

ABSTRAKT

Tato práce se věnuje problematice moderních evolučních algoritmů. Nastiňuje jejich základní rozdělení a stručně představuje princip fungování některých z nich. Dále se zaměřuje na podrobnější popis algoritmu Particle Swarm (rojení částic), založeného na souvislostech sociálního chování zvířat. Praktická část práce představuje řešení několika optimalizačních problémů pomocí tohoto algoritmu.

This study deals with issue of modern evolutionary algorithms, presents their possible division and briefly introduces their basics. Next part of the study is focused on Particle Swarm Optimization (PSO), based on principles of animal social behaviour. Practical part contains several PSO solved optimization problems.

Poděkování:

Rád bych na tomto místě poděkoval panu doc. Ing. Ivanu Zelinkovi, PhD. za podněty a shovívavost při vedení mé bakalářské práce.

Prohlašuji, že na celé diplomové práci jsem pracoval samostatně a použitou literaturu jsem citoval.

Ve Zlíně červen 2005

.....

Martin Ježek

ÚVOD.....	6
I. TEORETICKÁ ČÁST	7
1 EVOLUČNÍ ALGORITMY	8
1.1 ROZDĚLENÍ OPTIMALIZAČNÍCH ALGORITMŮ	9
1.1.1 Enumerativní algoritmy.....	10
1.1.2 Deterministické algoritmy	10
1.1.3 Stochastické algoritmy	11
1.1.4 Smíšené algoritmy	11
1.2 NÁSTIN PRINCIPU VYBRANÝCH ALGORITMŮ	11
1.2.1 Stochastic Hill – Climbing (Stochastický horolezecký algoritmus).....	11
1.2.2 Tabu Search (Zakázané prohledávání)	12
1.2.3 Ant Colony Optimization (ACO optimalizace mravenčí kolonií)	13
1.2.4 Scatter Search (Roptýlové prohledávání).....	14
1.2.5 Immunology system method (Metoda imunologického systému).....	14
1.2.6 Differential evolution (Diferenciální evoluce)	14
1.2.7 SOMA (Samo-organizující se migrační algoritmus).....	15
1.3 ALGORITMUS PARTICLE SWARM OPTIMISATION (PSO)	16
1.3.1 Historické pozadí vzniku PSO algoritmu.....	16
1.3.2 Základní princip PSO	17
1.3.3 Popis celého algoritmu PSO.....	19
1.3.4 Varianty PSO.....	19
II. PRAKTICKÁ ČÁST	22
2 EXPERIMENTÁLNĚ-SIMULAČNÍ ČÁST.....	23
2.1 TESTOVÁNÍ FUNKČNOSTI ALGORITMU PSO NA NĚKOLIKA VYBRANÝCH TESTOVACÍCH FUNKCÍCH	23
2.1.1 Funkce f6.....	23
2.1.2 Funkce Sphere (první De Jongova funkce)	25
2.1.3 Rosenbrockova funkce (Rosenbrockovo sedlo, druhá De Jongova funkce).....	26
2.1.4 Rastriginova funkce	27
2.1.5 Griewankova funkce.....	29
2.1.6 Ackleyova funkce	30
2.1.7 Schwefelova funkce	31
2.2 TESTOVÁNÍ FUNKČNOSTI ALGORITMU PSO NA JIŽ ŘEŠENÝCH PROBLÉMECH.....	33
2.2.1 Příklad 1: Návrh ozubeného kola	33
2.2.2 Příklad 2: Návrh tlakové nádoby	35
2.3 ZHODNOCENÍ EXPERIMENTÁLNĚ-SIMULAČNÍ ČÁSTI	37
SEZNAM POUŽITÉ LITERATURY.....	39
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	41
SEZNAM OBRÁZKŮ	42
SEZNAM TABULEK.....	43

ÚVOD

V 60. letech 20. století vzniká nová skupina optimalizačních algoritmů, tzv. evoluční algoritmy. Jejich jméno bylo odvozeno z evolučních procesů, probíhajících v přírodě, jež v 19. století popsal Charles Darwin.

Během posledního čtvrtstoletí byly tyto algoritmy dovedeny téměř k dokonalosti a není proto divu, že v současnosti úspěšně nahrazují algoritmy založené na dnes již klasických metodách, využívajících matematický aparát.

Cílem této práce je seznámit čtenáře se základní problematikou evolučních metod.

Ve své první části proto přináší obecné seznámení s evolučními algoritmy a jejich výhodami při použití v inženýrské praxi. Dále nabízí rozdělení optimalizačních metod obecně a začlenění evolučních algoritmů v tomto dělení.

V další části poskytuje přehled některých moderních evolučních algoritmů a popisuje jejich základní principy.

Následně se zaměřuje na algoritmus Particle Swarm (rojení částic). Podává informace o historických základech a myšlenkových tezích, na nichž byl algoritmus vybudován. Přibližuje podstatu tohoto algoritmu, popisuje detailně jeho strukturu a nakonec uvádí jeho různé modifikace.

V praktické části potom demonstruje funkčnost algoritmu při optimalizaci několika funkcí, používaných pro testování optimalizačních algoritmů, a při řešení vybraných optimalizačních problémů z inženýrské praxe.

Součástí práce je i sestavený algoritmus v programovém prostředí *Mathematica*, který je dostupný na přiloženém médiu.

I. TEORETICKÁ ČÁST

1 EVOLUČNÍ ALGORITMY

Většina problémů inženýrské praxe může být definována jako optimalizační problém, jako například nalezení optimální trajektorie robota, optimální tloušťky stěny tlakové nádoby, optimální nastavení parametrů regulátoru atd. Jinými slovy, řešený problém lze převést na matematický problém daným vhodným funkčním předpisem, jehož optimalizace vede k nalezení argumentů tzv. účelové funkce, což je cílem optimalizace.

V 60. letech 20. století vzniká nová skupina optimalizačních algoritmů, tzv. evoluční algoritmy. Jejich jméno je odvozeno z evolučních procesů, probíhajících v přírodě. Princip evolučních procesů odhalil v 19. století Charles Darwin. Spočívá v tom, že se cyklicky tvoří generace nových jedinců a z nich přežívají jen ti, pro dané prostředí nejvhodnější. Tyto principy byly zautomatizovány a používají se dnes na řešení mnohých, jinak obtížně řešitelných, problémů.

Řešení takových problémů obvykle vyžaduje práci s argumenty optimalizovaných funkcí, přičemž definiční obor těchto argumentů může být různorodého charakteru, jako například obor celočíselný, reálný, komplexní, diskrétní apod. Navíc se může stát, že pro určité subintervaly z povoleného intervalu hodnot může příslušný argument optimalizované funkce nabývat různých typů hodnot. Navíc v rámci optimalizace mohou být uplatněny různé penalizace a omezení nejen na dané argumenty, ale také na funkční hodnotu optimalizované funkce.

Pro úspěšné řešení takových problémů byla v posledních třech desetiletích vyvinuta množina výkonných algoritmů, které umožňují řešit velmi složité problémy efektivním způsobem. Mezi jejich výhody patří především to, že řešitel daného problému nemusí znát optimalizační metodu jako takovou, ale je po něm požadována pouze velmi dobrá znalost dané optimalizované problematiky a schopnost ji dobře popsat. Z dalších výhod můžeme uvést, že na rozdíl od běžných optimalizačních metod se tyto algoritmy soustředí na hledání globálních a ne lokálních extrémů. Oproti klasickým numerickým metodám, poskytují tyto algoritmy při ukončení zpravidla více než jedno řešení. Evoluční algoritmy mají i své nevýhody, které zpravidla vyplývají z faktu, že součástí algoritmů je práce s pseudonáhodnou veličinou. Díky tomu je těžké předpovídat jejich chování a rovněž je složité sestavovat jejich matematické důkazy. Přesto praxe jednoznačně potvrzuje, že evoluční algoritmy často nalézají lepší řešení než klasické algoritmy a jsou celkově použitelnější a výhodnější.

1.1 Rozdělení optimalizačních algoritmů

Existují různá rozdělení optimalizačních algoritmů. Pro naši práci jsme využili rozdělení dle *Zelinky (2002)*. Toto rozdělení se odvíjí od principu činnosti jednotlivých algoritmů. Zahrnuje jak klasické, tak i moderní optimalizační metody.

Tab. 1 Uspořádání optimalizačních algoritmů dle *Zelinky (2002)*

Globální prohledávání a optimalizace	
Enumerativní	
Deterministické	Greedy
	Hill - Climbing
	Branch & Bound
	Depth - First
	Breadth - First
	Best - First
	Calculus based
Stochastické	Random Search -Walk
	Simulated Annealing
	Monte Carlo
	Tabu Search
	Evolutionary Computation
	Stochastic Hill - Climbing

Smíšené	Mathematical Programming
	Ant Colony Optimization
	Immune System Methods
	Memetic Algorithms
	Scatter Search & Path Relinking
	Particle Swarm
	Genetic Algorithms
	Differential Algorithms
	SOMA

1.1.1 Enumerativní algoritmy

Tyto algoritmy vypočítávají hodnotu účelové funkce pro všechny možné kombinace daného problému. Jsou tedy vhodné pouze pro problémy, kde argumenty účelové funkce jsou diskrétní veličiny a nabývají malého množství hodnot. Pro jiné problémy jsou tyto algoritmy, z hlediska časové náročnosti, nepoužitelné.

1.1.2 Deterministické algoritmy

Skupina těchto algoritmů využívá pouze metody klasické matematiky. K dosažení efektivních výsledků vyžadují deterministické algoritmy tyto předpoklady:

- problém je lineární
- problém je konvexní
- prohledávaný prostor možných řešení je malý
- prohledávaný prostor možných řešení je spojitý
- účelová funkce má, pokud možno, jen jeden extrém (je unimodální)
- mezi parametry účelové funkce nejsou nelineární interakce

- jsou dostupné informace typu gradient
- problém je definován v analytickém tvaru

1.1.3 Stochastické algoritmy

Algoritmy stochastické jsou založeny na využití náhody. V podstatě jde o čistě náhodný výběr hodnot argumentů účelové funkce. Výsledkem je to nejlepší řešení, jenž bylo během činnosti algoritmu nalezeno. Tyto algoritmy jsou obvykle:

- pomalé
- vhodné pouze pro malý rozsah argumentů účelové funkce
- vhodné pro hrubý odhad

1.1.4 Smíšené algoritmy

Kombinaci metod předcházejících dvou skupin využívají algoritmy smíšené. Na základě vyváženého spolupůsobení řádu a chaosu, dosahují tyto algoritmy překvapivě dobrých výsledků. Do této skupiny patří i algoritmy evoluční. Hlavní rysy smíšených algoritmů:

- jsou robustní - nezávisle na počátečních podmínkách nalézají kvalitní řešení (jeden či více globálních extrémů)
- jsou efektivní a výkonné - nalézají kvalitní řešení v krátkém čase
- mají minimální požadavky na předběžné informace
- nepotřebují ke své činnosti analytický popis problému
- jsou schopny nalézt více řešení během jednoho spuštění

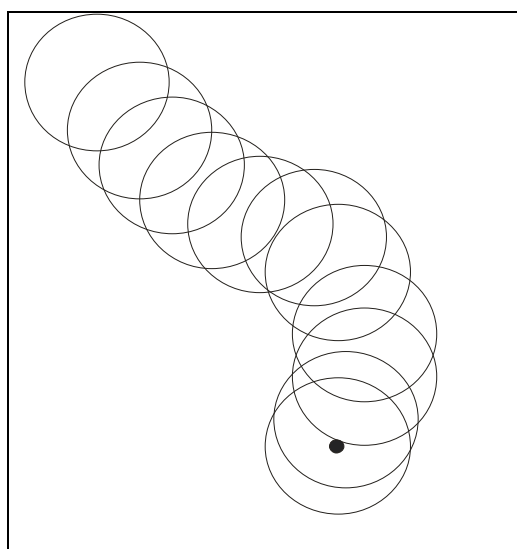
1.2 Nástin principu vybraných algoritmů

1.2.1 Stochastic Hill – Climbing (Stochastický horolezecký algoritmus)

Stochastic Hill – Climbing je verzí tzv. horolezeckého algoritmu, obohaceného o stochastickou složku. Patří mezi gradientní metody, což udává, že prohledává prostor možných řešení ve směru největšího spádu. Díky tomu ovšem velmi často uvázne v lokálním extrému.

