simulace není spuštěna, tlačítko je zablokované. Po spuštění simulace je tlačítko odblokováno a provede následující iteraci nezávisle na běhu samotné simulace.
6. Tlačítko „Stop“ – zastaví běh simulace, odstraní se dosavadní nalezené výsledky a nově se inicializuje pole částic.
7. „gbestX“, „gbestY“, „gbest“ – nejlepší hodnoty, které byly simulací dosaženy.
8. „Final gbestX“, „Final gbestY“, „Final gbest“ – hledané hodnoty funkce.
9. „Iterace“ – aktuální iterace simulace. Každá iterace posune částice podle algoritmu PSO. V jedné iteraci se posune částice pouze 1x.
10. Graf funkce – 3D zobrazení grafu funkce včetně pozic agentů. Graf lze libovolně otáčet myší a tím měnit pohled.

3.1 Ukázky kódu

Kód je strukturován do balíčků.

1. Graf – obsahuje soubory pro vytvoření částice, seznamu částic, mapování grafu, výpočet algoritmu PSO.
2. Gui – obsahuje veškerou grafickou část a propojení funkce, algoritmu a částic do grafického rozhraní. K animaci pohybu částic v grafu byly využity třídy Timer a TimerTask.
3. Testovací funkce – balíček s implementovanými funkcemi. Všechny funkce mají implementované rozhraní IFunkce a jsou potomky abstraktní třídy Funkce. Jména všech funkcí jsou zapsána ve výčtovém souboru enumFunkce.

Následující obrázek ukazuje strukturu programu – rozložení do tříd.



Obrázek 35: Struktura programu

Tato část kódu popisuje kontrolu $pBest$ a $gBest$. Následně je zde počítána rychlost, což je srdce celého algoritmu.

```
if (pbest.z > z) {
    pbest.set((float) x, (float) y, (float) z);
    if (gBest.z >= z) {
        gBest = pbest;
    }
}

rychlost.x = (float) (rychlost.x * konfidenčníKoeficient2 + a.generatorNahodnychCisel(0, konfidenčníKoeficient1) * (pbest.x - x)
        + a.generatorNahodnychCisel(0, konfidenčníKoeficient1) * (gBest.x - x));
rychlost.y = (float) (rychlost.y * konfidenčníKoeficient2 + a.generatorNahodnychCisel(0, konfidenčníKoeficient1) * (pbest.y - y)
        + a.generatorNahodnychCisel(0, konfidenčníKoeficient1) * (gBest.y - y));

x += rychlost.x;
y += rychlost.y;
```

Obrázek 36: Řešení PSO v kódu JavaFX

Tato část kódu popisuje kontrolu mezí, což znamená, že pokud by skok vedl mimo interval, nastaví se pozice na hranici intervalu, který byl překročen.

```
x += rychlost.x;
y += rychlost.y;

if (fc.getIntervalOd() > x) {
    x = fc.getIntervalOd();
    rychlost.x = 0;
}
if (fc.getIntervalDo() < x) {
    x = fc.getIntervalDo();
    rychlost.x = 0;
}
if (fc.getIntervalOd() > y) {
    y = fc.getIntervalOd();
    rychlost.y = 0;
}
if (fc.getIntervalDo() < y) {
    y = fc.getIntervalDo();
    rychlost.y = 0;
}
z = fc.funkce(x, y);

aktualniPozice.setData(new Coord3d(x, y, z));
```

Obrázek 37: Kontrola hranic skoku v kódu JavaFX

Ukázka třídy, jenž předepisuje informace o testovací funkci.

```
public class Ackleyho_funkce extends Funkce {

    final float intervalOd = (float) -32.768;
    final float intervalDo = (float) 32.768;

    final double X = 0;
    final double Y = 0;
    final double hodnota = 0;

    @Override
    public double funkce(double x, double y) {
        return ((-20) * Math.exp((-0.2) * Math.sqrt((x * x + y * y) / 2))
                - Math.exp((Math.cos(2 * x * Math.PI) + Math.cos(2 * y * Math.PI)) / 2)
                + 20 + Math.E);
    }

    @Override
    public double getX() {
        return X;
    }

    @Override
    public double getY() {
        return Y;
    }

    @Override
    public float getIntervalOd() {
        return intervalOd;
    }

    @Override
    public float getIntervalDo() {
        return intervalDo;
```

Obrázek 38: Ukázka předpisu funkce v JavaFX

3.2 Porovnání výsledků

Výsledky byly vyhodnoceny v programu Microsoft Excel pro Office 365 MSO (16.0.10730.20264) 32bitová verze.

