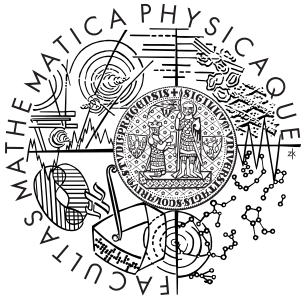
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

# BAKALÁŘSKÁ PRÁCE




Zdeněk Pátek

# Možnosti zvýšení výkonu přírodou inspirovaných globálních optimalizačních metod

Katedra Aplikované Matematiky

Vedoucí bakalářské práce: Mgr. Pavel Rytíř
Studijní program: Informatika, Obecná informatika

2009

Rád bych poděkoval Mgr. Pavlu Rytířovi za vedení této práce a své rodině za stálou podporu.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 27.5.2009 Zdeněk Pátek

# Obsah

Název práce: Možnosti zvýšení výkonu přírodou inspirovaných globálních optimalizačních metod
Autor: Zdeněk Pátek
Katedra: Katedra Aplikované Matematiky
Vedoucí bakalářské práce: Mgr. Pavel Rytíř
E-mail vedoucího: rytir@kam.mff.cuni.cz

Abstrakt: Tato práce se zabývá optimalizací funkcí reálných proměnných pomocí přírodou inspirovaných metod. Obsahuje popis vybraných globálních optimalizačních metod (Differential Evolution, Self-Organizing Migrating Algorithm, Steady-State Evolutionary Algorithm, Particle Swarm Optimization, Gregarious Particle Swarm Optimizer a Hybrid Particle Swarm with Differential Evolution Operator). Nalezl jsem čtyři zlepšení těchto metod, zjistil jejich vhodná nastavení parametrů a porovnal je na vybraných testovacích funkcích. Experimentální výsledky prokázaly, že popsaná zlepšení mohou zvýšit výkon přírodou inspirovaných optimalizačních metod.
Klíčová slova: globální optimalizace, evoluční metody, funkce reálných proměnných

Title: Possible improvements of global optimization methods inspired by nature
Author: Zdeněk Pátek
Department: Department of Applied Mathematics
Supervisor: Mgr. Pavel Rytíř
Supervisor's e-mail address: rytir@kam.mff.cuni.cz

Abstract: This study focuses on the global optimization of functions of real variables using methods inspired by nature. It contains a description of selected global optimization techniques (Differential Evolution, Self-Organizing Migrating Algorithm, Steady-State Evolutionary Algorithm, Particle Swarm Optimization, Gregarious Particle Swarm Optimizer a Hybrid Particle Swarm with Differential Evolution Operator). I have found four improvements of these techniques, discovered their suitable parameter configurations and compared them on chosen trial functions. Experimental results proved that described improvements can increase performance of the optimization techniques inspired by nature.
Keywords: global optimization, evolutionary methods, functions of real variables

# Kapitola 1

# Úvod

Optimalizace je proces výběru nejlepší varianty ze všech možných. Je nedílnou součástí naší společnosti již od jejího vzniku. Každý z nás se snaží optimalizovat své jednání při řešení každodenních problémů. Optimalizace se tak uplatňuje ve všech odvětvích lidské činnosti, a s příchodem výpočetní techniky se začala používat i tam. Dnes se optimalizace běžně využívá při návrhu výrobních postupů i výrobků a jejich částí na počítači. Stala se jednou z nejdůležitějších metod moderního průmyslu a výzkumu. Optimalizace však není výdobytkem lidské civilizace, hraje totiž důležitou roli i v přírodě – živočichové a rostliny musí optimalizovat svůj růst, vývoj a chování vzhledem k okolnímu prostředí, aby dokázali přežít.

V této práci se budu zabývat optimalizací pomocí výpočetních metod, které jsou inspirovány právě přírodou. Tyto metody se snaží napodobit některé (optimalizační) mechanismy, které dobře fungují v přírodě už milióny let. Od poloviny dvacátého století se výzkumníci nechávají inspirovat například přírodní evolucí, pohybem mravenců, chováním hejen ryb a ptáků nebo rozmnožováním rostlin. Za přispění těchto principů se podařilo vytvořit velice účinné optimalizační nástroje, které dobře slouží při řešení mnoha praktických problémů. S pokrokem vědy a techniky jsou však vědcům předkládány stále obtížnější problémy, a proto je nutné vyvíjet efektivnější optimalizační metody.

Tato práce se pokusí ukázat, že lze najít jednoduché a poměrně obecné způsoby, jak zvýšit výkon již existujících přírodou inspirovaných optimalizačních metod bez nutnosti podstatně měnit jejich původní principy. Navržená zlepšení popíši z teoretického hlediska, prověřím na testovacích funkcích a nakonec se pokusím shrnout jejich užitek. Názorná implementace všech představených metod a zlepšení a veškeré experimentální výsledky jsou k této práci přiloženy.

# Kapitola 2

# Přírodou inspirované globální optimalizační metody

## 2.1 Úvod do globální optimalizace

Globální optimalizace je část aplikované matematiky a numerické analýzy zabývající se optimalizací matematických funkcí, nazývaných účelové funkce.

**Definice.** Funkce $f : X \longmapsto Y$, kde $Y \subseteq \mathbb{R}$, je účelová funkce.

Definiční obor $X$ může obsahovat jakékoliv prvky, obor hodnot $Y$ je vždy podmnožina reálných čísel. Účelové funkce mohou být funkce vyjádřitelné analyticky, ale také například algoritmy, simulace nebo fyzická zařízení. Výsledkem optimalizačního procesu je vstup, na nichž zadaná účelová funkce dosahuje optimální (dle naší volby minimální nebo maximální) hodnoty. Takový vstup se nazývá globální optimum účelové funkce (tedy buď globální minimum nebo globální maximum účelové funkce).

**Definice.** Bod $\widehat{x} \in X$ je globálním minimem funkce $f : X \longmapsto Y$, pokud $\forall x \in X : \ f(\widehat{x}) \leq f(x)$.

**Definice.** Bod $\widehat{x} \in X$ je globálním maximem funkce $f : X \longmapsto Y$, pokud $\forall x \in X : \ f(\widehat{x}) \geq f(x)$.

V této práci se budu zabývat optimalizací účelových funkcí, jejichž definiční obor je podmnožina $\mathbb{R}^D$ (funkce reálných proměnných). Optima-

lizace takových funkcí může být problematická, zejména pokud obsahují více lokálních optim:

**Definice.** Funkce $f$ má v bodě $\widehat{x}$ lokální minimum, jestliže existuje okolí bodu $\widehat{x}$ takové, že pro všechny body $x$ z tohoto okolí platí $f(\widehat{x}) \leq f(x)$.

**Definice.** Funkce $f$ má v bodě $\widehat{x}$ lokální maximum, jestliže existuje okolí bodu $\widehat{x}$ takové, že pro všechny body $x$ z tohoto okolí platí $f(\widehat{x}) \geq f(x)$.

Lokální optima znesnadňují optimalizaci, protože z nich musí použitá optimalizační metoda umět uniknout, aby mohla nalézt skutečné globální optimum. Důsledkem velkého množství optimalizačních problémů je existence více přístupů ke globální optimalizaci funkcí reálných proměnných (ke každému jsem přidal několik konkrétních metod):

- Gradientní metody – Conjugate Gradient Method, Line Search, Newton's Method, Orthogonal Search
- Stochastické metody – Cross-Entropy Method, Harmony Search, Generalized Pattern Search, Mesh Adaptive Direct Search, Simulated Annealing
- Evoluční metody – Covariance Matrix Adaptation, Differential Evolution, Genetic Algorithms, Self-Organizing Migrating Algorithm, Steady-State Evolutionary Algorithm,
- Přírodou inspirované (neevoluční) metody – Ant Colony Optimization, Bacterial Foraging Optimization, Invasive Weed Optimization, Particle Swarm Optimization

Různé metody jsou vhodné na optimalizaci různých účelových funkcí. Přírodou inspirované (evoluční i neevoluční) metody jsou použitelné na široké spektrum funkcí, proto jsem se rozhodl zaměřit svou práci právě na ně. Popis účelových funkcí, které jsem vybral na optimalizaci, je ve třetí kapitole. V praxi je globální optimalizace používána v mnoha oborech, například:

- Bioinformatika
- Dobývání znalostí
- Ekonomie

- Elektrotechnika
- Chemické inženýrství
- Operační výzkum
- Strojírenství

O globální optimalizaci existuje velké množství literatury. Thomas Weise napsal vynikající a volně dostupnou knihu [1].

## 2.2 Základy přírodou inspirovaných globálních optimalizačních metod

Přírodou inspirované globální optimalizační metody jsou podstatnou součástí současného výzkumu v oboru optimalizace. Mezi jejich hlavní vlastnosti spadají robustnost, vysoký výkon, jednoduchá implementace, možnosti paralelizace, schopnost optimalizovat rozličné druhy účelových funkcí a fakt, že ke své práci nepotřebují žádné dodatečné informace o optimalizované funkci. Jsou schopné optimalizovat funkce s velkým definičním oborem, spojité i nespojité, unimodální i multimodální, separabilní i neseparabilní (popis těchto vlastností se nachází ve třetí kapitole).