Základní verze funguje tak, že se vždy vychází z náhodného bodu v prostoru možných řešení. Účelová funkce se minimalizuje jen v určitém okolí tohoto bodu, daném konečným souborem transformací. Získané lokální řešení se pak použije jako střed pro výpočet nového okolí. Tento proces projde předepsaným počtem opakování. V průběhu celé historie algoritmu se zaznamenává nejlepší řešení, které slouží jako výsledné optimum.

Stochastický horolezecký algoritmus je v podstatě klasický horolezecký algoritmus, spuštěný několikrát po sobě s náhodně vygenerovanými počátečními stavy.



Obr. 1 Princip horolezeckého algoritmu

1.2.2 Tabu Search (Zakázané prohledávání)

Tabu Search je vylepšenou verzí horolezeckého algoritmu. Navrhl ji koncem 80. let profesor Fred Glover z University of Colorado. Do horolezeckého algoritmu zavedl tak zvanou krátkodobou paměť. Ta po určitý krátký interval předcházející historie algoritmu uchovává inverzní transformace k lokálně optimálním transformacím, použitým k získání nových středů. Tyto inverzní transformace jsou zakázány při tvorbě nového okolí. Tímto jednoduchým způsobem je možné podstatně omezit výskyt zacyklení, při pádu do lokálního optima.

Numerické zkušenosti s metodou zakázaného prohledávání ukazují, že velikost této paměti (délka zakázaného seznamu) je velmi důležitým parametrem, ovlivňujícím schopnost vymanit se z lokálního extrému. Příliš malá délka nemusí zabránit zacyklení a příliš velká může způsobit přeskočení nadějných lokálních minim.

V současnosti tato metoda patří mezi hlavní hity v oblasti operačního výzkumu a algoritmů na řešení kombinatorických úloh.

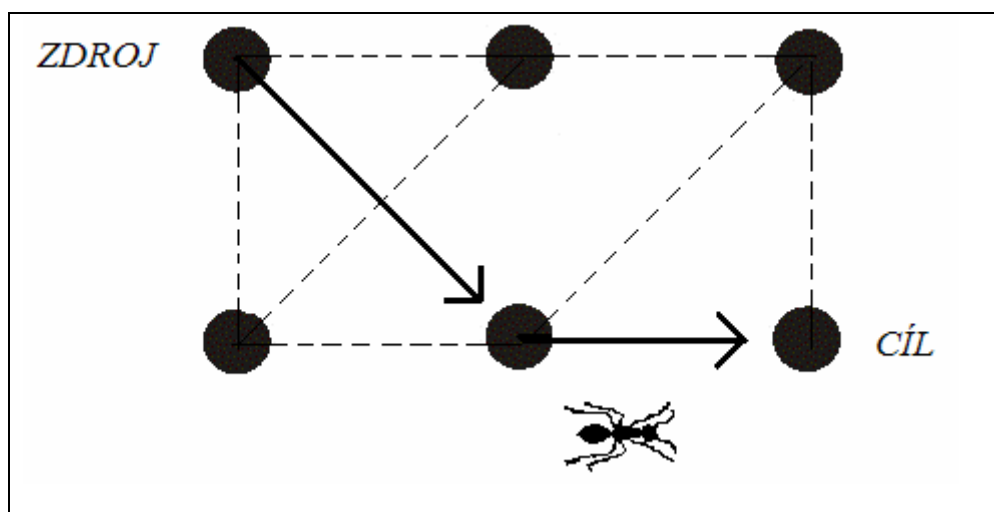
1.2.3 Ant Colony Optimization (ACO optimalizace mravenčí kolonií)

Činnost tohoto algoritmu je založena na chování mravenců v kolonii. Využívá tato základní pravidla pohybu mravenců:

- izolovaný mravenec se při cestě za potravou pohybuje zcela náhodně
- každý pohybující se mravenec zanechává za sebou na zemi feromonové stopy a tím označuje svoji trajektorii
- mravenec, který narazí na feromonové stopy si vybírá cestu s větší koncentrací feromonů

Po nejkratší cestě mezi zdrojem mravenců (mraveniště) a jejich cílem snažení (potrava) se za stejnou dobu přesune nejvíce mravenců. Zanechají zde nejvíce feromonových stop, na jejichž základě další mravenci tuto cestu preferují. Feromon je v algoritmu zastoupen vahou, která je přiřazena dané cestě, vedoucí k cíli. Tato váha je, stejně jako v případě mravenčích feromonů, aditivní. To znamená, že narůstá s počtem „mravenců“, kteří touto cestou projdou. V ACO algoritmu je zohledněno i „vypařování feromonů“ (váhy u jednotlivých spojů časem slábnou), což zvyšuje robustnost algoritmu.

ACO byl použit s úspěchem na optimalizační problémy typu „problém obchodního cestujícího“ či „návrh telekomunikačních sítí“.



Obr. 2 Algoritmus Ant Colony Optimization

1.2.4 Scatter Search (Roptylové prohledávání)

Algoritmus Scatter Search je svoji podstatou podobný algoritmu Tabu Search. Je to vektorově orientovaný algoritmus, který generuje nové vektory (řešení) na základě pomocných heuristických technik. Při startu se vychází z řešení, získaných pomocí vhodné heuristické techniky. Poté jsou generována nová řešení na základě podmnožiny nejlepších řešení ze startu. Z nově nalezených řešení se opět vybere množina těch nejlepších a celý proces se opakuje.

Tento algoritmus byl použit k řešení problémů, jako je řízení dopravy, učení neuronové sítě či optimalizaci bez omezení.

1.2.5 Immunology system method (Metoda imunologického systému)

Tento algoritmus je založen na principech fungování imunologického systému v živých organismech. Na imunologický systém je nahlíženo jako na multi-agentní systém, kde jednotlivé typy agentů mají svůj specifický úkol, různé pravomoci a schopnosti komunikovat s jinými agenty. Na základě této komunikace a určité „svobody“ rozhodování vzniká hierarchická struktura, schopná řešit komplikované problémy.

Tyto algoritmy byly použity například pro antivirovou ochranu u velkých a rozlehlých počítačových systémů.

1.2.6 Differential evolution (Diferenciální evoluce)

Diferenciální evoluce je poměrně nový typ evolučního algoritmu (od roku 1995), který vyvinuli a poprvé použili Ken Price a Rainer Storm při řešení problémů s Čebyševovými polynomy. Jeho princip je podobný algoritmům genetickým, s nimiž má několik společných rysů, jako je například tvorba potomků, používání generací, mutace apod.

Narozdíl od genetických algoritmů však k vytvoření dalšího potomka potřebuje ne dva, ale čtyři rodiče. Pro každého jedince jsou náhodně vybráni tři další nestejní jedinci z populace. Z těchto tří jedinců se vytvoří tzv. šumový vektor, který není ničím jiným, než-li mutací kombinace oněch tří, náhodně vybraných, rodičů. Poté dochází k procesu křížení – opět rozdíl oproti genetickým algoritmům, kde nejdříve probíhá křížení a teprve poté mutace potomka. Potomek, vzniklý v diferenciální evoluci křížením šumového vektoru se čtvrtým, doposud nepoužitým rodičem, s ním soutěží o místo v nové populaci.

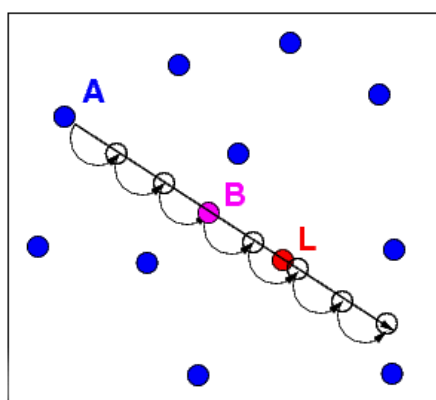
Dalším rozdílem oproti genetickým algoritmům je práce s dekadickými čísly namísto binárních. Výhodou algoritmu diferenciální evoluce je mimo jiné také to, že nastavení jeho parametrů citelně nesnižuje schopnost nalézt globální extrém.

Tento algoritmus, i přes své relativní mládí, byl použit s velkým úspěchem na poměrně bohatou množinu problémů a v komparativních studiích s již existujícími algoritmy si vedl stejně nebo lépe.

1.2.7 SOMA (Samo-organizující se migrační algoritmus)

Jedná se o algoritmus existující od roku 1999. Využívá pro svoji funkci některé podobné přístupy jako genetické algoritmy. Při běhu algoritmu se však nevytvářejí žádné nové jedinci, dochází pouze k jejich přemísťování ve stavovém prostoru. „Samoorganizace“ vzniká vzájemným ovlivňováním se jedinců populace při pohybu.

Výhodou tohoto algoritmu, proti genetickým algoritmům, je velice dobrá geometrická interpretovatelnost. Poloha nejlepšího řešení v populaci ovlivňuje změnu polohy ostatních jedinců v následujícím kroku. V tomto kroku se všichni jedinci, kromě nejlepšího, posunují v prostoru poloh několika skoky určité délky ve směru k nejlepšímu jedinci, ten zůstává na místě. V další populaci se každý jedinec usadí na místě, které bylo na jeho trase nejvýhodnější. Nakonec se z těchto jedinců vybere opět nejlepší jedinec a celý cyklus se opakuje.



Migrace jednoho jedince (A) směrem k leaderovi (L).

Jedinec se pohybuje skoky, přičemž ohodnocuje každou pozici, na kterou skočí.

Po projití celé trajektorie se usadí na té pozici ze všech, která měla nejlepší hodnotu (na obrázku bod B).

Obr. 3 Migrace jedince v algoritmu SOMA

Protože není SOMA založen na klasických principech evolučních algoritmů (křížení rodičů za vzniku potomstva), ale je založen na principu kooperace jedinců na problému prohledávání prostoru možných řešení, je častěji klasifikován jako algoritmus memetický,

tj. fungující na základě vztahů a principů. Vlastní název „memetický“ byl původně převzat z knihy *R. Dawkinse (1976)*.

Ačkoliv je algoritmus SOMA relativně nový, byl už úspěšně použitý při řešení reálných úloh. Například na plazmovém reaktoru na The Open University of Oxford ve Velké Británii.

1.3 Algoritmus Particle Swarm Optimisation (PSO)

1.3.1 Historické pozadí vzniku PSO algoritmu

Jak uvádí *Pomeroy (2003)*, PSO byl vyvinut sociálním psychologem Jamesem Kennedym a elektroinženýrem Russelem Eberhartem v roce 1995. Vyplýval z dřívějších pokusů s algoritmy, které modelovaly chování hejna (in orig.: „flocking behavior“), pozorovaného u mnohých druhů ptáků. Autoři si vybrali algoritmus, který v té době nebyl zrovna v centru pozornosti - model vytvořený biologem Frankem Heppnerem v roce 1990.