Program byl testován na testovacích funkcích. Každá funkce byla testována 100krát. Každé testovaní probíhalo do doby, dokud byl skok větší než 0,001. První sloupec udává konečný počet iterací do splnění ukončující podmínky. Druhý sloupec je hodnota, kterou algoritmus nalezl v aktuální iteraci.

3.2.1 Ackleyho funkce

Iterace	Hodnota
32	0,140333
32	0,050061
32	0,165023
34	0,039288
39	0,138876
40	0,065380
40	0,018603
40	0,018148
40	0,011014
41	0,082777
41	0,025626
42	0,020168
42	0,050554
43	0,054820
43	0,032203
44	0,020211
44	0,014412
44	0,047475
45	0,005075
45	0,017598
45	0,000780
46	0,003377
46	0,004184
46	0,027733
46	0,041437
46	0,025800
47	0,017453
47	0,025115
47	0,024807
47	0,004930
48	0,006152
48	0,071534
49	0,011189
49	0,003162
49	0,015689
49	0,070983
49	0,007995
50	0,002624
51	0,003520
52	0,011172
54	0,007396
54	0,015900
55	0,002333
55	0,003557
55	0,007084
55	0,002898
56	0,001034
56	0,001176
56	0,001707
56	0,002505
56	0,007010
56	0,002330
56	0,013533
57	0,010670
57	0,000971
57	0,000971
58	0,007070
58	0,004994
58	0,002968
58	0,003728
59	0,002642
59	0,001480
59	0,001135
59	0,000542
59	0,001103
59	0,001132
59	0,003065
59	0,003169
60	0,006780
60	0,002520
60	0,005164
60	0,005816
60	0,002720
61	0,001653
63	0,001804
63	0,000935
63	0,000945
63	0,000547
63	0,000529
64	0,002920
64	0,000562
64	0,001141
64	0,000763
64	0,000763
65	0,002448
65	0,000516
65	0,000516
66	0,000278
66	0,001788
67	0,000229
67	0,000229
68	0,000494
68	0,000494
68	0,000335
68	0,000829
68	0,000829
68	0,000819
68	0,000819
71	0,001931
82	0,000392

Tabulka 1: Ackleyho funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla 0,015599,
- směrodatná odchylka byla 0,029352,
- medián je 0,003273,
- modus je 0,000971,
- průměrný počet iterací k nalezení použitelného optima je 54,41.

Nejhorší experiment poskytl výsledek odlišný o 0,165023 od skutečné hodnoty.

3.2.2 Drop-wave funkce

Iterace	Hodnota
6	-0,785703
7	-0,935096
9	-0,936044
10	-0,936098
11	-0,936187
11	-0,936198
11	-0,936244
12	-0,936241
12	-0,932438
13	-0,935474
13	-0,935968
13	-0,785083
14	-0,936232
14	-0,936158
14	-0,936018
14	-0,934805
14	-0,936222
14	-0,936229
14	-0,928391
15	-0,936236
16	-0,935557
16	-0,936245
16	-0,936199
18	-0,936239
18	-0,936208
18	-0,936098
18	-0,936242
19	-0,936213
19	-0,936244
19	-0,936117
20	-0,933936
20	-0,935865
20	-0,936091
20	-0,936235
20	-0,935943
21	-0,935656
21	-0,935956
21	-0,936216
21	-0,936180
21	-0,936240
21	-0,935898
21	-0,936238
21	-0,977507
22	-0,934040
23	-0,936182
23	-0,936238
23	-0,935968
23	-0,936236
23	-0,936241
23	-0,936240
24	-0,936163
24	-0,936220
24	-0,931596
24	-0,936242
25	-0,936051
25	-0,935597
25	-0,936245
26	-0,936030
27	-0,936210
27	-0,936241
27	-0,934720
27	-0,999638
27	-0,936245
27	-0,997362
28	-0,999513
28	-0,999513
28	-0,936245
28	-0,936063
29	-0,936169
29	-0,936243
30	-0,935931
30	-0,935940
31	-0,936245
32	-0,936033
32	-0,996931
34	-0,999085
35	-0,936194
35	-0,999369
35	-0,999369
36	-0,936080
36	-0,936170
37	-0,999757
37	-0,986531
38	-0,999445
38	-0,999445
38	-0,936115
40	-0,936236
40	-0,999174
41	-0,999193
41	-0,999193
42	-0,999860
42	-0,999860
42	-0,999899
44	-0,936240
45	-0,999071
45	-0,999071
45	-0,999231
47	-0,936245
49	-0,999834
49	-0,999990

Tabulka 2: Drop-wave funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla -0,947676,
- směrodatná odchylka byla 0,035416,
- medián je -0,936226,
- modus je -0,936245,
- průměrný počet iterací k nalezení použitelného optima je 25,41.