Klíčovým pojmem přírodou inspirovaných optimalizačních metod je populace. Populace je množina jedinců, kde každý jedinec reprezentuje jedno řešení (množinu vstupních proměnných) účelové funkce. Na jedince lze také nahlížet jako na bod v definičním oboru účelové funkce. Dále má každý jedinec svou hodnotu – hodnota jedince je rovna hodnotě účelové funkce v bodě, který je reprezentován tímto jedincem. Samotná optimalizace je iterační proces. Nejprve je vytvořena počáteční populace jedinců (například jako množina náhodně generovaných bodů v definičním oboru účelové funkce). V každé iteraci dochází ke vzniku nových jedinců nebo k pohybu jedinců prostorem s cílem je zlepšit (tedy nalézt optimálnější body v definičním oboru účelové funkce). Toto zlepšování probíhá dle rovnic a postupů, které jsou v každé metodě odlišné, a proto se různé metody chovají jinak a na odlišných účelových funkcích jsou různě úspěšné. Optimalizační proces je zastaven až po splnění ukončovacího kritéria – například po uplynutí přiděleného času, po provedení zadaného počtu iterací nebo po konvergenci populace do jediného bodu v definičním oboru účelové funkce. Celý optimalizační mechanismus evolučních metod lze shrnout do následujícího schématu (neevoluční metody mají svá vlastní schémata, často však podobná):

Inicializuj populaci $x$
**while** *ukončovací kritérium není splněno* **do**
    $y \leftarrow$ Vytvoř populaci potomků za použití $x$
    Ohodnoť $y$
    $x \leftarrow$ Vyber novou populaci z $x$ a $y$
**end**
Vrať pozici nejlepšího jedince z $x$

Optimalizace pomocí přírodou inspirovaných metod ovšem nezaručuje nalezení skutečného globálního optima – je možné, a často se stává, že se optimalizační metoda zastaví v lokálním optimu a není schopná z něj uniknout. Navíc v praxi typicky není možné zjistit, zda je nalezeno globální optimum nebo pouze lokální. Definiční obory účelových funkcí mohou být obrovské, a proto není možné je prohledávat celé dostatečně jemně. Z tohoto důvodu je často nutné se spokojit s dostatečně dobrým, i když ne optimálním, řešením. Velkému množství aplikací totiž stačí nalezení suboptimálního řešení, pokud je ho dosaženo v rozumném čase.

Optimalizační proces je řízen parametry metody – ty mohou být pevně zvolené nebo se mohou měnit v průběhu optimalizace. Každá metoda má své vlastní parametry, mezi nimi je však téměř vždy počet jedinců v populaci (velikost populace). Parametry je potřeba správně nastavit, aby byla optimalizace co nejvýkonnější. Vzhledem k tomu, že vhodné nastavení závisí na účelové funkci, neexistují univerzální hodnoty těchto parametrů. Popisem vybraných přírodou inspirovaných globálních optimalizačních metod se zabývá následující text.

## 2.3 Differential Evolution

Differential Evolution (DE) [2] je evoluční optimalizační metoda vyvinutá Kennethem V. Pricem a Rainerem Stornem v roce 1997. Ideově vychází z genetických algoritmů a simulovaného žíhání. Již první publikovaná verze byla velmi úspěšná na testovacích problémech, a brzy byla DE akceptována vědeckou komunitou jako silný nástroj na globální optimalizaci. Od té doby vznikly o DE dvě vetší publikace [3, 4], vysvětlující DE z teoretického i aplikačního hlediska, a velké množství článků popisujících její rozličné varianty. Metoda byla v praxi úspěšně použita na mnoho optimalizačních problémů, zde uvádím pouze skromný výběr:

- Inverzní fraktální problém
- Nastavení fuzzy regulátoru

- Optimalizace sítě rozhlasových vysílačů
- Optimalizace tlakové nádoby, ozubeného převodu a pružiny
- Polynomial fitting problem
- Učení neuronové sítě.

Metoda DE využívá evoluční schéma popsané v předcházející části. Vstupem uživatele je účelová funkce včetně definičního oboru dimenze $D$ a parametry $NP$ (počet jedinců v populaci), $F$ (mutační konstanta) a $CR$ (práh křížení). Počáteční inicializace jedinců v populaci je stejně jako ve všech následujících metodách náhodná (každý jedinec zaujme náhodné místo v definičním oboru účelové funkce). V každé iteraci je provedena mutace a křížení všech jedinců tak, že každý jedinec z populace má právě jednoho potomka. Při vzniku potomka z i-tého jedince ($x_i$) se nejprve uskuteční mutace – náhodně jsou vybráni tři různí jedinci $x_1$, $x_2$, $x_3$ z celé populacea a z nich je vytvořen mutační vektor $v$ dle rovnice:

$$v = x_1 + F \cdot (x_2 - x_3)\,.$$

Nyní je provedeno křížení, které zkombinuje i-tého (zatím nepoužitého) jedince s mutačním vektorem. Pro každou složku nového jedince se rozhodne, zda se má rovnat příslušné složce i-tého jedince nebo příslušné složce mutačního vektoru. Toto rozhodování probíhá porovnáním konstanty $CR$ s náhodně vygenerovaným číslem z intervalu $\langle 0, 1\rangle$ – když je konstanta $CR$ větší, je použita složka z i-tého jedince, jinak složka z mutačního vektoru. Algoritmicky lze křížení z i-tého jedince popsat:

**for** $d \leftarrow 1$ *to* $D$ **do**
  **if** $CR > rand_{0,1}$ **then**
    $y_{id} \leftarrow x_{id}$
  **else**
    $y_{id} \leftarrow v_{id}$
  **end**
**end**

Zde $D$ je dimenze definičního oboru účelové funkce, $y_i$ je nový jedinec (potomek jedince $x_i$). Křížení zajistí, že část informace i-tého jedince bude zachována v jeho potomkovi. Dále je kvalita každého potomka ohodnocena účelovou funkcí. Do další iterace jsou jedinci vybráni v selekčním procesu – z každé dvojice (rodič, potomek) se do nové populace dostane ten, který má lepší hodnotu účelové funkce. Optimalizační proces probíhá, dokud není splněno ukončovací kritérium, typicky do provedení

předem stanoveného počtu iterací. Díky upřednostňování lepších jedinců při selekci se v průběhu optimalizace celá populace zlepšuje, jedinci se k sobě přibižují a umožňují jemnější prohledávání prostoru v okolí nalezených lokálních optim. Po určitém počtu iterací se celá populace dostane do jednoho optima a optimalizační proces se zastaví. Tomuto mechanismu se také říká konvergence populace. Nakonec je jedinec zaujímající nejoptimálnější pozici vydán jako nejlepší nalezené řešení.

Průběh optimalizace lze ovlivňovat změnou parametrů $NP$, $F$ a $CR$ a počtem provedených iterací. Zvýšením $NP$ (počtu jedinců v populaci) a počtu iterací lze často docílit lepších výsledků za cenu zvýšení doby optimalizace (pro správné fungování mutačního mechanismu jsou potřeba nejméně čtyři jedinci). Parametr $F$ je doporučeno nastavit z intervalu $\langle 0, 2\rangle$. Parametr $CR$ lze nastavit z intervalu $\langle 0, 1\rangle$. Při vysokém parametru $CR$ je velký rozdíl mezi potomky a rodiči – prohledávání prostoru je efektivnější, ale může vést k předčasné konvergenci do lokálního optima. Naopak příliš nízká hodnota může evoluci brzdit.

V předchozí části byla ukázána pouze jedna z původních variant (strategií) DE – DE/rand/1/bin. Zde rand znamená, že základem každého mutačního vektoru je náhodný jedinec (jiné možnosti jsou best, kde je základem nejlepší jedinec populace, a rand-to-best, ve které dochází k posunu jedince směrem k nejlepšímu jedinci), následující číslo udává počet přičítaných diferencí náhodných jedinců (typicky jedna nebo dvě) a bin je druh křížení (bin připomíná uniformní křížení v genetických algoritmech, druhou možností je exp, které je podobné jednobodovému křížení). Výběrem vhodné strategie na danou účelovou funkci lze zvýšit rychlost nalezení řešení nebo jeho kvalitu. Proto byly vyvinuty další strategie a modifikace metody DE, například Either-Or [3] a Trigonometric Mutation Operation [5].

## 2.4 Self-Organizing Migrating Evolution

Self-Organizing Migrating Evolution (SOMA) [6] je optimalizační metoda vyvinutá Ivanem Zelinkou v roce 2001. SOMA byla aplikována například na eliminaci chyb měření stavu plazmatu v plazmovém reaktoru a optimalizaci řízení chemického reaktoru. Průběh optimalizace metodou SOMA je řízen parametry $PopSize$ (velikost populace), $Mass$ (délka pohybu), $Step$ (velikost kroku) a $PRT$ (perturbace). Obdobně jako v DE probíhá optimalizace po iteracích. V každé iteraci metody (nazývané migration loop) je každý jedinec postupně přesouván do nových pozic, jak ukazuje následující algoritmus:

```
for s ← 0; s < Mass; s ← s + Step do
    for d ← 1 to D do
        if PRT > rand_{0,1} then
            y_id ← x_id + s · (leader_d − x_id)
        end
    end
    Zkontroluj y_i
    Ohodnoť y_i
end
```

$D$ je dimenze definičního oboru účelové funkce, $x_i$ je původní pozice i-tého jedince, $y_i$ je jeho nová pozice. *leader* je jedinec zvolený z populace dle strategie metody SOMA – může se jednat o nejlepšího jedince populace (strategie AllToOne) nebo o náhodně vybraného jedince (strategie AllToOneRand). Každý jedinec se tedy posouvá směrem k *leaderovi* po krocích daných parametrem *Step*. Tento pohyb probíhá pouze v některých dimenzích díky parametru $PRT$ (ten má podobnou funkci jako $CR$ v DE). Všechny pozice, které jedinec navštíví (včetně jeho počáteční pozice), jsou ohodnoceny účelovou funkcí, a finální pozice jedince v nové populaci je ta, která dostala nejlepší ohodnocení. Optimalizace probíhá, dokud není splněno ukončovací kritérium. Nejlepší nalezené řešení zadané účelové funkce je vráceno uživateli.

Správným nastavením parametrů lze podstatně zlepšit chování metody SOMA. Zvýšení $PopSize$ může vést k nalezení lepšího řešení, ale optimalizace bude probíhat déle. Parametr *Step* je vhodné nastavit na hodnotu z intervalu $\langle 0,1; 0,9\rangle$. Snížením *Step* se budou jedinci pohybovat po menších krocích, prohledávání prostoru bude jemnější, ale doba optimalizace vyšší. $Mass$ lze nastavit na hodnotu z intervalu $\langle 1, 3\rangle$. Pomocí nastavení $Mass$ nad 1 se budou jedinci pohybovat až za *leadera*, což může zvýšit šanci na nalezení globálního optima i urychlit jeho nalezení, opět za cenu vyšší doby optimalizace. Parametr $PRT$ omezuje pohyb jedinců na některé náhodně vybrané dimenze (při přesunu jedince směrem k *leaderovi* se bude jedinec v některých dimenzích pohybovat a v některých zůstane na původní pozici). Příliš nízká hodnota může zpomalit optimalizační proces, příliš vysoká může způsobit konvergenci populace do lokálního optima.