Heppnerovi „ptáci“ vykazovali většinu schémat chování stejných, jako i ostatní modely, Heppner však přidal něco navíc: Jeho ptáci byli přitahováni k hnízdišti (in orig.: „roosting area“). V simulaci se „ptáci“ nejdříve pohybovali bez určitého cíle a ve spontánně vytvořeném hejnu, dokud některý z jedinců nepřeletěl blízko hnízdiště. Jestliže programem nastavená hodnota „touhy přistát“ byla větší než hodnota „touhy zůstat v hejnu“, jedinec se odtrhl od hejna a přistál.

Pohyb ptáků (určení vektorů rychlosti a pozic) byl daný jednoduchými pravidly:

- každý jedinec se snaží dostat mezi nejbližší ptáky
- zároveň se však snaží do žádného z nich nenarazit

Tato pravidla měla za následek, že jedinec, který se odtrhl od hnízda a přistál, přitahoval blízko letící ptáky směrem k hnízdišti. Jakmile i tito ptáci objevili hnízdiště, přistáli v něm i oni a přitahovali směrem k němu další a další ptáky. A tak pořád dále, dokud zde nepřistáli všichni ptáci.

Je zřejmé, co právě na tomto modelu zaujalo Eberharta s Kennedym. Nalezení hnízdiště je analogické k nalezení řešení v prostoru možných řešení. A způsob, jakým ptáci, kteří již našli hnízdiště, „přitahují“ ostatní ptáky a zvyšují tak pravděpodobnost, že i oni je najdou, koresponduje se sociokognitivním vnímáním mysli, a tedy i intelligence.

Eberhart a Kennedy přepracovali Heppnerovu metodiku na problém pohybu jedinců nad prostorem řešení s cílem přistát na nejlepším z nich. Hlavní úskalí bylo vymyslet, jak docílit, aby jedinci přistáli ne na libovolném, ale právě na tom nejlepším řešení. Zde právě přichází do hry sociální aspekt mysli a inteligence.

Podstata (nebo přinejmenším významné hledisko) pohledu na mysl, jako sociální, je v tom, že každý jedinec se učí především z úspěchů svých sousedů. Srovnáváme se s ostatními a napodobujeme chování lidí, kteří jsou úspěšní ve věcech, na kterým nám záleží. Ale jak toto vše implementovat do algoritmu, jehož výstupem má být nalezení nejlepšího řešení?

Nezbytné je zcela určitě dosáhnout rovnováhy mezi objevováním nového (in orig.: „exploration“) a využívání zkušeností úspěchů jiných (in orig.: „exploitation“). Příliš malá míra objevování nového, a společnost se spokojí už s prvním objeveným, dobrým, ale ne nutně nejlepším řešením. Na druhou stranu malá míra využívání zkušeností společnosti vede k tomu, že společnost nikdy nenalezne společné optimální řešení.

Existuje i jiný pohled na věc: Spíše než „objevování“ a „využívání“, tj. chování, je potřeba vybalancovat „individualitu“ a „sociálnost“, vlastnosti, jež chování ovlivňují.

Díky výše uvedeným metaforám by se mohlo zdát, že jedinec, částice, musí mít nějakou formu inteligence. Mnohem přesnější je však představa částic, jako „nemyslících smítek“, které se pohybují (Eberhart a Kennedy používají výraz „létají“) n -rozměrným prostorem, prostorem řešení.

K určení jejich „individuality“ a „sociálnosti“ a jejich uplatnění v algoritmu se dostaneme v další kapitole, kde nastíníme fungování algoritmu PSO.

1.3.2 Základní princip PSO

Jak již bylo zmíněno, algoritmus PSO simuluje sociální chování jako je pohyb hejna ptáků v prostoru, obecně tedy pohyb částic v n -rozměrném prostoru. Stejně jako evoluční algoritmy používá PSO k hledání řešení populaci (roj, in orig.: „swarm“) jedinců (částic, in orig.: „particles“), která se s každou další iterací algoritmu aktualizuje. S částicí je nakládáno jako s bodem v n -rozměrném prostoru a stav částice je charakterizován jeho polohou a rychlostí (n -rozměrné vektory).

Označme vektor polohy částice i v iteraci t jako:

$$X_i(t) = \{x_{i1}(t), x_{i2}(t), \dots, x_{in}(t)\} \quad (1)$$

a vektor rychlosti stejné částice jako:

$$V_i(t) = \{v_{i1}(t), v_{i2}(t), \dots, v_{in}(t)\} \quad (2)$$

Pohyb každé částice je ovlivňován její „individuální zkušeností“ a „sociální zkušeností“. Individuální zkušenost je reprezentována vektorem místa, na kterém nabyla částice během svého dosavadního pohybu v prostoru nejlepší řešení (osobní minimální hodnoty účelové funkce, HUF), označme jej X_i^L . Sociální zkušenost je reprezentována „nejlepším“ z těchto vektorů, tzn. takovým, jehož hodnota účelové funkce je ze všech vektorů X_i^L nejnižší. Označme tento vektor X^G . Tento vektor reprezentuje nejlepší řešení (zkušenost) celého roje.

Mechanismus aktualizace stavů částic v jednom iteračním kroku algoritmu pak vypadá takto:

$$V_i(t) = w(t)V_i(t-1) + c_1 r_1 (X_i^L - X_i(t-1)) + c_2 r_2 (X^G - X_i(t-1)) \quad (3)$$

$$X_i(t) = V_i(t) + X_i(t-1) \quad (4)$$

kde $i = 1, 2, \dots, P$ a P je počet částic v roji, též nazývaný velikost populace; $t = 1, 2, \dots, T$ a T je počet iterací algoritmu; c_1 a c_2 jsou kladné konstanty, tzv. faktory učení, r_1 a r_2 náhodná čísla z intervalu $[0, 1]$ a $w(t)$ je váha omezující vliv předcházejících rychlostí na rychlost současnou. Parametrům c_1 , c_2 a $w(t)$ se budeme zabývat v další kapitole.

Rovnice (3) je použita pro výpočet nové rychlosti částice podle její rychlosti v minulém iteračním kroku a podle vzdálenosti od osobního a skupinového dosavadního nejlepšího řešení. Míra „individuality“ a „sociálnosti“ je dána parametry r_1 a r_2 , v každém iteračním kroku je tedy částice ve svém chování někdy více individualistická, někdy více sociální, někdy obojí a někdy ani jedno. Jak udává *Pomeroy (2003)*, zdá se, že částice mají hodně společného s teenagery.

Rovnice (4) je použita pro výpočet nové polohy částice, využívá její dosavadní zkušenosti (uplatnění X_i^L), stejně jako zkušenosti celého roje (uplatnění X^G). Rovnice (3) a (4) také přehledně odrážejí mechanismus sdílení informací v PSO.

1.3.3 Popis celého algoritmu PSO

Algoritmus PSO můžeme v jednotlivých krocích popsat takto:

1. Stanovení parametrů – jde o parametry, které určují chod celého algoritmu. Jsou to parametry P – počet částic v roji, T – počet iterací, $X_{\min}$ a $X_{\max}$ – hodnoty, resp. vektory, omezující prostor řešení, $V_{\min}$ a $V_{\max}$ – hodnoty, resp. vektory omezující velikost počítané rychlosti (pro udržení faktoru chaosu při výpočtu rychlosti v rozumné míře), c_1 a c_2 – váhy vlivu individualistického a sociálního chování, w – váha vlivu předchozích rychlostí a n – počet dimenzí.
2. Inicializace roje – vygenerování matice počátečních pozic částic roje $P \times n$ pomocí pseudonáhodné veličiny na intervalu $[X_{\min}, X_{\max}]$ a matice počátečních rychlostí částic roje $P \times n$ pomocí pseudonáhodné veličiny na intervalu $[V_{\min}, V_{\max}]$, vyhodnocení počátečních podmínek k získání n hodnot X_i^L a jedné X^G .
3. Migrační cyklus algoritmu, který sestává z:
 - a. vypočtení nových rychlostí $V_i(t)$ dle rovnice (3) a jejich přizpůsobení intervalu $[V_{\min}, V_{\max}]$.
 - b. vypočtení nových pozic částic $X_i(t)$ dle rovnice (4) a jejich přizpůsobení intervalu $[X_{\min}, X_{\max}]$.
 - c. vyhodnocení aktuálních podmínek k aktualizaci hodnot X_i^L a X^G .
4. Vyhodnocení – celý proces migračního cyklu se opakuje, dokud neproběhne zadaný počet iterací nebo dokud HUF nedosáhne požadované hodnoty. Během každého iteračního kroku se zaznamenává hodnota X^G , která po ukončení znázorňuje průběh optimalizačního procesu.

1.3.4 Varianty PSO

Algoritmus PSO se vyskytuje v mnoha různých variantách. Zmíníme na těchto řádcích alespoň dva z nich – původní verzi algoritmu PSO a přístup k výběru parametrů PSO definovaného rovnicemi (3) a (4).

Původní algoritmus dle Eberharta a Kennedyho, Pomeroy (2003)

Tak jako v Heppnerově modelu chování hejna ptáků je chování jedince ovlivněno pouze jedinci v jeho okolí (každý jedinec se snaží dostat mezi nejbližší ptáky, zároveň se však

snaží do žádného z nich nenarazit) a ne chováním celého hejna, je i v původním algoritmu Eberharta a Kennedyho při předávání sociální zkušenosti počítáno jen se zkušeností nejbližších „sousedů“.

Eberhart a Kennedy zjistili, že pro jisté typy optimalizačních problémů „globální sousedství“ (všichni jedinci ovlivňují všechny) zvyšuje pravděpodobnost konvergence k méně než optimálnímu řešení. Naopak použití malých, překrývajících se „sousedství“, namísto „globálního sousedství“, bývá často efektivnější.

Například, pro sedm jedinců **a**, **b**, **c**, **d**, **e**, **f** a **g** dochází při vytvoření sousedství **gab**, **abc**, **bcd**, **cde**, **def** a **fga** k objevování větší části prostoru řešení, než pro sousedství **abcdefg**. Výše zmíněná, tříprvková topologie sousedů má tendence ke zpomalování konvergence a tak poskytuje více prostoru pro objevování.

Jak bylo definováno, sousedy částice **a** jsou **g** a **b**. Ty ovlivňují chování částice **a**. Zároveň ale patří do svých sousedství (**fga** a **abc**) a budou se pohybovat pod vlivem zkušenosti svých sousedů. Tento způsob umožňuje pomalé šíření nejlepšího řešení všemi sousedstvími, tedy celým rojem. Podstatné je, že šíření zabere nějaký čas. Během něj částice dosud nezasažené touto informací stále hledají „své“ nejlepší řešení. To zvyšuje šanci, že pokud je v jejich blízkém okolí nějaké nové „globální nejlepší řešení“, tyto částice je naleznou.

Je důležité zmínit, že narozdíl od Heppnerova modelu nemá koncept „sousedství“ v PSO nic společného s reálnými vzdálenostmi částic v prostoru řešení. (Stejně jako lidé, kteří cestují po světě, ale žijí v sousedních domech, také zůstávají sousedy.) Prohledávání „v rámci sousedství“ se tedy může uplatňovat i v navzájem vzdálených částech prostoru řešení.