Nejhorší experiment poskytl výsledek odlišný o -0,214917 od skutečné hodnoty.

3.2.3 Eggholderova funkce

Iterace	Hodnota
16	-59,640600
17	-21,196400
19	-33,842200
21	-21,196350
22	-94,564150
22	-33,841900
22	-59,640600
24	-59,640440
24	-33,841300
25	-33,840800
25	-18,029000
26	-21,196300
26	-59,640700
26	-59,639950
29	-21,196400
31	-59,638900
31	-21,173950
32	-59,640600
34	-21,196400
34	-88,947800
34	-59,640500
34	-59,640260
38	-18,161500
38	-94,575500
38	-16,663150
39	-88,948800
39	-21,196350
40	-88,948850
41	-59,640600
41	-88,916930
42	-59,628400
42	-59,640500
42	-18,167400
42	-935,32056
43	-88,948550
43	-94,578200
43	-86,514600
43	-18,167000
44	-88,948670
44	-59,640700
44	-35,335450
45	-94,573550
45	-94,578400
46	-94,576660
46	-21,196400
47	-59,636660
47	-86,525940
47	-16,670800
48	-88,945200
50	-16,666750
50	-88,948360
50	-57,361800
51	-35,327330
51	-04,792900
51	-57,857240
52	-94,577800
52	-57,857500
53	-94,578740
53	-86,522900
53	-94,575300
54	-15,921140
54	-18,167240
55	-38,012000
55	-88,949000
55	-94,578800
56	-88,949040
57	-94,577900
57	-94,578400
58	-93,959870
58	-94,578300
59	-93,960200
59	-86,521800
59	-94,571530
61	-88,936770
61	-88,949100
62	-88,937300
63	-53,049800
64	-15,975700
64	-59,640400
66	-86,525450
66	-59,640140
68	-94,569950
70	-59,640600
72	-88,948900
73	-94,578700
75	-86,517000
75	-94,573600
76	-94,573900
78	-53,049500
79	-53,049900
79	-86,525700
85	-56,888850
89	-56,916560
90	-56,917240
91	-86,525760
96	-94,578860
102	-94,576050
158	-56,900800
159	-956,90120
170	-56,917850

Tabulka 3: Eggholderova funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla -841,306682,
- směrodatná odchylka byla 115,551113,
- medián je -888,948875,
- modus je -959,640600,
- průměrný počet iterací k nalezení použitelného optima je 53,25.

Nejhorší experiment poskytl výsledek odlišný o -465,640130 od skutečné hodnoty.

3.2.4 Griewangkova funkce

Iterace	Hodnota
31	0,106681
33	0,111875
35	0,012636
39	0,025703
40	0,196883
43	0,148791
45	0,009785
45	0,010877
45	0,005856
48	0,040097
48	0,059523
49	0,095887
50	0,012094
50	0,057445
50	0,040978
51	0,011621
51	0,002833
52	0,000236
52	0,293099
54	0,009877
56	0,008988
57	0,014562
57	0,095855
57	0,030575
58	0,014403
59	0,148580
59	0,029168
59	0,106640
60	0,112246
60	0,080599
61	0,020364
61	0,049943
62	0,158943
62	0,179841
63	0,246938
63	0,053019
64	0,041542
65	0,040778
66	0,029821
66	0,011081
66	0,012023
67	0,013502
67	0,043366
67	0,041806
68	0,011152
69	0,019792
70	0,010082
70	0,041189
70	0,021207
70	0,132636
71	0,014600
71	0,026318
71	0,020562
71	0,020038
72	0,025639
72	0,019830
73	0,089553
73	0,010256
74	0,080346
74	0,010193
74	0,012541
74	0,019828
74	0,010187
74	0,012241
75	0,011259
75	0,020372
77	0,025997
77	0,052877
77	0,049750
78	0,025686
78	0,009964
79	0,017426
79	0,000588
79	0,051192
80	0,010302
80	0,012594
80	0,010112
82	0,049443
83	0,004921
85	0,000138
86	0,050136
87	0,004568
87	0,040809
87	0,010404
88	0,004019
92	0,011125
94	0,114315
96	0,013785
98	0,040268
100	0,050123
101	0,050056
102	0,004614
102	0,250442
103	0,039461
105	0,009871
106	0,011401
107	0,011396
130	0,012413
131	0,012002
153	0,001911

Tabulka 4: Griewangkova funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla 0,045413,
- směrodatná odchylka byla 0,057108,
- medián je 0,020467,
- konkrétní modus nebyl nalezen, protože funkce je multimodální,
- průměrný počet iterací k nalezení použitelného optima je 71,47.