Pro úplnost je nutné dodat, že existují alternativní strategie AllToAll, AllToAllAdaptive [6], AllToOneRand with Binary Tournament, AllToOne with Adaptive Leader a AllToOne with Adaptive Leader and Hypermutation [7]. Dále byly vyvinuty modifikace umožnující pracovat s diskrétními obory účelových funkcí a omezeními [6].

## 2.5 Steady-State Evolutionary Algorithm

Steady-State Evolutionary Algorithm [8] je optimalizační mechanismus popsaný Z. Chenem a K. Lishanem v roce 2005, inspirovaný DE. Zde se budu zaobírat první fází jejich metody, protože druhá fáze je pouze lokální prohledávání prostoru, které zpřesňuje nalezené optimum. Jediným parametrem metody je velikost populace. Stejně jako v DE vzniká každou iteraci nová populace. Noví jedinci jsou vytváření pomocí operátoru Multi-parent Crossover (MPCO) – nejprve je náhodně vybráno $n$ jedinců $x_1$, $x_2$, ..., $x_n$ z celé populace ($n$ doporučují autoři zvolit 3 nebo 4). Mezi nimi je nalezen jedinec $x_w$, který je nejhůře hodnocený účelovou funkcí. Z těchto jedinců je vypočten střed $c$ podle:

$$c = \frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - x_w) .$$

Nový jedinec je pak vytvořen dle rovnice:

$$y_w = x_w + 2 \cdot (c - x_w) .$$

V nové populaci nahradí vytvořený jedinec $y_w$ svého nejhoršího rodiče $x_w$ jen pokud je jeho hodnota účelové funkce lepší. Optimalizace probíhá, dokud není splněna ukončovací podmínka. Opět je možné dosáhnout lepších výsledků za cenu prodloužení výpočetní doby zvýšením velikosti populace nebo počtu iterací.

## 2.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO) [9] je výkonná optimalizační metoda, představená Jamesem Kennedym a Russellem Eberhartem v roce 1995. Při vzniku PSO se tvůrci inspirovali pohybem živočichů v přírodě, konkrétně chováním hejen ptáků a ryb při hledání potravy, a vlastní optimalizační metoda vychází z pokusů o jejich simulaci. Od svého vzniku byla PSO podrobně analyzována a vzniklo velké množství variant. Proto je dobře použitelná v praxi, například:

- Hledání motifů v proteinových sekvencích
- Návrh antény
- Predikce vývoje finančního trhu

- Rozpoznávání vzorů
- Učení neuronové sítě
- Zarovnávání DNA sekvencí.

PSO je od výše uvedených metod značně odlišná – jedinci obsahují dodatečné informace kromě pozice a ohodnocení účelovou funkcí a neexistuje zde žádná obdoba selekce. Informace v i-tém jedinci zahrnuje aktuální pozici v prostoru ($x_i$) a její ohodnocení, nejlépe hodnocenou pozici nalezenou tímto jedincem ($b_i$) a její ohodnocení a rychlost ($v_i$) – rychlost je chápána jako vektor stejné dimenze jako $x_i$. Parametry PSO jsou $PopSize$ (velikost populace), $\omega$ (setrvačnost), $c_1$ (kognitivní konstanta), $c_2$ (sociální konstanta) a $V_{max}$ (maximální rychlost). Pozice i rychlost jedinců je inicializována náhodně. Po inicializaci je spuštěn iterační proces, ve kterém jedinci mění svojí rychlost a pozici ve snaze najít globální optimum. Rychlost každého jedince je upravena dle vzorce:

$$v_i \leftarrow \omega \cdot v_i + c_1 \cdot r_1 \cdot (b_g - x_i) + c_2 \cdot r_2 \cdot (b_i - x_i)\,,$$

kde $b_g$ je nejlepší nalezená pozice, $r_1$ a $r_2$ jsou vektory stejné dimenze jako $v_i$ složené z náhodných čísel z rovnoměrného rozdělení na $\langle 0, 1\rangle$. Násobení vektorů probíhá vždy po složkách. Nyní rozeberu vzorec po jednotlivých částech:

- Část $\omega \cdot v_i$ zajišťuje, že je využita informace o rychlosti jedince v minulé iteraci, jedná se o jakousi setrvačnost jedince. Hodnota parametru $\omega$ je pevně nastavená z intervalu $\langle 0, 1\rangle$ nebo se v průběhu optimalizace mění.
- Část $c_1 \cdot r_1 \cdot (b_g - x_i)$ mění rychlost jedince v závislosti na pozici $b_g$ (nejlepšího nalezená pozice). Při kladném $c_1$ se bude jedinec přibližovat směrem k nejlepšímu jedinci, což zajišťuje konvergenci PSO. $r_1$ zavádí do procesu náhodu (změna rychlosti bude v některých dimenzích větší než v jiných).
- Část $c_2 \cdot r_2 \cdot (b_i - x_i)$ mění rychlost jedince v závislosti na pozici $b_i$ (nejlepší pozice nalezená i-tým jedincem). Při kladném $c_2$ se bude jedinec vracet do své nejlepší nalezené pozice, což snižuje rychlost konvergence. $r_2$ opět zavádí do procesu náhodu.

Pokud rychlost některého jedince přesáhne v absolutní hodnotě $V_{max}$, je jeho rychlost patřičně snížena. Díky tomuto mechanismu je snažší

udržet všechny jedince uvnitř definičního oboru účelové funkce, což se projeví zejména na počátku optimalizačního procesu (na začátku jsou jedinci daleko do sebe a jejich rychlost je vysoká). Po změně rychlosti se všichni jedinci přesunou na nové pozice dle:

$$x_i \leftarrow x_i + v_i.$$

Nakonec jsou nové pozice všech jedinců ohodnoceny a optimalizace pokračuje další iterací, dokud není naplněno ukončovací kritérium. Shrnutí iterační části metody:

**while** *ukončovací podmínka není splněna* **do**
  Aktualizuj nejlepší pozici $b_g$
  **for** $i \leftarrow 1$ *to* $PopSize$ **do**
    $v_i \leftarrow \omega \cdot v_i + c_1 \cdot r_1 \cdot (b_g - x_i) + c_2 \cdot r_2 \cdot (b_i - x_i)$
    Zkontroluj $v_i$ pomocí $V_{max}$
  **end**
  **for** $i \leftarrow 1$ *to* $PopSize$ **do**
    $x_i \leftarrow x_i + v_i$
    Zkontroluj $x_i$
    Ohodnoť $x_i$
  **end**
**end**

Správné nastavení parametru $\omega$ bylo identifikováno jako nejdůležitější pro dobré fungování metody. Původní návrh počítal s pevně zvoleným parametrem $\omega$, ale brzy se zjistilo, že je vhodnější měnit $\omega$ dynamicky v průběhu optimalizace. Y. Shi a R. Eberhart doporučují nastavit počáteční hodnotu $\omega$ na 0,9 a lineárně ji snižovat až do 0,4 [10]. Tento postup zajišťuje rychlé prohledávání prostoru na počátku optimalizace díky vyšší rychlosti jedinců a pomalejší a jemnější ke konci optimalizace. Jinou účinnou možností je nastavovat $\omega$ v každé iteraci náhodně z intervalu $\langle 0, 1\rangle$ [11]. Parametry $c_1$ a $c_2$ je vhodné nastavit okolo 2 [12]. Parametr $V_{max}$ zabraňuje nadměrné rychlosti jedinců, která by mohla způsobit jejich časté přesuny mimo definiční obor účelové funkce. Podrobnému nastavení parametrů PSO se věnují [10, 12, 13]. K metodě PSO existují desítky variant upravených pro různé druhy problémů. Dvě z nich popisuji v následujícím textu – Gregarious Particle Swarm Optimizer [14] a Hybrid Particle Swarm with Differential Evolution Operator [15].

## 2.7 Gregarious Particle Swarm Optimizer

Gregarious Particle Swarm Optimizer (G-PSO) [14] je modifikace metody PSO vyvinutá S. Pasupuletim a R. Battitim v roce 2006. Parametry jsou $\gamma$ (počáteční krok), $\epsilon$ (bias), $\gamma_{min}$ (nejmenší krok), $\gamma_{max}$ (největší krok), $\sigma$ (velikost změny kroku) a $V_{max}$ (maximální rychlost). Stejně jako v PSO obsahují jedinci rychlost, nikoliv však nejlepší nalezenou pozici (pouze aktuální). V každé iteraci se rychlost i-tého jedince změní následovně:

**if** $distance(x_i,\ x_g) \leq \epsilon$ **then**
    $v_i \leftarrow r_{(-V_{max},V_{max})}$
**else**
    $v_i \leftarrow \gamma \cdot r_{(0,1)} \cdot (x_g - x_i)$
**end**

$distance(x_i,\ x_g)$ je euklidovská vzdálenost mezi i-tým a nejlepším jedincem, $r_{(-V_{max},V_{max})}$ je náhodná rychlost z intervalu povolených rychlostí a $r_{(0,1)}$ je vektor náhodných čísel z intervalu $\langle 0, 1\rangle$. Oproti PSO se tedy rychlost jedince mění zcela jinak, změna závisí pouze na aktuální pozici jedince a na pozici nejlepšího jedince. Pozice jedince se stejně jako v PSO mění přičtením vektoru rychlosti. Na konci každé iterace je provedena změna parametru $\gamma$ dle:

**if** $f(x_g) < f(x_g)$ **then**
    $\gamma \leftarrow max(\gamma - \sigma, \gamma_{min})$
**else**
    $\gamma \leftarrow min(\gamma + \sigma, \gamma_{max})$
**end**

Tento mechanismus umožnuje dynamicky měnit $\gamma$ a tím chování metody v závislosti na aktuálním stavu optimalizace. Pokud se od poslední iterace nenašla žádná lepší pozice, $\gamma$ se zvýší o $\sigma$, to zajistí vetší změnu rychlosti v následující generaci, což dovolí jedincům uniknout z lokálního minima. Pokud naopak byla nalezena lepší pozice, $\gamma$ se sníží o $\sigma$, rychlost jedinců se v následující iteraci sníží a prohledávání bude lokálnejší, což umožní nalezení přesnějšího optima. Parametry $\gamma_{min}$ a $\gamma_{max}$ slouží proti snížení rychlosti na 0 a proti jejímu nekontrolovanému růstu. Autoři doporučují nastavit $\sigma$ na 0,5, počáteční $\gamma$ na 3, $\gamma_{min}$ na 2 a $\gamma_{max}$ na 4. Funkcí parametru $\epsilon$ je radikální změna rychlosti jedinců, kteří se příliš přiblíží k nejlepšímu jedinci – to účinně brání predčasné konvergenci.