Přístup k Výběru parametrů (Fukuyama)

PSO dle rovnice (3) a (4) má několik explicitních parametrů, jejichž různým nastavením je možné dosáhnout různých způsobů, jimiž PSO prohledává prostor řešení. Jsou to:

c_j : faktory učení, $j \in \{1,2\}$

$w(t)$: váha omezující vliv předcházejících rychlostí na rychlost současnou

váha je často používána jako funkce:

$$w(t) = w_{\max} - \frac{w_{\max} - w_{\min}}{T} * t, \quad (5)$$

kde $w_{\max}$ je počáteční váha, $w_{\min}$ je koncová váha, T je celkový počet iterací a t je aktuální iterace.

Shi a Eberhart provedli výzkum výběru parametrů výše uvedených. Na základě jejich výsledků, jsou následující hodnoty vhodné a nezávislé na řešeném problému:

$$c_j = 2.0, w_{\max} = 0.9, w_{\min} = 0.4.$$

II. PRAKTICKÁ ČÁST

2 EXPERIMENTÁLNĚ-SIMULAČNÍ ČÁST

Předmětem této části práce je ověření funkčnosti algoritmu PSO, a to na:

- několika testovacích funkcích, běžně užívaných k testování evolučních algoritmů
- na dvou již řešených problémech z praxe

Pro testování jsme použili variantu PSO dle rovnic (3) a (4) s úpravou váhové funkce dle (5).

Algoritmus pro výpočet byl sestaven v programovém prostředí *Mathematica* (viz. příloha č.1 – CD-ROM).

2.1 Testování funkčnosti algoritmu PSO na několika vybraných testovacích funkcích

Testování proběhlo s těmito parametry:

Počet jedinců v roji: 20

Počet iterací: 1000

Nastavení vah a učících faktorů: $c_j = 2.0$, $w_{\max} = 0.9$, $w_{\min} = 0.4$

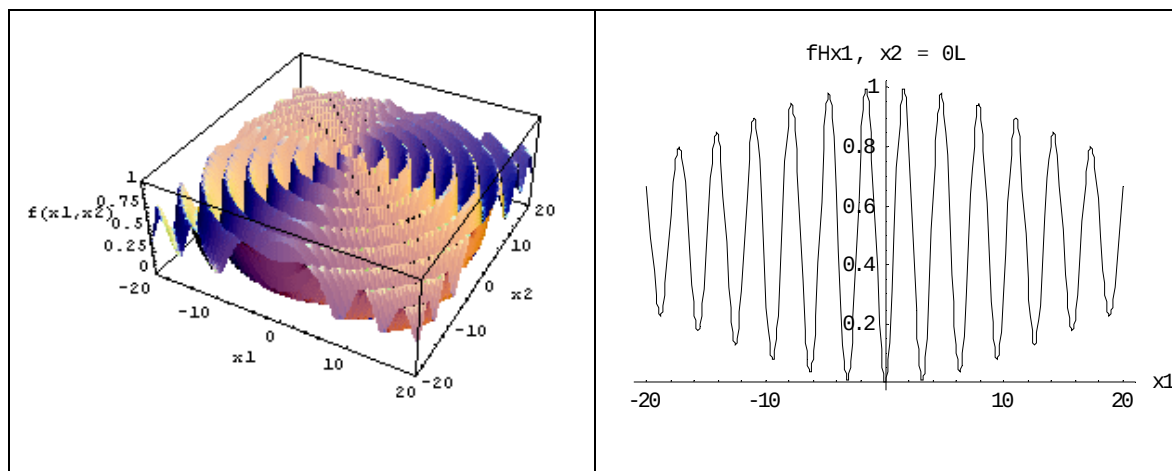
Počet dimenzí: 2

Každý výpočet byl stokrát zopakován, u každé funkce je uveden průběh grafu globálního nejlepšího řešení – hodnoty účelové funkce v X^G a tabulka s nejlepším a nejhorším výsledkem, celkovým průměrem a procentuální úspěšností dosažení globálního minima po 1000 iteracích. Za úspěšné dosažení globálního minima je považováno získání nižší hodnoty účelové funkce než $1E-9$ (pro dané testovací funkce zaručeně vždy splňuje podmínku výskytu vektoru X^G v „úzkém“ okolí globálního minima). Testovací funkce byly vybrány z [13], [14], [15].

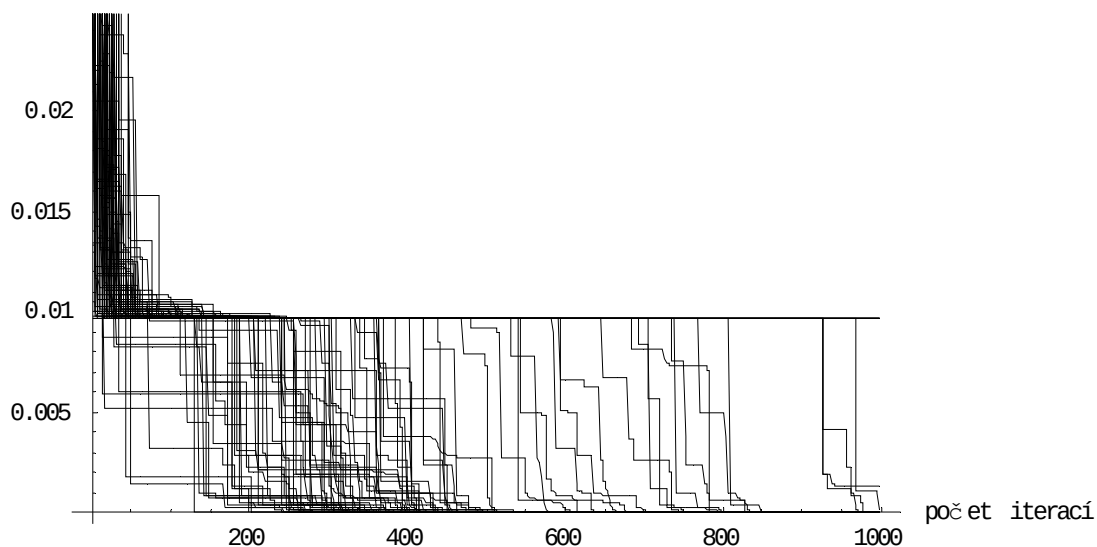
2.1.1 Funkce f6

$$0,5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0,5}{\left(1 + 0,001 * (x_1^2 + x_2^2)\right)^2} \quad (6)$$

Funkce byla testována na intervalu $[-20, 20]$, následují grafy pro přiblížení jejího průběhu..

Obr. 4 Graf funkce f_6 , ve 2D graf funkce $f_6(x_1, x_2=0)$

globalní minimum

Obr. 5 Graf globálního nejlepšího řešení PSO na funkci f_6 Tab. 2 Výsledky testu PSO na funkci f_6

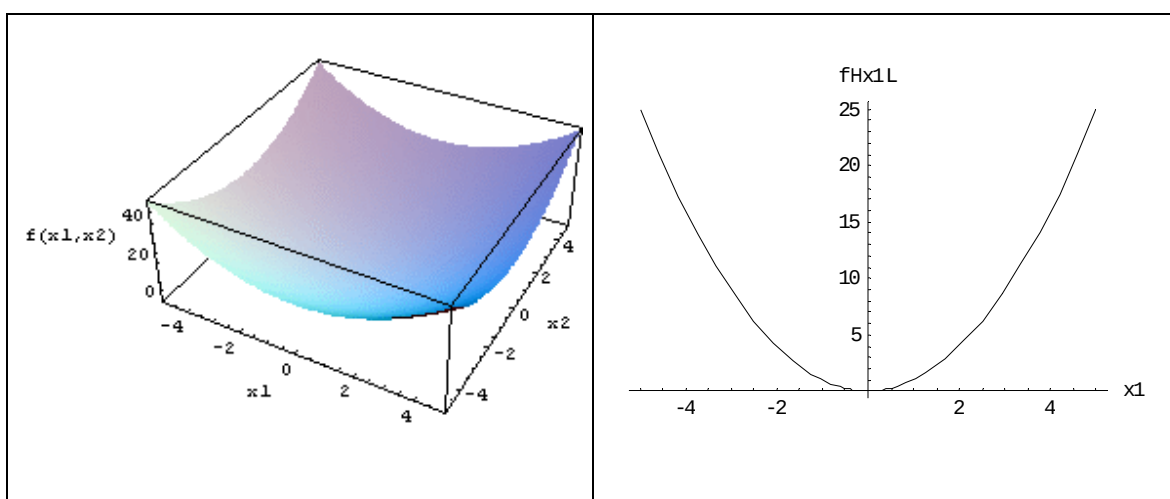
Nejlepší výsledek	0
Nejhorší výsledek	0.00971591
Celkový průměr	0.00127819
Procentuální úspěšnost	83%

Z výsledků vyplývá, že PSO se s tímto typem problému vypořádává docela dobře, při zvýšení počtu iterací by snad dosáhl ještě lepších výsledků

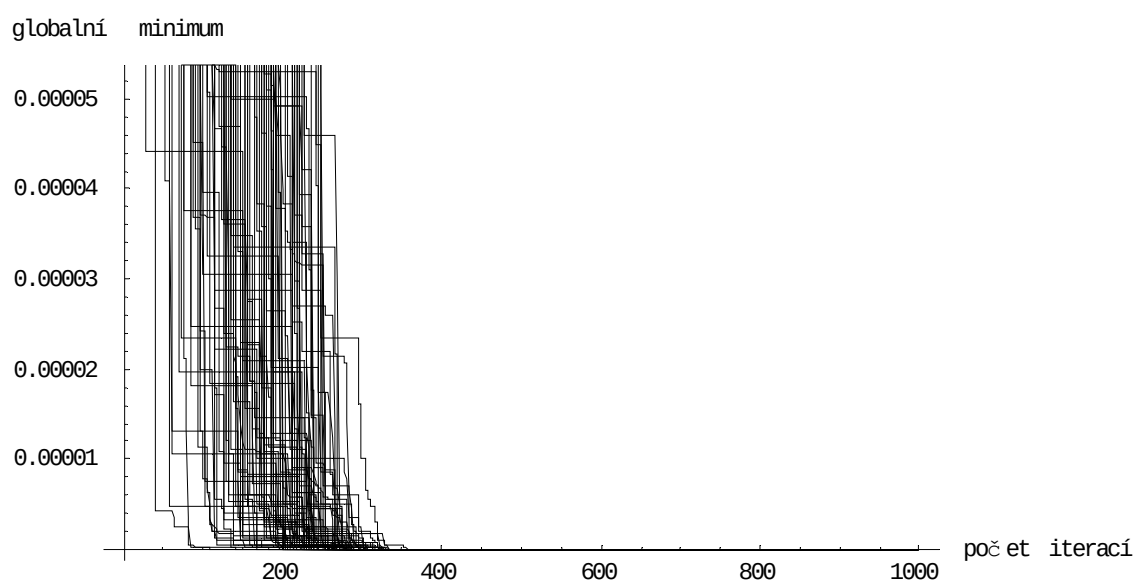
2.1.2 Funkce Sphere (první De Jongova funkce)

$$\sum_{i=1}^{Dim} x_i^2 \quad (7)$$

Funkce byla testována na intervalu $[-5, 5]$, následují grafy pro přiblížení jejího průběhu. Funkce má velmi jednoduchý průběh a jedno globální minimum v počátku.