Nejhorší experiment poskytl výsledek odlišný o 0,293099 od skutečné hodnoty.

3.2.5 Levyho funkce

Iterace	Hodnota
4	0,000255
7	0,000543
8	0,000266
11	0,000140
11	0,000902
11	0,032513
12	0,035432
12	0,002039
13	0,000977
13	0,000977
13	0,001460
13	0,073409
14	0,000663
14	0,000550
14	0,000550
14	0,002508
15	0,003428
15	0,000205
15	0,028140
15	0,000060
16	0,000898
16	0,000466
16	0,000817
16	0,014675
16	0,000921
16	0,000546
17	0,000698
17	0,000631
17	0,000631
17	0,001029
17	0,000767
17	0,000483
17	0,000263
17	0,002752
17	0,001183
17	0,000040
17	0,000206
18	0,000534
18	0,000534
18	0,001503
18	0,000513
18	0,000513
18	0,005596
18	0,010062
18	0,000924
18	0,000924
18	0,000709
18	0,000709
19	0,000380
19	0,001173
19	0,000101
19	0,001333
19	0,000132
19	0,000063
19	0,001433
19	0,000177
19	0,000344
19	0,000660
19	0,000660
19	0,002554
19	0,000442
19	0,000442
20	0,000352
20	0,000352
20	0,000198
20	0,002345
20	0,001324
20	0,000681
20	0,000681
21	0,000116
21	0,002606
21	0,000416
22	0,000341
22	0,001703
22	0,000123
22	0,000942
22	0,001919
22	0,001129
23	0,000631
24	0,000871
24	0,000871
25	0,000597
25	0,000597
25	0,000689
25	0,000689
25	0,000041
25	0,000680
26	0,000677
26	0,000677
26	0,002229
27	0,000492
27	0,000492
27	0,000391
31	0,000313
31	0,000313
32	0,000087
32	0,000087
32	0,000551
32	0,000551
4	0,000255

Tabulka 5: Levyho funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla 0,002749,
- směrodatná odchylka byla 0,009143,
- medián je 0,000660,
- modus je 0,000977,
- průměrný počet iterací k nalezení použitelného optima je 19,15.

Nejhorší experiment poskytl výsledek odlišný o 0,073409 od skutečné hodnoty.

3.2.6 Rastrigova funkce

Iterace	Hodnota
23	1,997409
24	1,997408
27	0,158946
30	1,991139
30	0,998521
30	0,995693
31	0,000955
32	1,011648
33	0,997817
33	4,978357
34	0,000010
34	1,029676
36	0,003710
36	1,011877
36	0,000352
37	0,997698
38	0,135904
39	0,000901
39	0,000901
39	0,995941
39	0,005601
40	0,000027
40	0,000027
40	0,000510
40	0,000578
40	0,996358
41	0,002783
41	0,005900
42	0,995161
42	0,005649
42	0,023379
43	1,990027
43	0,001172
43	0,994965
44	0,002284
44	0,000374
44	0,000374
45	0,015644
45	3,982140
46	0,000975
46	0,995093
46	0,995526
46	0,000580
46	0,000108
46	0,000135
46	0,002457
46	0,000530
47	0,000699
47	0,000699
47	0,000197
47	0,000197
47	0,000551
47	0,001394
47	0,000604
48	0,001023
49	0,000203
49	0,997097
49	0,000572
49	0,000572
49	0,000682
50	0,000438
50	0,000438
51	0,001853
52	0,000342
52	0,995845
52	0,000418
53	0,000354
53	0,995266
53	0,000845
53	0,000845
55	0,000645
55	0,000645
56	0,000186
56	0,000186
56	0,007142
57	0,018476
58	0,997085
59	0,007032
59	1,013283
59	0,000127
61	0,000078
63	0,001497
63	0,995868
64	0,994974
64	0,000021
69	0,000255
69	0,000255
69	0,011983
70	1,990466
71	0,000071
72	0,000388
72	0,013113
73	0,000870
73	0,000295
73	0,000295
82	0,994985
84	0,000246
88	0,000453
89	0,000123
89	0,000123

Tabulka 6: Rastrigova funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla 0,403755,
- směrodatná odchylka byla 0,811866,
- medián je 0,000965,
- modus je 0,000901,
- průměrný počet iterací k nalezení použitelného optima je 49,96.