## 2.8 Hybrid Particle Swarm with Differential Evolution Operator

Hybrid Particle Swarm with Differential Evolution Operator (DEPSO) [15] je hybridní metoda představená W.-J. Zhangem a X.-F. Xiem roku 2003. Hybridita znamená v reči evolučních algoritmů sloučení dvou a více metod do jedné, často kvůli zvýšení robustnosti. Zde autoři využívají metody PSO a DE. Jejich spojení je vyřešeno jednoduše – v lichých iteracích se budou jedinci chovat jako v PSO, v sudých jako v DE. Jedinci tedy obsahují i rychlost, ta je však využita pouze v lichých iteracích. Poslední jemnou modifikací je pevné nastavení mutačního parametru $F$ (platí pro část DE) na 0,5.

# Kapitola 3

# Zlepšování optimalizačních metod

## 3.1 Možnosti zlepšení

Rostoucí využívání přírodou inspirovaných optimalizačních metod s sebou přináší nové obtížné optimalizační problémy, na něž je potřeba najít použitelnou optimalizační metodu. To je hlavní motivací dalšího výzkumu těchto metod a jejich modifikací. Po výběru vhodné metody pro konkrétní aplikaci se však zjevuje další otázka – je možné dále zvýšit výkon metody, aniž by bylo porušeno její základní schéma? V následujících částech této práce budu popisovat, jak toho dosáhnout pomocí těchto postupů:

- Náhodná změna
- Perturbace
- Sekvenční schéma
- Zavedení predátora

Výhodou výše vypsaných vylepšení je fakt, že se jedná o obecnější principy, které je možné aplikovat na různé přírodou inspirované optimalizační metody. Některá z těchto zlepšení mohou být silnější než jiná, což je také ovlivněno zadanou účelovou funkcí. Vylepšení budu testovat na optimalizačních metodách popsaných v předchozí kapitole (DE, SOMA, SEA, PSO, G-PSO a DEPSO) a účelových funkcích popsaných v následující podkapitole. Všechny testované metody a jejich zlepšení jsem implementoval do přiloženého programu – ten poskytnul veškeré experimentální výsledky, které budu v dalším textu používat.

## 3.2 Účelové funkce a jejich vlastnosti

Účelové funkce realných proměnných mají různé vlastnosti. Některé z nich podstatně zvyšují obtížnost optimalizace. Účelové funkce mohou být:

- Unimodální / multimodální – unimodální funkce má jediný lokální (a tedy i globální) extrém, naopak multimodální má více lokálních extrémů. Multimodální funkce jsou typicky obtížnější na optimalizaci, protože je nutné zajistit, aby se optimalizační metoda nezastavila už v lokálním optimu.

- Separovatelné / neseparovatelné – separovatelnou funkci lze rozložit na několik částí (typicky rozkladem definičního oboru po dimenzích), které lze optimalizovat nezávisle, a platí, že sloučením globálních optim všech částí je získáno globální optimum celé funkce. Neseparovatelnost postatně zvyšuje obtížnost optimalizace.

- Lineární / nelineární – lineární funkce obsahují pouze lineární členy a lze je snáze optimalizovat.

- S různým počtem proměnných – s rostoucí dimenzí definičního oboru většinou roste obtížnost, ale existují i výjimky (Griewangkova funkce [18]).

- Patologické – tímto termínem jsou označovány funkce, které jsou záměrně vytvořené tak, aby na nich optimalizační metody selhaly. Může se například jednat o funkci typu „jehla v kupce sena“ – funkce je v grafickém smyslu plochou rovinou a její extrém je jen úzkým „zářezem“ v této rovině. Takovou funkci je velice obtížné optimalizovat jakoukoliv metodou a nalezení globálního optima je jen otázkou náhody při inicializaci a v průběhu optimalizace.

- Klamné – funkce navržené tak, aby se snažily oklamat optimalizační algoritmus (například mohou mít dobré, lehce dosažitelné lokální optimum daleko od skutečného globálního optima – pro optimalizační metodu je pak obtížné z lokálního optima uniknout).

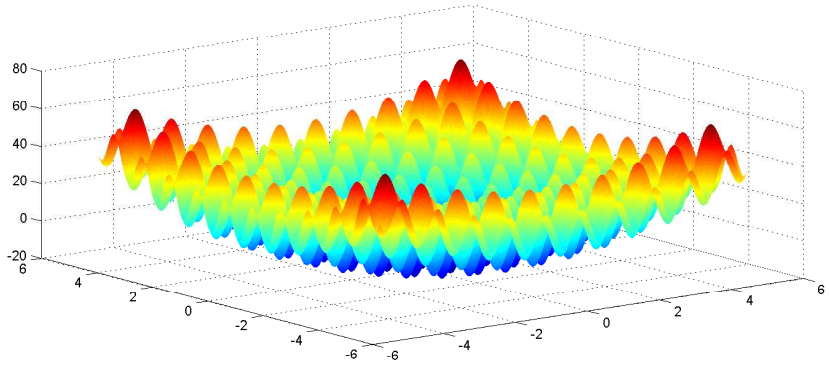
Pro testování optimalizačních metod se většinou používají určité účelové funkce [16, 17, 18]. Výhodou tohoto přístupu je, že je možné porovnávat výsledky s jinými optimalizačními metodami. Navíc jsou tyto funkce navržené tak, aby se dostatečně prověřila robustnost použité metody, a typicky jsou rozšiřitelné na libovolný počet dimenzí definičního oboru

(škálovatelnost). Tím lze uměle zvyšovat nároky na jejich optimalizaci. V této práci budu optimalizovat (vždy se jedná o minimalizaci) následující funkce:

**Rastrigin's Function**
Tato funkce je vysoce multimodální, ale separabilní. Velké množství lokálních extrémů může způsobovat některým metodám problémy. Analytický vzorec:

$$f(x) = \sum_{i=1}^{D} \left(x_i^2 - 10\,\cos\left(2\,\pi\,x_i^2\right)\right)$$

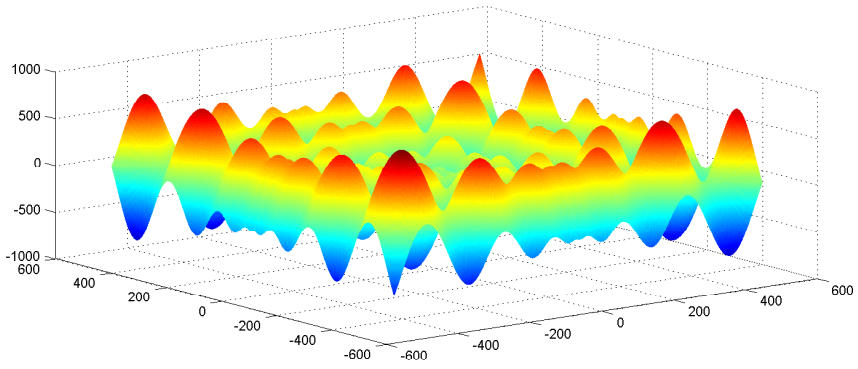
Definiční obor je omezen na $\langle -5, 12, 5, 11\rangle$ v každé dimenzi. Dvourozměrný případ funkce:




**Schwefel's Function**
Funkce je multimodální a separabilní. Specifickou vlastností je umístění globálního extrému na kraji definičního oboru. Analytický vzorec:

$$f(x) = \sum_{i=1}^{D} -x_i\,\sin\left(\sqrt{|x_i|}\right)$$

Definiční obor je omezen na $\langle -512, 511\rangle$ v každé dimenzi. Dvourozměrný případ funkce:

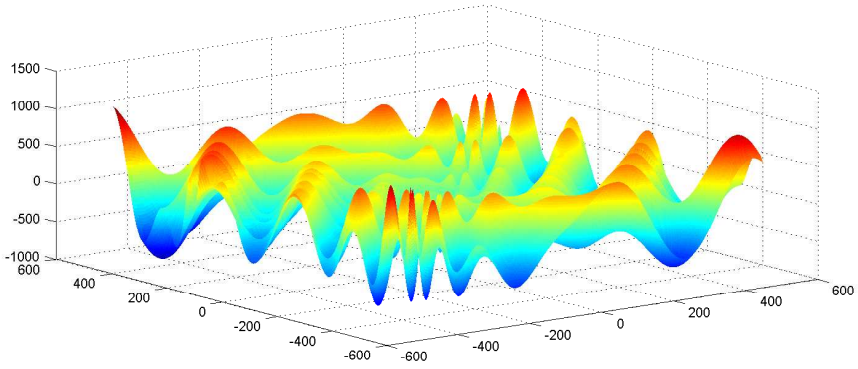



**Egg Holder Function**

Tato funkce je multimodální a neseparabilní. Globální extrém je na kraji definičního oboru. Analytický vzorec:

$$f(x) = \sum_{i=1}^{D-1} \left( -(x_{i+1} + 47) \sin\left( \sqrt{\left| x_{i+1} + \frac{x_i}{2} + 47 \right|} \right) - x_i \sin\left( \sqrt{|x_i - x_{i+1} - 47|} \right) \right)$$