Obr. 6 Graf funkce Sphere



Obr. 7 Graf globálního nejlepšího řešení PSO na funkci Sphere

Tab. 3 Výsledky testu PSO na funkci Sphere

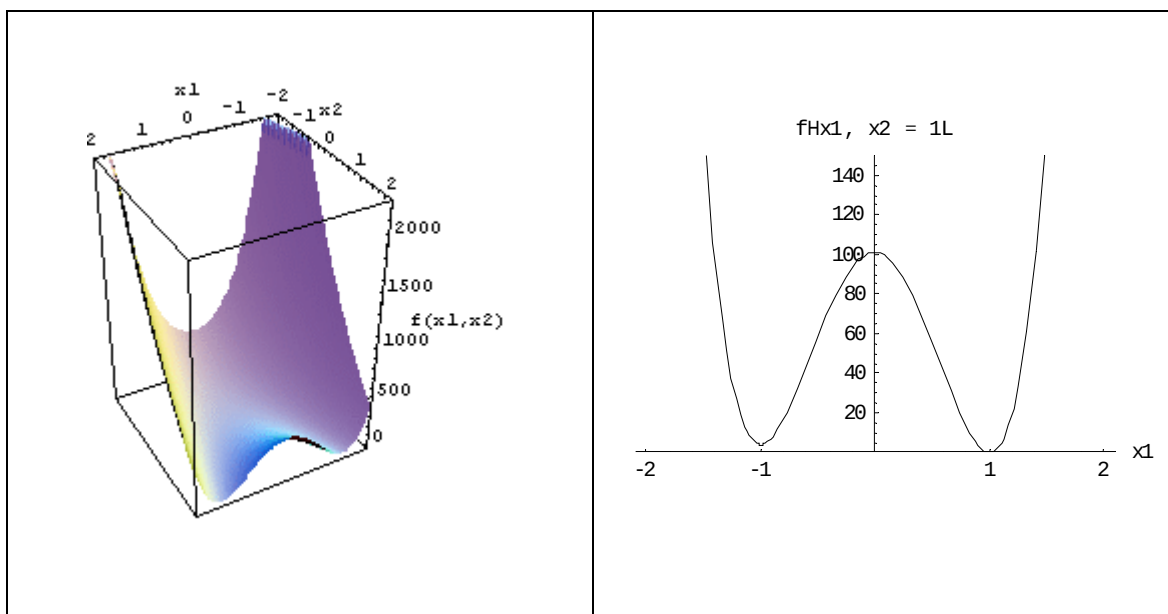
Nejlepší výsledek	6.2674e-72
Nejhorší výsledek	7.5733e-59
Celkový průměr	1.6214e-60
Procentuální úspěšnost	100%

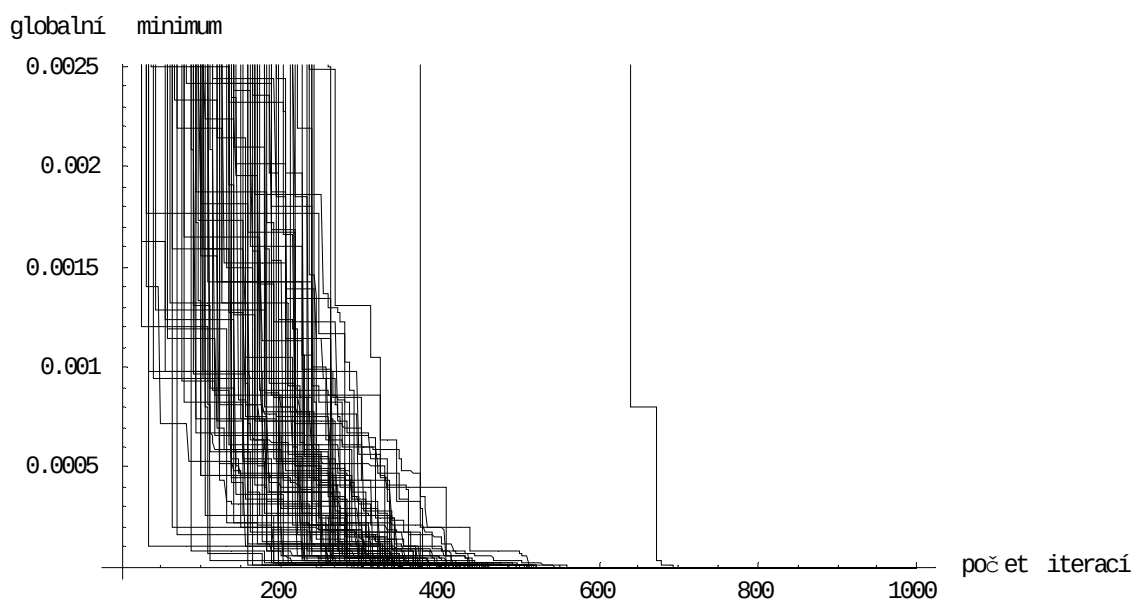
Z výsledků plyne, že s touto jednoduchou funkcí si PSO poradil bez problémů zpravidla během méně než 400 iterací.

2.1.3 Rosenbrockova funkce (Rosenbrockovo sedlo, druhá De Jongova funkce)

$$\sum_{i=1}^{Dim-1} \left[100 \cdot (x_i^2 - x_{i+1})^2 + (1 - x_i)^2 \right] \quad (8)$$

Funkce byla testována na intervalu $[-2, 2]$, následují grafy pro přiblížení jejího průběhu.

Obr. 8 Graf Rosenbrockovy funkce, 2D graf zobrazuje řez 3D grafem v $x_2 = 1$



Obr. 9 Graf globálního nejlepšího řešení PSO na Rosenbrockově funkci

Tab. 4 Výsledky testu PSO na Rosenbrockově funkci

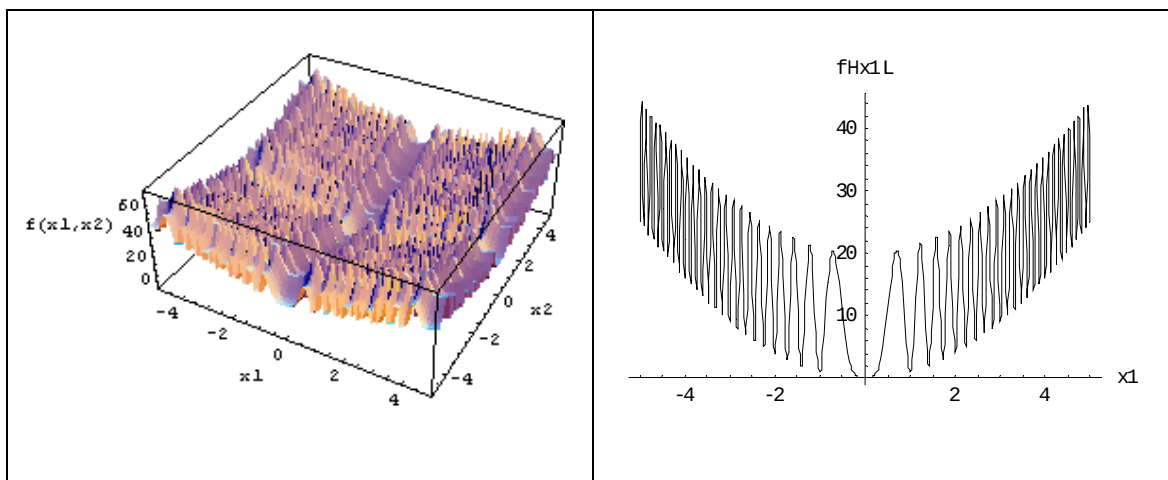
Nejlepší výsledek	1.4608e-23
Nejhorší výsledek	2.0092e-12
Celkový průměr	2.6837e-14
Procentuální úspěšnost	100%

Rosenbrockova funkce má jediný extrém v bodě [1, 1], nacházející se na jedné straně „sedla“ funkce. Na druhé straně sedla se nachází lokální minimum s jen o málo vyšší funkční hodnotou. Tento fakt může činit funkci obtížnou pro optimalizaci. Jak je vidět, PSO se s tímto problémem dokáže důstojně vypořádat.

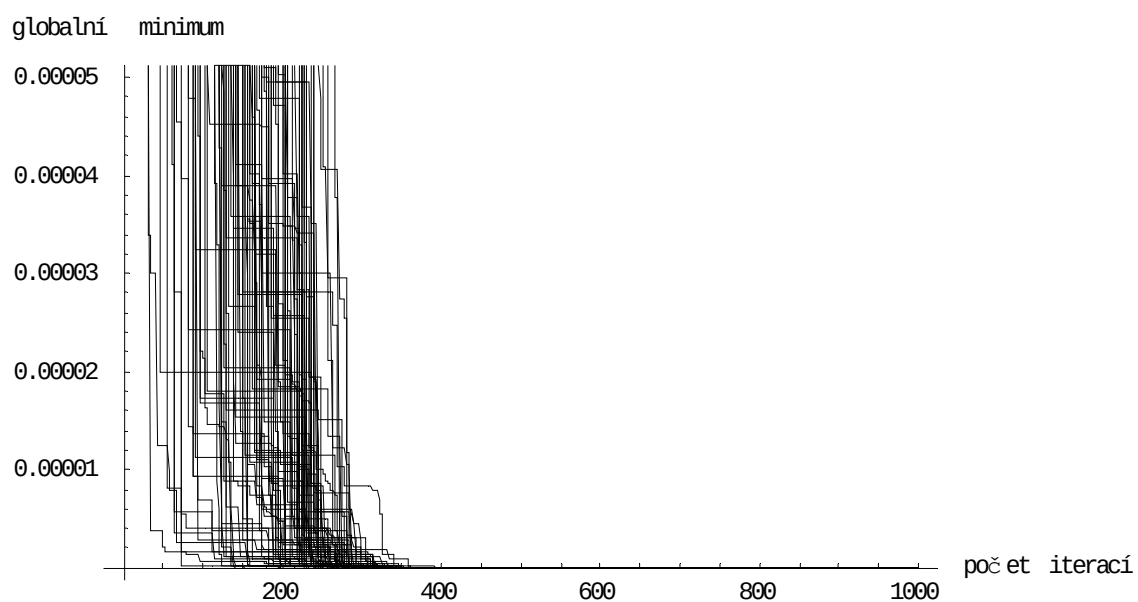
2.1.4 Rastriginova funkce

$$10 + \sum_{i=1}^{Dim} \left[x_i^2 - 10 \cdot \cos(2 \cdot p \cdot x_i^2) \right] \quad (9)$$

Funkce byla testována na intervalu [-5, 5], následují grafy pro přiblížení jejího průběhu



Obr. 10 Graf Rastrigrinovy funkce



Obr. 11 Graf globálního nejlepšího řešení PSO na Rastrigrinově funkci

Tab. 5 Výsledky testu PSO na Rastrigrinově funkci

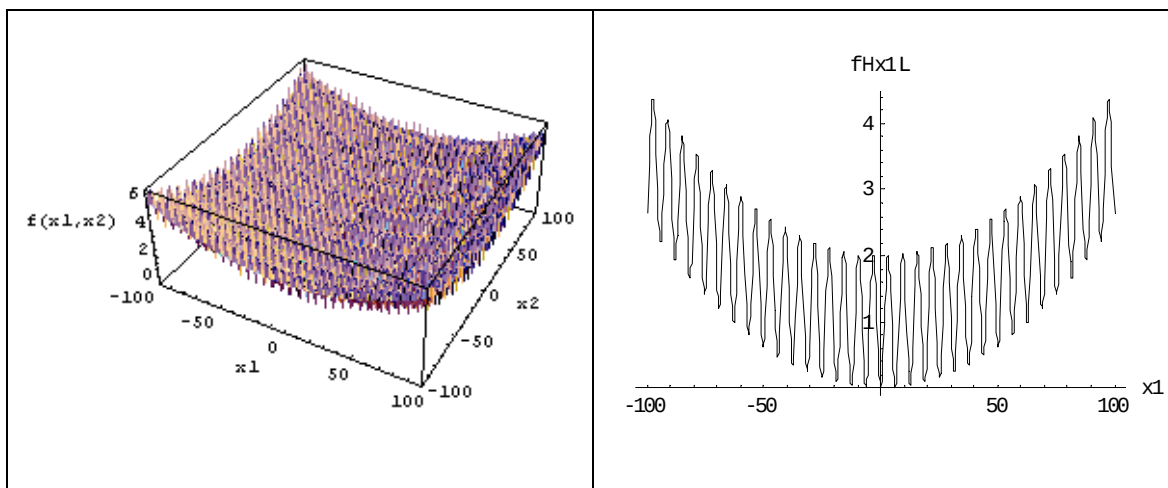
Nejlepší výsledek	0
Nejhorší výsledek	0
Celkový průměr	0
Procentuální úspěšnost	100

Z výsledků vyplývá, že PSO dokáže velmi dobře optimalizovat problémy tohoto typu.