Nejhorší experiment poskytl výsledek odlišný o 4,978357 od skutečné hodnoty.

3.2.7 Rosenbrockova funkce

Iterace	Hodnota
5	0,105921
10	0,000351
10	0,000587
15	0,026498
15	0,005187
16	0,004822
17	0,029708
19	0,087253
20	0,313449
20	0,000875
21	0,000078
21	0,000486
21	0,000486
22	0,037441
23	0,073184
23	0,367563
23	0,311952
23	0,008590
24	0,040855
24	0,004952
25	0,025170
25	0,029592
25	0,003468
27	0,000592
27	0,000951
27	0,000951
29	0,001684
29	0,218152
30	0,071446
30	0,001560
30	0,000390
30	0,081333
31	0,070829
31	0,002052
31	0,016006
31	0,019649
31	0,066746
31	0,003239
32	0,015642
32	0,114701
32	0,000821
33	0,003672
33	0,035994
33	0,012689
33	0,001006
33	0,004596
33	0,021213
34	0,143453
34	0,006472
34	0,115222
34	0,012018
34	0,001201
34	0,000837
34	0,139407
35	0,142462
35	0,000912
35	0,012333
35	0,065788
37	0,001553
38	0,046642
38	0,009499
39	0,000432
39	0,000432
39	0,000403
39	0,000297
39	0,000700
39	0,000057
39	0,097224
40	0,020207
40	0,036522
42	0,198897
43	0,002536
43	0,057232
43	0,003651
44	0,286368
44	0,096786
45	0,014198
45	0,096455
45	0,010503
45	0,000262
45	0,000262
46	0,105951
46	0,035135
46	0,148080
47	0,034111
47	0,035099
47	0,023325
48	0,034016
48	0,047251
50	0,027190
52	0,052517
53	0,000895
53	0,000895
54	0,002968
55	0,119934
57	0,011965
72	0,036074
73	0,035924
108	0,051571
109	0,051206

Tabulka 7: Rosenbrockova funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla 0,047197,
- směrodatná odchylka byla 0,072850,
- medián je 0,017828,
- modus je 0,000486,
- průměrný počet iterací k nalezení použitelného optima je 36,25.

Nejhorší experiment poskytl výsledek odlišný o 0,367563 od skutečné hodnoty.

3.2.8 Schwefelova funkce

Iterace	Hodnota
22	336,086550
23	118,452060
29	217,301590
30	0,000097
31	118,458880
32	0,032984
33	0,000241
35	0,000226
35	236,877820
37	217,143200
38	335,579300
38	0,000626
38	118,463326
39	118,445120
39	0,012622
40	0,000663
41	0,000130
41	236,880940
41	0,000147
42	0,000467
42	0,001530
42	118,440310
43	0,001955
43	0,000885
44	236,877290
45	0,001602
45	0,002357
45	217,140030
45	236,876860
46	118,438610
46	217,140700
46	118,442314
46	236,876750
46	0,001166
46	217,139850
47	118,438484
47	236,876880
47	0,000733
47	0,000733
47	118,438730
47	0,000108
47	0,000512
47	236,876740
47	0,000679
47	335,578900
48	118,438610
48	0,000249
48	0,000609
49	118,439110
49	0,001097
49	236,877100
49	118,439285
50	0,001440
50	0,000157
50	0,000157
50	236,878420
51	0,001079
51	118,438900
51	236,880280
52	118,438820
52	118,441690
53	236,876700
53	118,438940
54	118,452225
54	0,001173
54	118,438420
54	236,876710
55	0,000539
55	0,000828
55	0,000828
55	118,438390
56	0,000179
56	118,438515
57	0,000818
57	0,000818
57	118,438490
57	0,001197
57	0,000337
58	0,002716
58	0,000823
58	0,000823
62	236,877820
63	0,000502
63	236,877430
63	236,883090
64	0,000381
64	118,438700
64	118,438370
65	118,439430
65	118,438360
66	118,455010
66	118,439156
66	118,438770
68	217,157730
73	0,002623
81	118,438370
84	0,000779
85	118,438660
86	118,438420
90	118,450290

Tabulka 8: Schwefelova funkce - hodnoty testu

Uvedeme výsledky pro 100 experimentů:

- Střední hodnota byla 96,536721,
- směrodatná odchylka byla 99,48390681,
- medián je 118,438452,
- modus je 236,877820,
- průměrný počet iterací k nalezení použitelného optima je 50,92.