Definiční obor je omezen na $\langle -512, 512 \rangle$ v každé dimenzi. Dvourozměrný případ funkce:

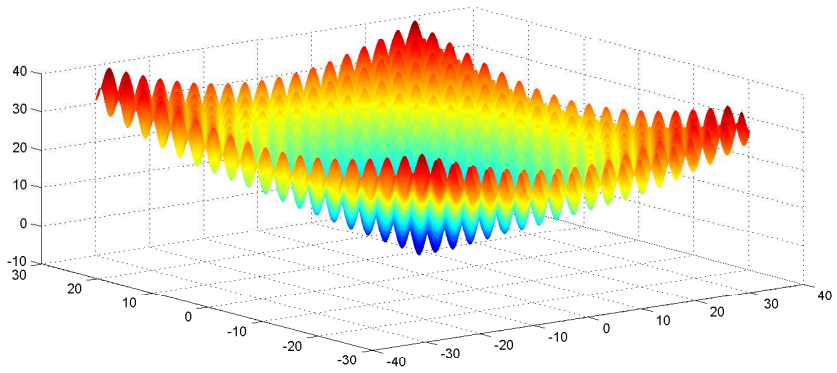



**Ackley's Function**

Tato funkce je rovněž multimodální a neseparabilní. Analytický vzorec:

$$f(x) = \sum_{i=1}^{D-1} \left( 3\,(\cos(2\,x_i) + \sin(2\,x_i)) + \frac{\sqrt{x_{i+1}^2 + x_i^2}}{e^{0.2}} \right)$$

Definiční obor je omezen na $\langle -30, 30 \rangle$ v každé dimenzi. Dvourozměrný případ funkce:




Definiční obory popsaných účelových funkcí se shodují s nejčastěji používanými v odborné literatuře. Další často používané účelové funkce se dají najít na internetových stránkách Ivana Zelinky [19].

## 3.3 Srovnání výkonu vybraných optimalizačních metod

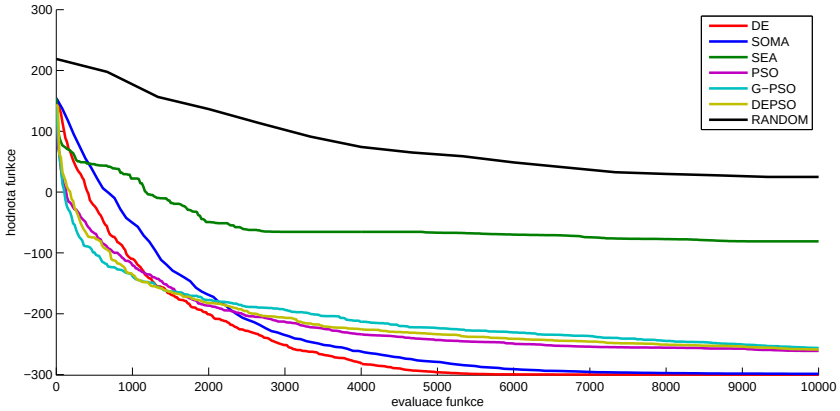
V této kapitole porovnávám výkon globálních optimalizačních metod popsaných v této práci. Testování metod (a poté i jejich zlepšení) probíhalo na účelových funkcích Rastrigin's Function, Schwefel's Function, Egg Holder a Ackley's Function, vždy se jednalo o minimalizaci (nižší hodnota účelové funkce je lepší). Definiční obor každé účelové funkce má 30 dimenzí, v každé dimenzi je omezen na stejný interval (ten je uveden v předchozí části u každé funkce). Porovnávat výkon optimalizačních metod lze více způsoby – já každé metodě dovolím provést pevný počet ohodnocení jedince účelovou funkcí a srovnám nejlepší nalezená řešení. Počet povolených ohodnocení jsem zvolil 10 000. Díky závislosti pouze na počtu ohodnocení je možné experimenty zopakovat se stejným výsledkem i na jiné hardwarové konfiguraci. Aby se vyloučil příliš velký vliv náhody v optimalizačním procesu, byla každá metoda spuštěna na každé účelové funkci desetkrát (se zafixovanými vstupy generátoru náhodných čísel). Ve výsledcích ukazuji vždy průměr ze všech deseti běhů. Pro srovnání jsem otestoval také náhodné hledání (generování náhodných bodů z de-

finičního oboru účelové funkce a jejich ohodnocování, ve výsledcích pod zkratkou RANDOM). Výsledky původních metod (zatím tedy bez zlepšení) jsou v Tabulce 1 – nejlepší výsledky jsou zvýrazněny tučně:

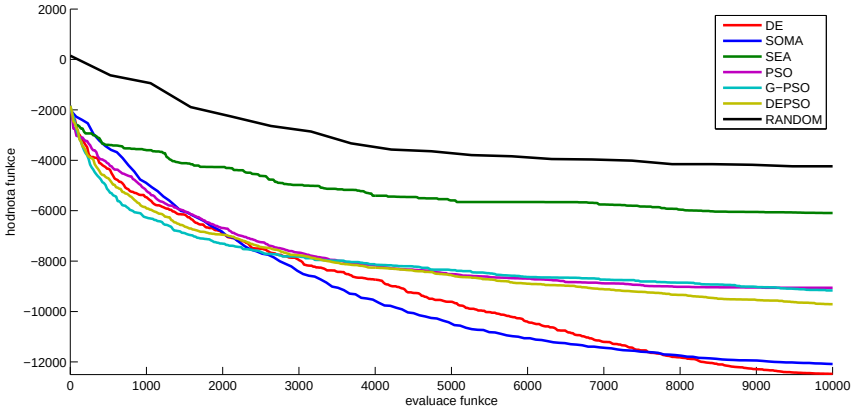
| Metoda | Rastrigin's F. | Schwefel's F. | Egg Holder F. | Ackley's F. |
|---|---|---|---|---|
| RANDOM | 24,89933 | -4239,3404 | -6189,8079 | 332,39947 |
| DE | **-299,99994** | **-12474,975** | -13674,671 | **-85,398867** |
| SOMA | -298,86712 | -12084,987 | **-15956,97** | -75,879108 |
| SEA | -80,975342 | -6093,246 | -7805,9872 | 219,29208 |
| PSO | -261,18227 | -9059,6876 | -11688,545 | -44,317431 |
| G-PSO | -256,6445 | -9164,0927 | -13608,588 | 49,781268 |
| DEPSO | -259,21233 | -9709,3126 | -12446,652 | -36,95492 |

Tabulka 1: Srovnání globálních optimalizačních metod

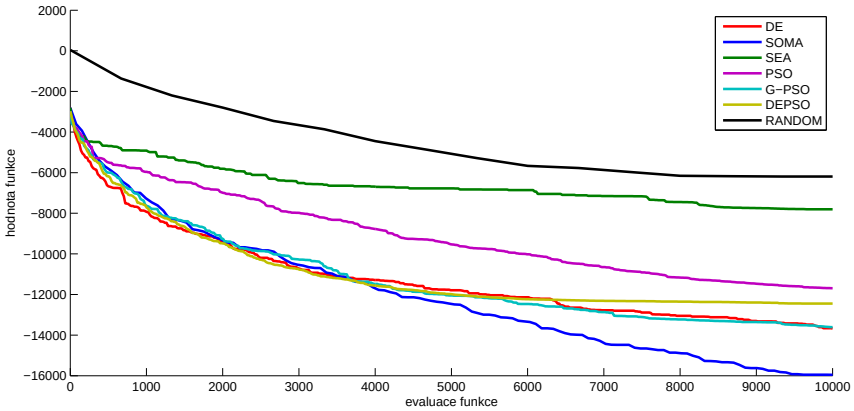
Z tabulky je vidět, že SEA dává na všech testovacích funkcích nejhorší výsledky (nepočítám-li náhodné hledání), naopak DE je nejlepší na třech účelových funkcích. Průběh optimalizace v průměrném případě je znázorněn na následujících grafech. Vodorovná osa udává počet ohodnocení účelovou funkcí, svislá osa nejlepší dosaženou hodnotu účelové funkce. Z grafů lze například vyvodit, že metoda G-PSO je schopná najít velmi rychle relativně dobré lokální optimum, ale následně má problémy z něj uniknout.



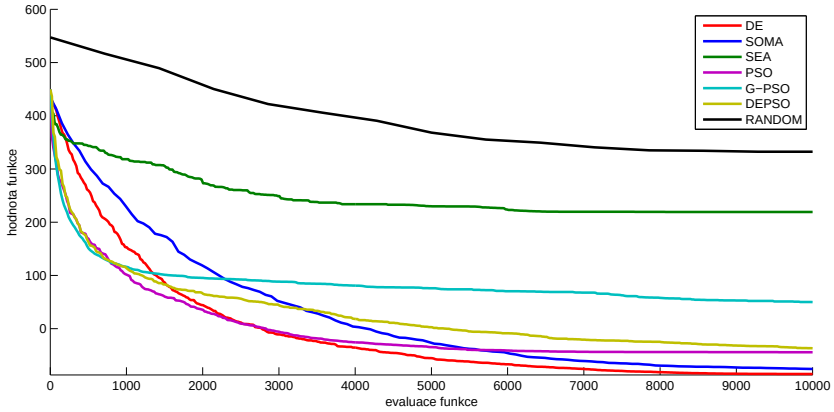

Průběh optimalizace – Rastrigin's Function




Průběh optimalizace – Schwefel's Function




Průběh optimalizace – Egg Holder Function




Průběh optimalizace – Ackley's Function

## 3.4 Náhodná změna

V roce 2002 byla představena metoda Dissipative Particle Swarm Optimization (DPSO) [20]. Jedná se o implementačně jednoduchou modifikaci metody PSO zavedením náhodných změn rychlosti i pozice jedinců. Každému jedinci je ještě před ohodnocením účelovou funkcí změněna rychlost:

**for** $i \leftarrow 1$ *to* $PopSize$ **do**
  **for** $d \leftarrow 1$ *to* $D$ **do**
    **if** $r_{(0,1)} < c_v$ **then**
      $v_{id} \leftarrow rand_{(-V_{max},V_{max})}$
    **end**
  **end**
**end**

a také pozice:

**for** $i \leftarrow 1$ *to* $PopSize$ **do**
  **for** $d \leftarrow 1$ *to* $D$ **do**
    **if** $r_{(0,1)} < c_p$ **then**
      $x_{id} \leftarrow rand_{(lb_d,ub_d)}$
    **end**
  **end**
**end**

Zde $rand_{(-V_{max},V_{max})}$ je náhodná rychlost a $rand_{(lb_d,ub_d)}$ je náhodná pozice v definičním oboru účelové funkce. Parametry $c_v$ a $c_p$ řídí pravděpodobnost náhodné změny. Vzhledem k tomu, že bývají nastaveny na malou hodnotu (například 0,002), ke změnám dochází výjimečně a většinou jen v jediné složce měněného jedince. Tento postup má dva důsledky – udržuje diverzitu populace a umožňuje uniknout z lokálního optima. Při optimalizaci některých funkcí však může přinést podstatné zvýšení výkonu, pouze za cenu velmi mírného snížení rychlosti konvergence populace. Při implementaci v metodách odvozených od PSO použiji zcela identický postup jako v DPSO, v metodách s jedinci bez rychlosti (DE, SOMA, SEA) se bude měnit jen pozice.

Vliv na průběh optimalizace se zjistil až po důkladných experimentech na testovacích funkcích. Výsledky porovnání původních metod a metod s náhodnou změnou jsou rozepsány v Tabulce 2 (původní metody jsou značeny svými zkratkami, metody s náhodnou změnou mají přidáno + NZ):

| Metoda | Rastrigin's F. | Schwefel's F. | Egg Holder F. | Ackley's F. |
|---|---|---|---|---|
| DE | -299,99994 | -12474,975 | -13674,671 | -85,398867 |
| DE + NZ | **-299,99995** | **-12529,407** | -13960,138 | **-86,043666** |
| SOMA | -298,86712 | -12084,987 | -15956,97 | -75,879108 |
| SOMA + NZ | -299,57859 | -12449,751 | **-19318,683** | -77,399041 |
| SEA | -80,975342 | -6093,246 | -7805,9872 | 219,29208 |
| SEA + NZ | -285,5693 | -11086,734 | -14315,09 | -32,729109 |
| PSO | -261,18227 | -9059,6876 | -11688,545 | -44,317431 |
| PSO + NZ | -282,53509 | -10579,922 | -13157,791 | -73,807882 |
| G-PSO | -256,6445 | -9164,0927 | -13608,588 | 49,781268 |
| G-PSO + NZ | -299,99991 | -11636,327 | -15261,986 | -83,657109 |
| DEPSO | -259,21233 | -9709,3126 | -12446,652 | -36,95492 |
| DEPSO + NZ | -287,76828 | -11615,582 | -15141,702 | -76,790469 |

Tabulka 2: Vliv náhodné změny

Je vidět, že náhodná změna je velice přínosné vylepšení, které posílilo každou testovanou metodu. Pravděpodobnost náhodné změny $c_p$ jsem zkoušel nastavit na 0,02, 0,004, 0,0008 a 0,00016. Nejlepší je nastavit ji velice nízkou, mě se nejlépe osvědčily hodnoty 0,0008 a 0,00016 (platí pro náhodnou změnu pozice ve všech metodách i rychlosti v PSO, G-PSO a DEPSO). Pokud je pravděpodobnost náhodné změny nastavena příliš vysoko, začne poškozovat původní algoritmus příliš častými změnami jedinců, což zpomaluje konvergenci ke globálnímu optimu. Samozřejmě existují výjimky, například metoda SOMA byla nejúspěšnější na Egg Holder Function při pravděpodobností náhodné změny 0,02 (zdůrazňuji, že prezentované výsledky jsou průměrem deseti běhů, zřejmě se tedy jednalo o zvláštní synergii mezi metodou SOMA a vysokou pravděpodobností náhodné změny, nikoliv o „štěstí“ při používání náhodně generovaných čísel). Celkově je toto zlepšení velice účinné – při správném nastavení parametrů se zvýšil výkon všech metod na všech účelových funkcích.

## 3.5 Sekvenční schéma

Klasické schéma přírodou inspirovaných optimaizačních metod používá koncept populace jedinců, která se mění po iteracích. V případě evolučních metod jsou vytvářeni noví jedinci paralelně a populace se obměňuje skokově – vždy až po ohodnocení všech nově vzniklých jedinců. PSO a metody z ní odvozené fungují podobně, díky zavedení rychlosti místo přímé

změny pozice jedinců. Tento přístup je výhodný zejména tím, že je jednoduše paralelizovatelný. Máme-li k dispozici $n$ výpočetních jednotek, budeme v každé iteraci vytvářet $k \cdot n$ nových jedinců a každá výpočetní jednotka bude ohodnocovat $k$ jedinců.

V určitých situacích však nelze paralení výpočty použít nebo je vhodnější program paralelizovat na jiné úrovni (na vyšší úrovni například zavedením více populací, na nižší například paralelizací částí účelové funkce). V tom případě může být efektivní zaměnit klasické paralelní schéma se skokovou obměnou populace na sekvenční schéma. V evolučních metodách DE, SOMA a SEA bude v sekvenčním schématu každý nový jedinec zařazen do populace ihned po svém ohodnocení na místo rodiče (pouze pokud je nový jedinec lepší než jeho rodič). Sekvenční schéma pro tyto metody vypadá následovně:

Inicializuj populaci $x$
**while** *ukončovací podmínka není splněna* **do**
  **foreach** $x_i \in x$ **do**
    $y_i \leftarrow$ Vytvoř nového jedince pomocí $x$
    Ohodnoť $y_i$
    **if** $f(y_i)$ *je lepší než* $f(x_i)$ **then**
      Nahraď jedince $x_i$ jedincem $y_i$
    **end**
  **end**
**end**
Vrať pozici nejlepšího jedince z $x$

Pro metody odvozené od PSO je sekvenční schéma podobné – ihned po vypočtení nové rychlosti každého jedince je tento jedinec přesunut na novou pozici (narozdíl od původního PSO, kde se jedinci pohybovali až po výpočtu všech rychlostí). Samotné vytváření jedinců probíhá přesně podle rovnic konkrétní metody.

Sekvenční přístup je rozdílný v tom, že se stav populace mění častěji, i když po menších krocích. Například pokud již první ohodnocovaný jedinec nalezne nové nejlepší řešení, ovlivní to okamžitě změnu ostatních jedinců (narozdíl od paralelního přístupu, při kterém by byli ovlivněni až v další iteraci). Nalezení globálního optima tak může být rychlejší. Vliv sekvenčnosti roste s počtem jedinců v populaci. Další výhodou je, že tento postup nepřidává narozdíl od jiných žádné nové parametry, pro které by bylo třeba hledat správná nastavení. Má však i podstatnou nevýhodu – optimalizace může být příliš agresivní a populace se může brzy zastavit v lokálním optimu. Pro porovnání obou přístupů posloužily experimenty na testovacích funkcích, jejichž výsledky jsou uvedeny v Tabulce 3 (sek-

venční verze jsou označeny SEQ):

| Metoda | Rastrigin's F. | Schwefel's F. | Egg Holder F. | Ackley's F. |
|---|---|---|---|---|
| DE | -299,99994 | -12474,975 | -13674,671 | -85,398867 |
| SEQ. DE | **-299,99996** | **-12495,117** | -13604,382 | **-85,641857** |
| SOMA | -298,86712 | -12084,987 | **-15956,97** | -75,879108 |
| SEQ. SOMA | -299,08222 | -11761,174 | -15954,982 | -76,129055 |
| SEA | -80,975342 | -6093,246 | -7805,9872 | 219,29208 |
| SEQ. SEA | -90,471095 | -5492,7677 | -8737,611 | 194,58268 |
| PSO | -261,18227 | -9059,6876 | -11688,545 | -44,317431 |
| SEQ. PSO | -265,43227 | -9277,7444 | -12625,249 | 1,6099522 |
| G-PSO | -256,6445 | -9164,0927 | -13608,588 | 49,781268 |
| SEQ. G-PSO | -247,9957 | -8760,1565 | -14373,189 | 92,05592 |
| DEPSO | -259,21233 | -9709,3126 | -12446,652 | -36,95492 |
| SEQ. DEPSO | -258,65284 | -9669,1999 | -13087,276 | 2,0079794 |

Tabulka 3: Srovnání paralelního a sekvenčního schématu

Z tabulky je patrné, že někdy je sekvenční úprava velmi účinná (například v metodách SEA a PSO na Rastrigin's Function a Egg Holder Function), naopak jindy zcela nevhodná (PSO, G-PSO a DEPSO na Ackley's Function). Proto ji narozdíl od náhodné změny nelze obecně doporučit, na druhou stranu je často možné vyzkoušet oba přístupy – na to je potřeba pouze dvojnásobného počtu ohodnocení účelové funkce.

## 3.6 Perturbace

V metodách DE a SOMA je zavedeno kombinování původních a nových jedinců (v DE se jedná o křížení s parametrem CR, v SOMA o perturbaci s parametrem PRT). Kombinování probíhá po dimenzích – parametry CR a PRT lze nastavit pravděpodobnost, s jakou dojde ke změně v každé dimenzi původního jedince. Je prokázáno, že takové kombinování je žádoucí a může značně urychlit nalezení globálního extrému [3, 4, 6].