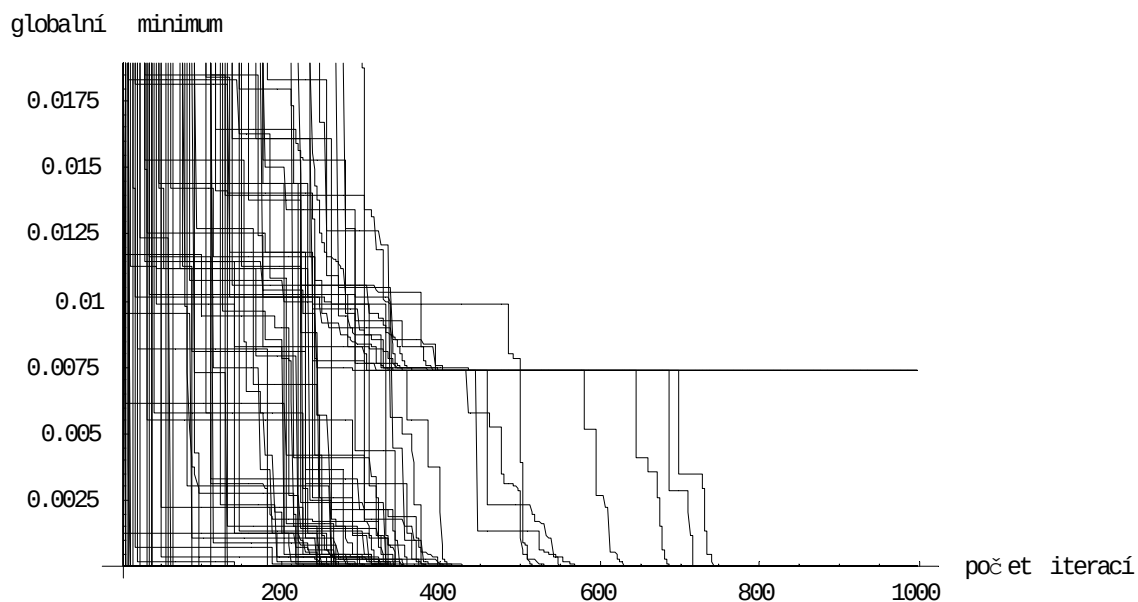
2.1.5 Griewankova funkce

$$\sum_{i=1}^{Dim} \frac{x_i^2}{4000} - \prod_{i=1}^{Dim} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (10)$$

Funkce byla testována na intervalu $[-20, 20]$, následují grafy pro přiblížení jejího průběhu.



Obr. 12 Graf Griewankovy funkce



Obr. 13 Graf globálního nejlepšího řešení PSO na Griewankově funkci

Tab. 6 Výsledky testu PSO na Griewankově funkci

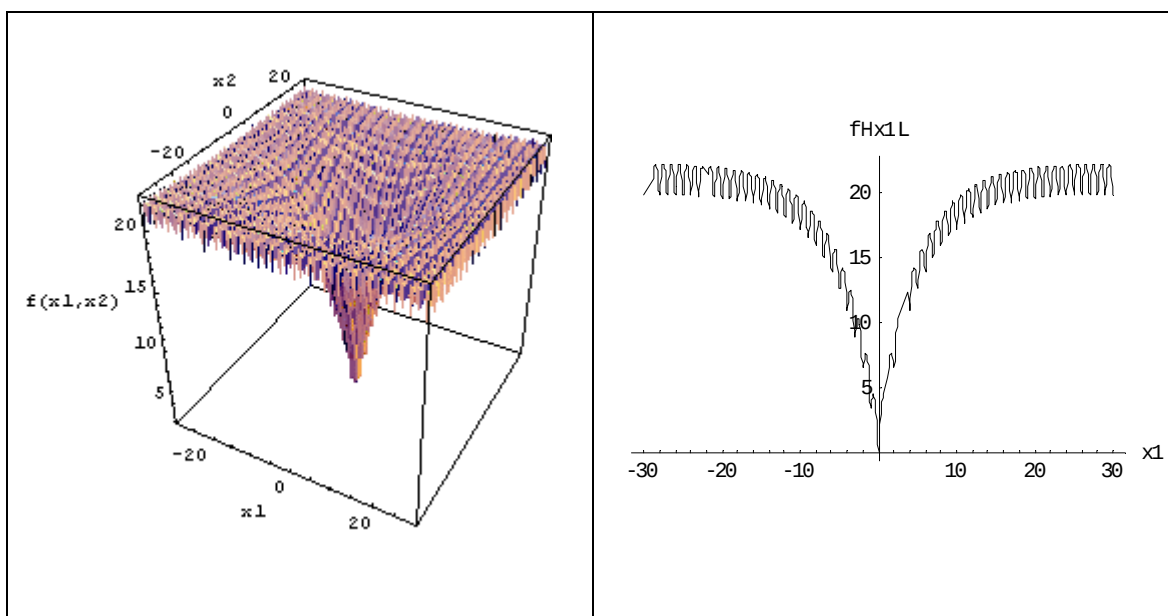
Nejlepší výsledek	0.
Nejhorší výsledek	0.00739604
Celkový průměr	0.000369802
Procentuální úspěšnost	95%

V případě optimalizování funkce, jež má mnoho extrémů o nepříliš rozdílných funkčních hodnotách, dosahuje PSO mírně horších výsledků než v předešlém případě.

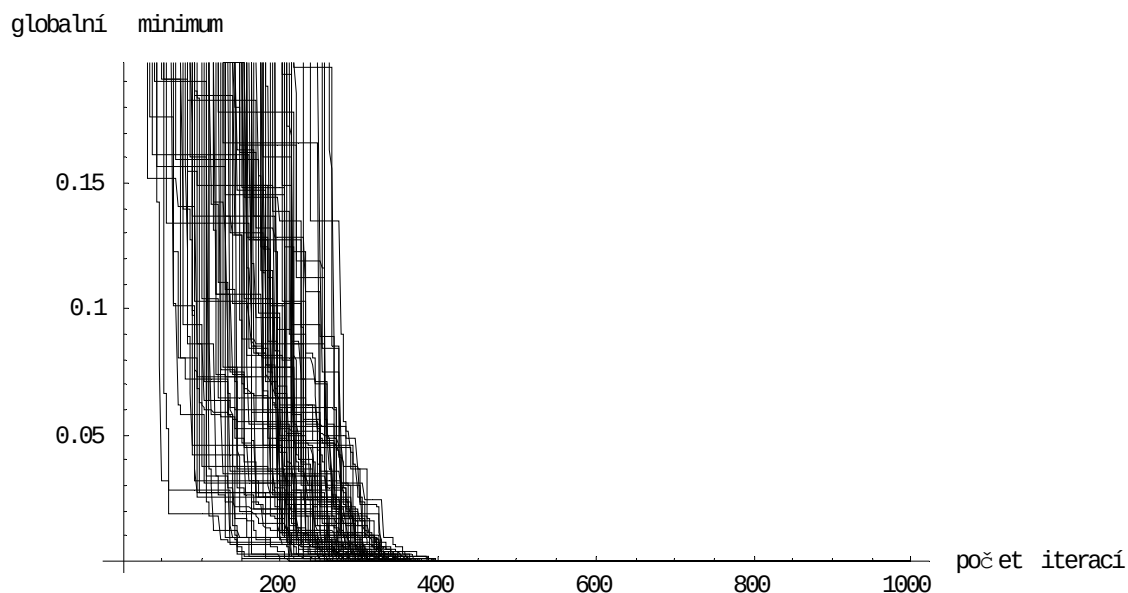
2.1.6 Ackleyova funkce

$$20 + e - 20 \cdot e^{-0,2 \sqrt{\frac{1}{Dim} \sum_{i=1}^{Dim} x_i^2}} - e^{\frac{1}{Dim} \sum_{i=1}^{Dim} \cos(2 \cdot \pi \cdot x_i)} \quad (11)$$

Funkce byla testována na intervalu $[-30, 30]$, následují grafy pro přiblížení jejího průběhu



Obr. 14 Graf Ackleyovy funkce



Obr. 15 Graf globálního nejlepšího řešení PSO na Ackleyově funkci

Tab. 7 Výsledky testu PSO na Ackleyově funkci

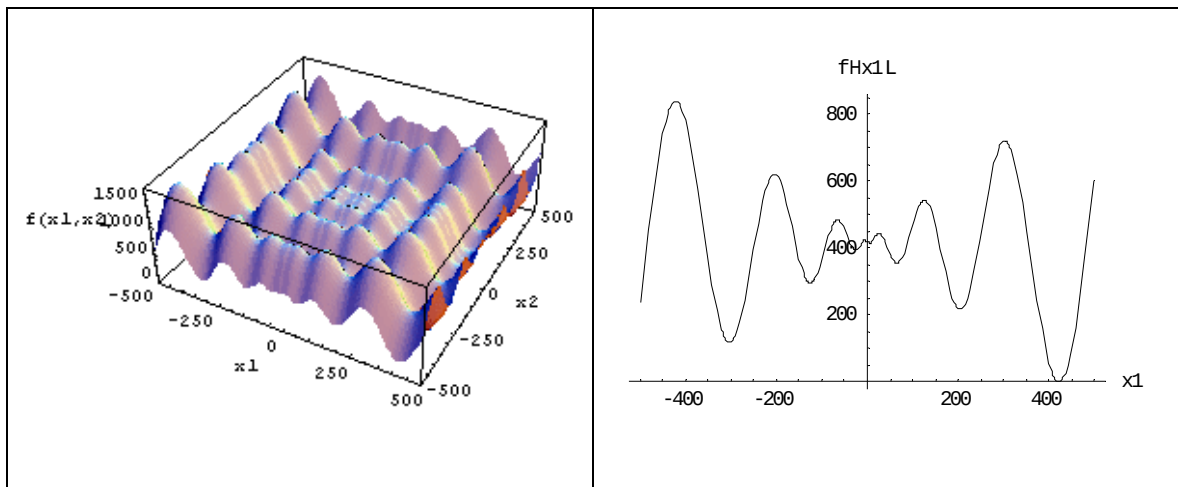
Nejlepší výsledek	0
Nejhorší výsledek	0
Celkový průměr	0
Procentuální úspěšnost	100

Jak je z výsledků testování patrné, PSO dokáže velmi dobře optimalizovat i funkci tohoto typu.