Nejhorší experiment poskytl výsledek odlišný o 336,086550 od skutečné hodnoty.

4 ZÁVĚR

Cílem mé práce bylo seznámit se s algoritmem PSO, následně implementovat algoritmus na testovací funkce a vytvořit aplikaci s uživatelským rozhraním.

Testování algoritmu na testovacích funkcích dopadlo úspěšně a ve většině případů bylo nalezeno použitelné optimum v rámci základní implementace. U některých složitějších testovacích funkcí, např. Eggholderova funkce, bych pro častější nalezení globálního optima doporučil úpravy koeficientů akcelerace. V případě že by úprava koeficientů byla nedostačující, bylo by vhodné zvýšit dobu explorace a následné zvýšení počtu iterací.

V budoucnu lze tuto práci dále rozvíjet z toho důvodu, že tento algoritmus lze použít ke strojovému učení, které je v moderní době velice perspektivní.

POUŽITÁ LITERATURA

[1] KENNEDY, James a Russell EBERHART. Particle swarm optimization: treaties and international agreements registered or filed and recorded with the Secretariat of the United Nations. *Proceedings of ICNN'95 - International Conference on Neural Networks* [online]. IEEE, 1995, 1987, 1942-1948 [cit. 2019-05-09]. DOI: 10.1109/ICNN.1995.488968. ISBN 0-7803-2768-3. Dostupné z: http://ieeexplore.ieee.org/document/488968/

[2] HEPPNER, Frank a Ulf GRENANDER. *A stochastic nonlinear model for coordinated bird flocks*. Washington, D.C., 1990. Dostupné také z: https://www.researchgate.net/profile/Frank_Heppner/publication/278411001_A_Stochastic_Non-Linear_Model_for_Bird_Flocking/links/5580a02908aea3d7096e4c8a.pdf. Vědecký článek. University of Rhode Island.

[3] MILLONAS, Mark. *Swarms, Phase Transitions, and Collective Intelligence*. Ithaca, New York, 1993. Dostupné také z: https://arxiv.org/abs/adap-org/9306002. Vědecký článek. Cornell university.

[4] SURJANOVIC, Sonja a Derek BINGHAM. Virtual Library of Simulation Experiments: Test Functions and Datasets [online]. Simon Fraser University: Bingham, 2013 [cit. 2019-05-07]. Dostupné z: https://www.sfu.ca/~ssurjano/.

[5] HEDAR, Abdel-Rahman. *Global Optimization Test Problems* [online]. Egypt: Hedar, 2006 [cit. 2019-05-07]. Dostupné z: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

[6] NĚMEC, František. *Particle Swarm Optimization: Optimalizace hejnem částic* [online]. Brno: Němec, 2018 [cit. 2019-05-07]. Dostupné z: http://www.fnemec.cz/projects/mgr/sem2/evo-pso-www/implementace.html.

[7] EBERHART, Russell C. a James KENNEDY. a *new optimizer using particle swarm theory* [online]. Nagoya: IEEE International Conference, 1995 [cit. 2019-05-07]. Dostupné z: https://ieeexplore.ieee.org/document/494215.

[8] VOLNÁ, Eva. *Evoluční algoritmy a neuronové sítě* [online]. Ostrava, 2012. Dostupné z: http://physics.ujep.cz/~mmaly/vyuka/MPVT_II/Heuristiky/SOMAdetail-Evolucni_algoritmy_a_neuronove_site+Genetika.pdf. Studijní opora. Ostravská univerzita v Ostravě.

[9] NĚMEČEK, Patrik. *Optimalizační úlohy na bázi částicových hejn (PSO)* [online]. Brno, 2014 [cit. 2019-05-07]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=119303. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Josef Schwarz.

[10] WANG, Dongshu, Dapei TAN a Lei LIU. *Particle swarm optimization algorithm: an overview* [online]. China: Soft Comput, 2017 [cit. 2019-05-07]. Dostupné z: https://doi.org/10.1007/s00500-016-2474-6.