V ostatních představených metodách však žádné přímé kombinování původních a nových jedinců zavedeno není. Proto jsem tento postup implementoval i do ostatních metod a pokusil jsem se zjistit, jestli přinese nějaké zvýšení výkonu. V metodě SEA se nový i-tý jedinec mění následovně:

**for** $d \leftarrow 1$ *to* $D$ **do**
  **if** $perturbation > rand_{0,1}$ **then**
    $x_{id} \leftarrow x_{id} + 2 \cdot (c - x_w)$
  **else**
    $x_{id} \leftarrow x_{pd}$
  **end**
**end**

$c$ je střed spočítaný z rodičů i-tého jedince (dle původního postupu), *perturbation* je pravděpodobnost změny i-tého jedince v každé dimenzi a $x_p$ je jedinec, se kterým je prováděno kombinování. Tohoto jedince je možné vybrat více způsoby – vyzkouším strategie Best (nejlepší jedinec populace), Rand (náhodně vybraný jedinec), BestParent (nejlepší rodič), RandParent (náhodně vybraný rodič), WorstParent (nejhorší rodič) a Center (střed $c$). Rovněž v metodách odvozených od PSO jsem pozice jedince aktualizoval odlišně od původního návrhu:

**for** $d \leftarrow 1$ *to* $D$ **do**
  **if** $perturbation > rand_{0,1}$ **then**
    $x_{id} \leftarrow x_{id} + v_{id}$
  **end**
**end**

Tento postup zajistí, že se jedinec bude pohybovat v každé dimenzi s pravděpodobností *perturbation*. Výsledky jsou v Tabulce 4, metody s perturbací jsou označeny + PRT:

| Metoda | Rastrigin's F. | Schwefel's F. | Egg Holder F. | Ackley's F. |
|---|---|---|---|---|
| DE | **-299,99994** | **-12474,975** | -13674,671 | **-85,398867** |
| DE + PRT | **-299,99994** | **-12474,975** | -13674,671 | **-85,398867** |
| SOMA | -298,86712 | -12084,987 | -15956,97 | -75,879108 |
| SOMA + PRT | -298,86712 | -12084,987 | -15956,97 | -75,879108 |
| SEA | -80,975342 | -6093,246 | -7805,9872 | 219,29208 |
| SEA + PRT | -277,47053 | -11623,899 | **-17585,201** | -33,233253 |
| PSO | -261,18227 | -9059,6876 | -11688,545 | -44,317431 |
| PSO + PRT | -197,67443 | -7286,9756 | -7391,8754 | 9,7463337 |
| G-PSO | -256,6445 | -9164,0927 | -13608,588 | 49,781268 |
| G-PSO + PRT | -249,83325 | -10208,887 | -13869,558 | -15,905009 |
| DEPSO | -259,21233 | -9709,3126 | -12446,652 | -36,95492 |
| DEPSO + PRT | -276,42247 | -9978,8588 | -13505,833 | -65,948661 |

Tabulka 4: Vliv perturbace

Při testování tohoto zlepšení jsem zkoušel nastavení pravděpodobnosti *perturbation* na 0,1, 0,3, 0,5 a 0,7. Jako nejvhodnější hodnota se ve většině metod ukázala 0,1 nebo 0,3. Perturbace prospěla zejména metodě SEA, u které bylo zvýšení výkonu na všech účelových funkcích opravdu vysoké, na Egg Holder Function dokonce dosáhla nejlepšího výsledku ze všech testovaných metod. Ještě připomenu, že v metodách DE a SOMA je perturbace (v DE pod názvem křížení) zahrnuta již v původním návrhu, proto jsou v tabulce u těchto metod stejné výsledky.

## 3.7 Zavedení predátora

Predace znamená v ekologii aktivní vyhledávání kořisti pro potravu. Predátor se snaží ulovit kořist a kořist se snaží uniknout predátorovi. To vede k neustálému evolučnímu vývoji a zdokonalování predátora i kořisti, což výstižně popisuje Hypotéza červené královny [23]:

**Hypotéza.** Neustálý vývoj evolučního systému je nezbytný pro udržení stejné relativní biologické zdatnosti vzhledem k ostatním systémům.

Tuto hypotézu lze využít i v přírodou inspirovaných optimalizačních metodách – běžní jedinci se chovají jako kořist, a zavede se nová populace obsahující predátory. Predátoři mění pozici v prostoru podle odlišných rovnic než kořist a nejsou ohodnocování účelovou funkcí. Rovnice běžných jedinců se změní, protože musí reflektovat přítomnost predátorů. Podobně jako v přírodě se predátoři budou pohybovat směrem ke kořisti a běžní jedinci se budou pohybovat směrem od predátorů. Tímto způsobem predátoři zabrání předčasné konvergenci optimalizační metody – pokud se běžní jedinci zastaví v lokálním optimu, predátoři je vytlačí a umožní jim najít globální optimum.

Tento princip byl implementován do metody PSO [21, 22]. Autoři použili jediného predátora, který se pohybuje prostorem dle:

$$x_p \leftarrow x_p + rand_{0,1} \cdot speed \cdot (x_g - x_p)\,,$$

kde *speed* udává rychlost predátora, $x_g$ je aktuální pozice nejlepšího běžného jedince. Postup na změnu rychlosti běžného i-tého jedince v d-té dimenzi byl upraven následovně:

**if** $fear > rand_{0,1}$ **then**
$dist \leftarrow x_{id} - x_{pd}$
$v_{id} \leftarrow \omega \cdot \ v_{id} + c_1 \cdot \ r_1 \cdot (b_{gd} - x_{id}) + c_2 \cdot r_2 \cdot (b_{id} - x_{id}) +$
$sgn(dist) \cdot \ r_3 \cdot \alpha \cdot \ e^{-\beta \cdot dist}$
**else**
$v_{id} \leftarrow \omega \cdot v_{id} + c_1 \cdot r_1 \cdot (b_{gd} - x_{id}) + c_2 \cdot r_2 \cdot (b_{id} - x_{id})$
**end**

*dist* je Euklidovská vzdálenost i-tého jedince a predátora v d-té dimenzi. $sgn(dist)$ nastavuje pohyb jedince směrem od predátora. Parametr $fear$ určuje, jak často bude jedinec ovlivněn predátorem, parametry $\alpha$ a $\beta$ umožňují řídit sílu vlivu predátora na jedince. Čím větší je parametr $\alpha$, tím dále se jedinec pohybuje směrem od predátora. Zvýšením parametru $\beta$ se sníží vliv predátora na vyšší vzdálenost.

Výše popsanou rovnici lze použít pouze v metodách PSO a G-PSO, nikoliv však v metodách, které nepoužívají rychlost jedinců (DE, SOMA, SEA a částečně DEPSO). Abych zajistil použitelnost zlepšení ve větším množství optimalizačních metod, upravil jsem výše popsanou rovnici přímo na změnu pozice jedinců. Po vytvoření nového i-tého jedince (a před jeho ohodnocením) se změní jeho pozice v závislosti na pozici predátora podle:

**for** $d \leftarrow 1$ *to* $D$ **do**
**if** $fear > rand_{0,1}$ **then**
$dist \leftarrow x_{id} - x_{pd}$
$x_{id} = x_{id} + \alpha \cdot e^{-\beta \cdot dist}$
**end**
**end**

Veškeré parametry mají stejný význam jako v původním návrhu. Možné hodnoty jsou $speed \in \langle 0, 2 \rangle$, $fear \in \langle 0, 1 \rangle$, $\alpha \in \langle 0, 3 \rangle$ a $beta \in \langle 0, 3 \rangle$. Při provádění experimentů jsem se potýkal s obtížemi při hledání správné kombinace parametrů. Výsledky jsou uvedeny v Tabulce 5, verze s predátorem jsou označeny + P:

| Metoda | Rastrigin's F. | Schwefel's F. | Egg Holder F. | Ackley's F. |
|---|---|---|---|---|
| DE | **-299,99994** | -12474,975 | -13674,671 | **-85,398867** |
| DE + P | -261,93218 | **-12541,18** | -14065,659 | -65,228556 |
| SOMA | -298,86712 | -12084,987 | -15956,97 | -75,879108 |
| SOMA + P | -275,7412 | -12342,973 | **-17085,217** | -62,677149 |
| SEA | -80,975342 | -6093,246 | -7805,9872 | 219,29208 |
| SEA + P | -173,35396 | -8694,9743 | -12219,573 | 63,358491 |
| PSO | -261,18227 | -9059,6876 | -11688,545 | -44,317431 |
| PSO + P | -159,92699 | -9664,6287 | -12255,39 | -17,174335 |
| G-PSO | -256,6445 | -9164,0927 | -13608,588 | 49,781268 |
| G-PSO | -137,32437 | -10730,847 | -15429,665 | 32,621338 |
| DEPSO | -259,21233 | -9709,3126 | -12446,652 | -36,95492 |
| DEPSO + P | -162,13347 | -9789,4232 | -13535,045 | 15,180958 |

Tabulka 5: Vliv zavedení predátora

Z výsledků je možné vypozorovat, že přítomnost predátora měla kladný efekt na metodu SEA, spíše sporný na metody DE a SOMA (zlepšení na Schwefel's function a Egg Holder function za cenu snížení výkonu na Rastrigin's function a Ackley's function) a většinou záporný na metody PSO, G-PSO a DEPSO. SEA je navíc poměrně slabá metoda na optimalizaci vybraných účelových funkcí, takže ani po zlepšení zavedením predátora nedosahuje tak dobrých výsledků jako ostatní metody. Jedním z důvodů neúspěchu tohoto vylepšení může být příliš vysoký počet nových parametrů, které jsem nedokázal zcela správně nastavit. Různým metodám totiž vyhovovala různá nastavení – například dobré nastavení pro metodu SOMA na Egg Holder function je $speed = 1$, $fear = 0,1$, $\alpha = 2,0$ a $\beta$ libovolně mezi 0,1 a 1. Naopak pro DE na stejné účelové funkci je nejlepší nastavit $speed = 0,5$, $fear = 0,5$, $\alpha$ libovolně mezi 0,1 a 1 a $\beta = 0.5$. Vhodná nastavení se také liší na různých účelových funkcích. Po vyzkoušení mnoha různých kombinací parametrů jsem zjistil, že nejdůležitější je správně nastavit $speed$ a $fear$. Na přesném nastavení $\alpha$ a $\beta$ příliš nezáleží.