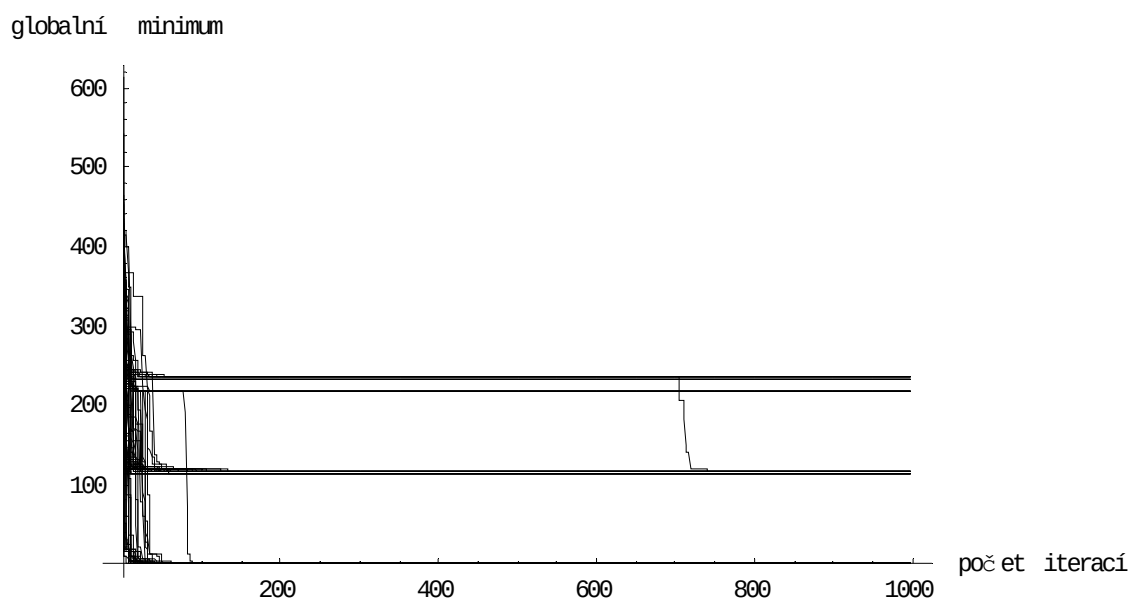
2.1.7 Schwefelova funkce

$$418,9829 \cdot Dim + \sum_{i=1}^{Dim} \left(-x_i \cdot \sin \sqrt{|x_i|} \right) \quad (12)$$

Funkce byla testována na intervalu $[-512, 512]$, následují grafy pro přiblížení jejího průběhu. Funkce je upravená tak, aby na tomto intervalu nabývala globálního minima o hodnotě 0.



Obr. 16 Graf Schwefelovy funkce



Obr. 17 Graf globálního nejlepšího řešení PSO na Schwefelově funkci

Tab. 8 Výsledky testu PSO na Schwefelově funkci

Nejlepší výsledek	3.2345e-11
Nejhorší výsledek	236.877
Celkový průměr	98.9356
Procentuální úspěšnost	33

V případě Schwefelovy funkce dosáhl PSO nejhorších výsledků – při opakovaném spuštění konvergoval k jednomu ze tří nejnižších extrémů. Nevypořádal se tak vždy se záludností této funkce, tj. s tím, že optima funkce přibývají střídavě ve směru od počátku souřadnic. PSO tedy zřejmě nepatří mezi nejlepší optimalizační algoritmy tohoto typu úloh. Při několikanásobném opakování optimalizace ale šance na nalezení globálního extrému roste.

2.2 Testování funkčnosti algoritmu PSO na již řešených problémech

Oba problémy byly převzaty z literatury [6], strana 118 a další. Autor zde srovnává výsledky algoritmu SOMA a dříve použitých metod k řešení těchto testovacích problémů.

Tab. 9 Alternativní metody použité k řešení testovacích problémů

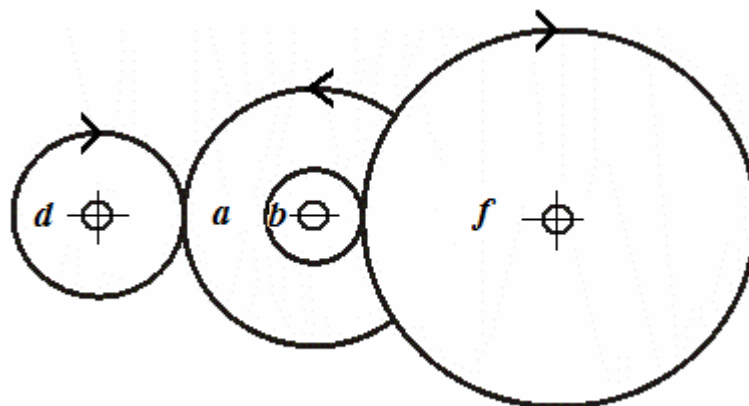
Dříve použité metody	
Publikováno	Metoda
Sandgren	Branch & Bound pomocí sekvenčního kvadratického programování
Fu, Fenton & Gleghorn	Celočíselné-diskrétní-reálné nelineární programování
Loh & Papalambros	Sekvenční lineární algorytmus
Zhang & Wang	Simulované žíhání
Wu & Chow	Meta-genetický algoritmus
Lin, Zhang & Wang	Modifikovaný genetický algoritmus
Cao & Wu	Evoluční programování
Lampien & Zelinka	Diferenciální evoluce

2.2.1 Příklad 1: Návrh ozubeného kola

Tento příklad byl zaměřen na optimalizaci poměru převodu čtyř ozubených kol. Poměr převodu je definován jako poměr úhlové rychlosti osy řízeného kola k úhlové rychlosti osy řízeného kola. Tento poměr i_{tot} je definován jako

$$i_{tot} = \frac{\omega_0}{\omega_i} = \frac{z_d z_b}{z_a z_f} \quad (13)$$

Kde ω_0 a ω_i byly jednotlivé úhlové rychlosti a z znamená počet zubů na každém kole. Cílem optimalizace bylo nalézt počet zubů d , a , b a f na všech kolech takový, aby i_{tot} bylo co nejbližší požadovanému poměru $i_{trg} = 1/6.931$ ($=0.1443$). Minimální povolený počet zubů byl 12 a maximální 60.



Obr. 18 Ozubený převod z příkladu 1

Formulace problému včetně účelové funkce zněla:

Nalézt

$$X = (x_1, x_2, x_3, x_4) = (z_d, z_b, z_a, z_f), \quad x \in \{12, 13, \dots, 60\}$$

tak, aby minimalizovaly:

$$f(X) = (i_{trg} - i_{tot})^2 = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4} \right) \quad (14)$$

vzhledem k omezením

$$12 \leq x_i \leq 60, \quad i=1,2,3,4.$$

Cílem tedy bylo nalézt takovou kombinaci všech čtyř argumentů účelové funkce, aby tato na nich nabývala svého minima. Účelová funkce byla nadefinována jako druhá mocnina rozdílu mezi požadovaným poměrem převodu i_{trg} a aktuálně vypočítaným i_{tot} . Jednotlivé argumenty podléhaly pouze omezením ve formě hranic, mezi nimiž mohly nabývat libovolných hodnot.

Algoritmus PSO byl použit s parametry: velikost roje: 100 jedinců a počet iterací: 300.

Algoritmus byl spuštěn celkem stokrát a výsledkem byly tyto hodnoty účelové funkce:

nejhorší výsledek: 2.7264×10^{-8}

průměrný výsledek: 3.0695×10^{-9}

nejlepší výsledek: 2.3078×10^{-11}

Nejnižší hodnotě UF odpovídalo řešení:

počty zubů $d = 30$, $b = 13$, $a = 51$, $f = 53$ a převodní poměr 0.144284.

Při srovnání s jinými algoritmy (Tab. 10) lze dojít k závěru, že PSO podává lepší výsledky, než řada jiných algoritmů (tedy je funkční), nicméně pro tuto problematiku zřejmě existují i algoritmy vhodnější, než je PSO.

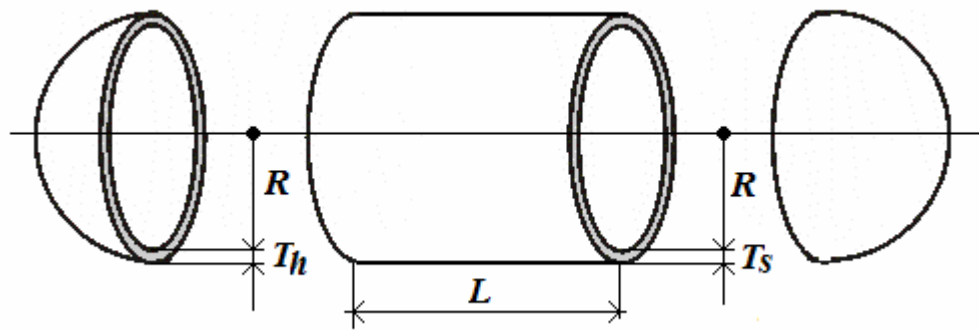
Tab. 10 Optimální řešení získané jinými algoritmy než PSO

<i>Položka</i>	<i>Optimální řešení</i>			
	<i>Sandgren</i>	<i>Fu et al.</i>	<i>Loh a Papalambros</i>	<i>Zhang a Wang</i>
<i>f(x)</i>	5.7×10^{-6}	4.5×10^{-6}	0.233×10^{-6}	2.36×10^{-9}
<i>Poměr převodu</i>	0.146666	0.146411	0.144762	0.144231

<i>Položka</i>	<i>Optimální řešení</i>			
	<i>Lin a další</i>	<i>Wu a Chow</i>	<i>Cao a Wu</i>	<i>Lampinen a Zelinka</i>
<i>f(x)</i>	2.7×10^{-12}	2.7×10^{-12}	2.36×10^{-9}	2.7×10^{-12}
<i>Poměr převodu</i>	0.144281	0.144281	0.144231	0.144281

2.2.2 Příklad 2: Návrh tlakové nádoby

Cílem tohoto příkladu byl design nádoby pro stlačený vzduch, pracující s tlakem 3000 psi a minimálním objemem 750 ft^3 . V tomto experimentu byla jako účelová funkce zvolena cena za tlakovou nádobu. Smyslem evoluční optimalizace tedy bylo tuto cenu co nejvíce snížit. Cena této nádoby byla dána cenou materiálu, cenou za sváření a jejím utvářením. Jednotlivé proměnné pro návrh nádoby byly následujícího charakteru – L a R spojité (in orig.: „Real“), T_s a T_h diskrétní. Tloušťka nádoby, T_s , a polokoulí T_h , byly ve standardní velikosti.



Obr. 19 Tlaková nádoba z příkladu 2

V tomto příkladu byla tloušťka ocelového obalu nádoby dostupná v násobcích 0.0625 palců (in orig.: „inch“). Vlastní problém byl definován následovně:

Nalézt

$$X = (x_1, x_2, x_3, x_4) = (T_s, T_h, R, L)$$

minimalizující

$$f(X) = 0.6224x_1 x_3 x_4 + 1.778 x_2 x_3^2 + 3.1611 x_1^2 x_4 + 19.84x_1^2 x_3$$

vzhledem k omezením

$$g_1(X) = 0.0193 x_3 - x_1 \leq 0$$

$$g_2(X) = 0.00954 x_3 - x_2 \leq 0$$

$$g_3(X) = 750.0 \times 1728.0 - \pi x_3^2 x_4 - 4/3 \pi x_3^3 \leq 0$$

$$g_4(X) = x_4 - 240.0 \leq 0$$

$$g_5(X) = 1.1 - x_1 \leq 0$$

$$g_6(X) = 0.6 - x_2 \leq 0$$

V této definici minimalizovaná funkce $f(X)$ reprezentuje finální výrobní cenu tlakové nádoby, jako funkci parametrů nádoby. Omezení $g_1, \dots, g_6$, mají význam restrikcí (ohraničení), ve kterých se evoluční optimalizace musela držet. Tato omezení nebyla zvolena samoučelně. Například minimální tloušťka stěny T_s (g_1) a polokouli T_h (g_2), s ohledem na poloměr a objem, je limitována tlakem, který bude v nádobě. Rovněž délka L (g_4) byla dána výrobními možnostmi – maximálně 20 stop (in orig.: „feet“, 1ft = cca 30

cm). Objem nádoby byl povolen minimálně na 750 ft^3 (g_3). Tloušťka T_s neměla být menší, než 1.1 palce (g_5) a tloušťka polokoulí minimálně 0.6 palců (g_6).

Pro srovnání s algoritmem PSO jsme vybrali Případ A, kde jsou všechny argumenty nadefinovány jako reálná čísla. Parametry algoritmu PSO byly: velikost roje: 40 jedinců a počet iterací: 500.

Algoritmus byl spuštěn celkem stokrát a výsledkem byly tyto hodnoty účelové funkce, tedy výrobní ceny tlakové nádoby v USD:

nejhorší výsledek: **7115.91**

průměrný výsledek: **7062.29**

nejlepší výsledek: **7020.95**

Nejnižší ceně odpovídalo řešení:

$T_s = 1.1$, $T_h = 0.6$, $R = 56.982$, $L = 51.1$

Po srovnání s ostatními algoritmy (Tab. 11) lze říci, že PSO v tomto příkladě podává velice slušné výsledky.

Tab. 11 Optimální řešení tlakové nádoby, případ A

<i>Položka</i>	<i>Optimální řešení</i>		
	<i>Sandgren</i>	<i>Fu et al.</i>	<i>Lampien & Zelinka</i>
<i>$f(X)$ [\$]</i>	7867.0	7790.588	7019.031

2.3 Zhodnocení experimentálně-simulační části

Na sedmi testovacích funkcích a dvou příkladech jsme ukázali, že algoritmus PSO je plnohodnotný evoluční algoritmus, dosahující poměrně kvalitních výsledků, jediný výraznější problém měl s optimalizací Schwefelovy funkce. Pro důkladnější zhodnocení pozitiv a negativ PSO by ale bylo nutné vypracovat studii, překračující rozsah této práce.

ZÁVĚR

Cílem této práce bylo nahlédnout do problematiky moderních evolučních metod, zpracovat, popsat a v programovém prostředí *Mathematica* sestavit algoritmus Particle Swarm. Dalším cílem bylo přesvědčit se, na základě již řešených problémů, o jeho funkčnosti.

V první části jsme poskytli přehled a rozdělení optimalizačních algoritmů. Uvedli jsme uspořádání optimalizačních algoritmů dle *Zelinky (2002)* na enumerativní, deterministické, stochastické a smíšené. Nastínili jsme princip těchto moderních evolučních algoritmů: Stochastic Hill-Climbing, Tabu Search, Ant Colony Optimization, Scatter Search, Immunology System Method, Differential Evolution a SOMA.

Druhá část práce je věnována specificky algoritmu Particle Swarm (rojení částic), který spadá do kategorie smíšených optimalizačních algoritmů. Tento algoritmus je inspirován sociálním chováním zvířat. Vznikl na základě matematického popisu hejna ptáků, publikovaném biologem Frankem Heppnerem v roce 1990. Vychází z předpokladů, že každý jedinec v hejnu, respektive roji, je (zejména při hledání potravy) ovlivňován jak svojí dosavadní zkušeností, tak i „globální zkušeností“ celého hejna. Jde tedy o vzájemné působení vlivu individuálního a sociálního chování na jedince.

V textu jsme podali informace o historických základech a jednotlivých myšlenkových tezích algoritmu Particle Swarm. Popsali jsme dále jeho strukturu a možné modifikace. Hlavním cílem této práce ovšem bylo sestavení algoritmu Particle Swarm v programovém prostředí *Mathematica*, testování jeho funkčnosti na několika vybraných testovacích funkcích a na již řešených problémech. Pro testování funkčnosti algoritmu jsme vybrali tyto funkce: funkce f_6 , první De Jongova funkce, Rosenbrockova funkce (druhá De Jongova funkce), Rastrigrinova funkce, Griewankova funkce, Ackleyova funkce a Schwefelova funkce. Dále jsme pro testování vybrali tyto, již řešené, problémy: Návrh ozubeného kola a Návrh tlakové nádoby (viz [6]). Na těchto sedmi testovacích funkcích a dvou příkladech jsme prokázali plnohodnotnost evolučního algoritmu Particle Swarm. Námi testovaný algoritmus dosahuje poměrně kvalitních výsledků, jediný výraznější problém měl s optimalizací Schwefelovy funkce.

Tato práce nám ukázala, že evoluční algoritmus Particle Swarm je zcela plnohodnotným, kvalitním a v praxi využitelným algoritmem a zasluhuje si do budoucna, pro důkladné zhodnocení svých pozitiv a negativ, vypracovat řádnou a rozsáhlou studii.

SEZNAM POUŽITÉ LITERATURY

- [1] Zelinka, Ivan, *Umělá inteligence I. Volume 1*. Zlín : Vutium, Brno, 1998. 126 p. ISBN 80-214-1163-5.
- [2] Zelinka, Ivan, *Umělá inteligence / kap.6 "Diferenciální evoluce"*, Academia, 33 p.
- [3] Kvasnička, V., Pospíchal, J., Tiňo, P., *Evoluční algoritmy*, STU Bratislava, 2000, ISBN 85-246-2000.
- [4] Zelinka, Ivan, *New Optimization Techniques in Engineering / kap.7 "SOMA - Self Organizing Migrating Algorithm"*, Springer-Verlag.
- [5] Corne, D. W., Dorigo, M., Glover, F., editors, *New Ideas in Optimization - Mechanical Engineering Design Optimization by Differential Evolution. Volume 1*. London : McGraw-Hill, 1999. 20 p. ISBN 007-709506-5.
- [6] Zelinka Ivan, *Umělá inteligence v problémech globální optimalizace*, BEN, 2002, 190 p. ISBN 80-7300-069-5.
- [7] Ken Price, Rainer Storn, Jouni Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, Springer, ISBN 3-540-20950-6.
- [8] Pomeroy, Paul, *An Introduction to Particle Swarm Optimization* [online] March 2003 [cit. březen, 2005] Dostupné na internetu: <<http://adaptiveview.com/articles/ipsop1.html>>
- [9] Particle swarm optimization-based schemes for resource-constrained project scheduling, *Automation in Construction, Volume 14, Issue 3*, June 2005, Pages 393-404, Hong Zhang, Xiaodong Li, Heng Li and Fulai Huang [online] [cit. březen, 2005] Dostupné na internetu: <www.sciencedirect.com>
- [10] Fukuyama, Y., *IEEE PES Tutorial on Modern Heuristic Optimization Techniques with Application to Power Systems* [online] [cit. březen, 2005] Dostupné na internetu: <<http://homepage2.nifty.com/fukuyama-yoshikazu/ECTutorial.htm>>
- [11] Hu, X., *PSO Tutorial* [online] 2001-2004 [cit. březen, 2005] Dostupné na internetu: <<http://swarmintelligence.org/tutorials.php>>
- [12] Babjak, J., *Algoritmus SOMA - nejlepší, čo doma máme* [online] 05.03.2003 [cit. březen, 2005] Dostupné na internetu: <<http://www.scienceworld.cz/>>

- [13] Zelinka, Ivan, *Test functions* [online] [cit. březem, 2005] Dostupné na internetu: <http://www.ft.utb.cz/people/zelinka/soma/func.html>
- [14] *GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab* [online] 2004 [cit. březem, 2005] Dostupné na internetu: <http://www.geatbx.com/docu/fcnindex.html>
- [15] Dolan, Ariel, *The GA Playground* [online] [cit. březem, 2005] Dostupné na internetu: <http://www.aridolan.com/ga/gaa/gaa.html>
- [16] <http://www.particleswarm.info>
- [17] Hu, X., *Swarm Intelligence* by James Kennedy, Russell C. Eberhart, with Yuhui Shi [online] 2002-11-07 [cit. březem, 2005] Dostupné na internetu: <http://www.swarmintelligence.org/SIBook/SI.php#Download>
- [18] Kennedy, J., Eberhart, R. C., Shi, Y., *Swarm Intelligence*, March 2001, 400 pages, Morgan Kaufmann Publishers, ISBN 1-55860-595-9
- [19] Bonabeau, E., Dorigo, M., Theraulaz, G., *Swarm Intelligence*, Oxford University Press, 1999, ISBN 0-19-513159-2
- [20] Dawkins, R., *The Selfish Gene*, Oxford University Press, 1976; 2nd edition, December 1989, hardcover, 352 pages, ISBN 0192177737; April 1992, ISBN 019857519X; trade paperback, September, 1990, 352 pages, ISBN 0192860925

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

HUF hodnota účelové funkce

PSO Particle Swarm Optimization (optimalizace „rojení částic“)

SOMA Samo-organizující se migrační algoritmus

UF účelová funkce

SEZNAM OBRÁZKŮ

Obr. 1 Princip horolezeckého algoritmu	12
Obr. 2 Algoritmus Ant Colony Optimization	13
Obr. 3 Migrace jedince v algoritmu SOMA	15
Obr. 4 Graf funkce f_6 , ve 2D graf funkce $f_6(x_1, x_2=0)$	24
Obr. 5 Graf globálního nejlepšího řešení PSO na funkci f_6	24
Obr. 6 Graf funkce Sphere	25
Obr. 7 Graf globálního nejlepšího řešení PSO na funkci Sphere.....	25
Obr. 8 Graf Rosenbrockovy funkce, 2D graf zobrazuje řez 3D grafem v $x_2 = 1$	26
Obr. 9 Graf globálního nejlepšího řešení PSO na Rosenbrockově funkci.....	27
Obr. 10 Graf Rastrigrinovy funkce	28
Obr. 11 Graf globálního nejlepšího řešení PSO na Rastrigrinově funkci	28
Obr. 12 Graf Griewankovy funkce.....	29
Obr. 13 Graf globálního nejlepšího řešení PSO na Griewankově funkci	29
Obr. 14 Graf Ackleyovy funkce.....	30
Obr. 15 Graf globálního nejlepšího řešení PSO na Ackleyově funkci.....	31
Obr. 16 Graf Schwefelovy funkce	32
Obr. 17 Graf globálního nejlepšího řešení PSO na Schwefelově funkci	32
Obr. 18 Ozubený převod z příkladu 1	34
Obr. 19 Tlaková nádoba z příkladu 2.....	36

SEZNAM TABULEK

Tab. 1 Uspořádání optimalizačních algoritmů dle <i>Zelinky (2002)</i>	9
Tab. 2 Výsledky testu PSO na funkci f_6	24
Tab. 3 Výsledky testu PSO na funkci Sphere	26
Tab. 4 Výsledky testu PSO na Rosenbrockově funkci.....	27
Tab. 5 Výsledky testu PSO na Rastrigrinově funkci.....	28
Tab. 6 Výsledky testu PSO na Griewankově funkci	30
Tab. 7 Výsledky testu PSO na Ackleyově funkci	31
Tab. 8 Výsledky testu PSO na Schwefelově funkci.....	32
Tab. 9 Alternativní metody použité k řešení testovacích problémů.....	33
Tab. 10 Optimální řešení získané jinými algoritmy než PSO	35
Tab. 11 Optimální řešení tlakové nádoby, případ A	37