## 3.8 Celkové výsledky

Celkové výsledky optimalizace vybraných funkcí jsou shrnuty v Tabulce 6:

| Metoda | Rastrigin's F. | Schwefel's F. | Egg Holder F. | Ackley's F. |
|---|---|---|---|---|
| RANDOM | 24,89933 | -4239,3404 | -6189,8079 | 332,39947 |
| DE | -299,99994 | -12474,975 | -13674,671 | -85,398867 |
| DE + NZ | -299,99995 | -12529,407 | -13960,138 | **-86,043666** |
| SEQ. DE | **-299,99996** | -12495,117 | -13604,382 | -85,641857 |
| DE + PRT | -299,99994 | -12474,975 | -13674,671 | -85,398867 |
| DE + P | -261,93218 | **-12541,18** | -14065,659 | -65,228556 |
| SOMA | -298,86712 | -12084,987 | -15956,97 | -75,879108 |
| SOMA + NZ | -299,57859 | -12449,751 | **-19318,683** | -77,399041 |
| SEQ. SOMA | -299,08222 | -11761,174 | -15954,982 | -76,129055 |
| SOMA + PRT | -298,86712 | -12084,987 | -15956,97 | -75,879108 |
| SOMA + P | -275,7412 | -12342,973 | -17085,217 | -62,677149 |
| SEA | -80,975342 | -6093,246 | -7805,9872 | 219,29208 |
| SEA + NZ | -285,5693 | -11086,734 | -14315,09 | -32,729109 |
| SEQ. SEA | -90,471095 | -5492,7677 | -8737,611 | 194,58268 |
| SEA + PRT | -277,47053 | -11623,899 | -17585,201 | -33,233253 |
| SEA + P | -173,35396 | -8694,9743 | -12219,573 | 63,358491 |
| PSO | -261,18227 | -9059,6876 | -11688,545 | -44,317431 |
| PSO + NZ | -282,53509 | -10579,922 | -13157,791 | -73,807882 |
| SEQ. PSO | -265,43227 | -9277,7444 | -12625,249 | 1,6099522 |
| PSO + PRT | -197,67443 | -7286,9756 | -7391,8754 | 9,7463337 |
| PSO + P | -159,92699 | -9664,6287 | -12255,39 | -17,174335 |
| G-PSO | -256,6445 | -9164,0927 | -13608,588 | 49,781268 |
| G-PSO + NZ | -299,99991 | -11636,327 | -15261,986 | -83,657109 |
| SEQ. G-PSO | -247,9957 | -8760,1565 | -14373,189 | 92,05592 |
| G-PSO + PRT | -249,83325 | -10208,887 | -13869,558 | -15,905009 |
| G-PSO | -137,32437 | -10730,847 | -15429,665 | 32,621338 |
| DEPSO | -259,21233 | -9709,3126 | -12446,652 | -36,95492 |
| DEPSO + NZ | -287,76828 | -11615,582 | -15141,702 | -76,790469 |
| SEQ. DEPSO | -258,65284 | -9669,1999 | -13087,276 | 2,0079794 |
| DEPSO + PRT | -276,42247 | -9978,8588 | -13505,833 | -65,948661 |
| DEPSO + P | -162,13347 | -9789,4232 | -13535,045 | 15,180958 |

Tabulka 6: Srovnání globálních optimalizačních metod a zlepšení

Po vyzkoušení všech vylepšení je stále nejlepší optimalizační metodou DE, která je nejvýkonnější na třech účelových funkcích. Pouze na Egg Holder Function je nejschopnější metoda SOMA. To neznamená, že jsou ostatní metody obecně horší, protože na jiných účelových funkcích by mohly být výsledky zcela odlišné.

# Kapitola 4

# Závěr

V této práci jsem popsal některé přírodou inspirované globální optimalizační metody a porovnal jejich výkon. Na testovacích funkcích Rastrigin's Function, Schwefel's Function a Ackley's Function byla nejúspěšnější metoda Differential Evolution, Egg Holder Function byla nejlépe optimalizována metodou Self-Organizing Migrating Algorithm.

Dále jsem představil čtyři možnosti zlepšení přírodou inspirovaných optimalizačních metod – náhodnou změnu, perturbaci, sekvenční schéma a zavedení predátora. Ukázalo se, že navrhovaná zlepšení mohou být v mnoha případech přínosná a neměla by být opomíjena při použití přírodou inspirovaných metod v praxi. Problematické však může být nalezení vhodného nastavení parametrů pro konkrétní úlohu. Potvrdilo se, že neexistuje žádná univerzální konfigurace optimalizační metody nebo jejího zlepšení, která by byla superiorní na všech úlohách.

Celkově nejlepším zlepšením se ukázala být náhodná změna. Zavedení náhodné změny jedinců v průběhu optimalizačního procesu umožnilo více či méně zlepšit každou ze studovaných metod. Jako nejslabší vylepšení bych označil použití predátora, na druhou stranu i tento postup je v určitých případech přínosný, ale je třeba pečlivě nastavit jeho parametry pro konkrétní účelovou funkci.

Domnívám se, že vývoj nových optimalizačních metod a zlepšování již existujících je důležité jak pro jejich lepší praktické využití, tak pro porozumění celému oboru globální optimalizace. S rostoucím výkonem počítačů a pokrokem na poli optimalizace je možné řešit stále obtížnější problémy, což je přínosné pro celou naší společnost. Proto lze předpokládat, že i v budoucnosti bude optimalizaci věnována vysoká pozornost v akademické i průmyslové sféře.

# Literatura


[1] Weise, T.: *Global Optimization Algorithms – Theory and Application* [online]. Thomas Weise, 2008. Dostupné na WWW: <http://www.it-weise.de/>.

[2] Storn R., Price. K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* **11**, 1997, 341–359.

[3] Price K., Storn R., Lampinen J. A.: *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin Heidelberg, 2005.

[4] Feoktistov V.: *Differential Evolution: In Search of Solutions*. Springer Science+Business Media, New York, 2006.

[5] Fan H., Lampinen J.: A Trigonometric Mutation Operation in Differential Evolution. *Journal of Global Optimization* **27**, 2003, 105–129.

[6] Zelinka I.: *Umělá inteligence v problémech globální optimalizace*. BEN – technická literatura, Praha, 2002.

[7] Babjak J., Palko M.: Strategies for Improving Performance of SOMA. In *Proceedings of the 3rd Czech-Slovak Seminar Cognition, Artificial Life and Computational Intelligence*, 2003, 279–284.

[8] Chen, Z., Lishan, K.: Steady-State Evolutionary Algorithm for Multimodal Function Global Optimization. In *CIS 2005, Part I, LNAI 3801*, 2005, 200–207.

[9] Kennedy J., Eberhart R.: Particle Swarm Optimization. In *Proc. of the IEEE Int. Conf. on Neural Networks*, 1995, 1942–1948.

[10] Shi Y., Eberhart R.: Parameter Selection in Particle Swarm Optimization. In *EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII*, 1998, 591–600.

[11] Zhang L., Yu H., Hu S.: A New Approach to Improve Particle Swarm Optimization. In *Genetic and Evolutionary Computation — GECCO 2003*, 2003, 134–139.

[12] Schutte J., Groenwold A.: A Study of Global Optimization Using Particle Swarm. *Journal of Global Optimization* **31**, 2005, 93–108.

[13] Zhang L., Yu H., Hu S.: Optimal choice of parameters for particle swarm optimization. *Journal of Zheijang University SCIENCE* **6**, 2005, 528–534.

[14] Pasupuleti, S., Battiti, R.: The Gregarious Particle Swarm Optimizer (G-PSO). In *GECCO '06: Proceeeding of the 8th annual conference on Genetic and evolutionary computation*, 2006, 64–74.

[15] Zhang, W.-J., Xie, X.-F.: DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. In *IEEE International Conference Systems, Man and Cybernetics*, 2003, vol. 4, 3816–3821.

[16] Mishra S.: *Repulsive Particle Swarm Method on Some Difficult Test Problems of Global Optimization* [online]. MPRA, 2006. Dostupné na WWW: <http://mpra.ub.uni-muenchen.de/1742/>.

[17] Mishra S.: *Some new test functions for global optimization and performance of repulsive particle swarm method* [online]. MPRA, 2006. Dostupné na WWW: <http://mpra.ub.uni-muenchen.de/2718/>.

[18] Whitley, L. D., Mathias K. E., Rana S. B., Dzubera J.: Building Better Test Functions. In *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, 239–247.

[19] Zelinka, I.: *Test Functions* [online]. SOMA Homepage, 2005. Dostupné na WWW: <http://www.ft.utb.cz/people/zelinka/soma/>.

[20] Xie X.-F., Zhang W.-J., Yang Z.-L.: A Dissipative Particle Swarm Optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002, vol. 2, 1456–1461.

[21] Silva, A., Neves, A. Costa E.: Chasing the Swarm: A Predator Prey Approach to Function Optimisation. In *Proceedings of the MENDEL2002 – 8th International Conference on Soft Computing*, 2002.

[22] Silva, A., Neves, A., Costa, E.: An Empirical Comparison of Particle Swarm and Predator Prey Optimisation. In *Artificial Intelligence and Cognitive Science*, 2002, 1–45.

[23] Heylighen, F.: *The Red Quen Principle* [online]. Principia Cybernetica Web, 1993. Dostupné na WWW: <http://pespmc1.vub.ac.be/REDQUEEN.html>.

# Programová příloha

Součástí práce je implementace přírodou inspirovaných globálních optimalizačních metod a jejich zlepšení a běžných účelových funkcí v jazyce C++ pro platformu Microsoft Windows. Příloha se skládá z programu NatOpti, který slouží k dávkovému testování optimalizačních metod, kompletních zdrojových kódů, uživatelské i programátorské dokumentace a veškerých experimentálních výsledků, kterých bylo s programem dosaženo.