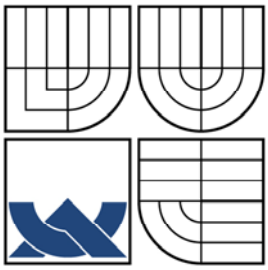
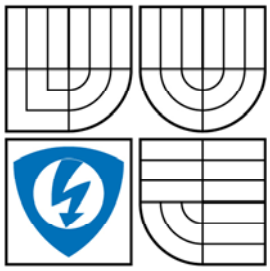


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A
KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

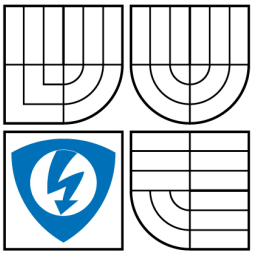
EVOLUČNÍ ALGORITMY
EVOLUTIONARY ALGORITHMS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DANIEL HAUPT
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR HONZÍK, PhD.
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Daniel Haupt	***ID:***	73032
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Evoluční algoritmy

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s následujícími optimalizačními metodami: genetický algoritmus, diferenciální evoluce. Metody naprogramujte a na zvolených příkladech proveďte jejich srovnání.

DOPORUČENÁ LITERATURA:

Dle vlastního literárního průzkumu a doporučení vedoucího práce.

Termín zadání:	9.2.2009	***Termín odevzdání:***	1.6.2009

Vedoucí práce: Ing. Petr Honzík, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

Anotace

První část práce je teoretická a zabývá se optimalizací a evolučními algoritmy, které jsou používány k řešení složitých optimalizačních problémů. Konkrétně jsou popsány algoritmy *diferenciální evoluce, genetický algoritmus, simulované žíhání* a deterministický neevoluční algoritmus *zakázané prohledávání*. Dále je diskutována problematika testování optimalizačních algoritmů pomocí tzv. galerie testovacích funkcí a testování pomocí srovnání výsledků algoritmů při řešení problému obchodního cestujícího.

Ve druhé části práce jsou všechny uvedené algoritmy testovány na 11 testovacích funkcích a na třech modelech rozmístění měst v problému obchodního cestujícího. Nejprve jsou algoritmy srovnávány s možností neomezeného přístupu k účelové funkci a dále s omezenou možností přístupu k účelové funkci. Veškerá data jsou statisticky a graficky zpracována. Jednotlivé algoritmy jsou seřazeny dle úspěšnosti.

Klíčová slova

diferenciální evoluce, genetický algoritmus, simulované žíhání, zakázané prohledávání, optimalizace, optimalizační algoritmy, evoluční algoritmy, galerie testovacích funkcí, problém obchodního cestujícího, účelová funkce, fitness funkce, srovnání algoritmů, populace, mutace, křížení, evoluce, generace, operátor křížení s rekombinací hran, relativní indexování pozicí

Annotation

The first part of this work deals with the optimization and evolutionary algorithms which are used as a tool to solve complex optimization problems. The discussed algorithms are *Differential Evolution*, *Genetic Algorithm*, *Simulated Annealing* and deterministic non-evolutionary algorithm *Taboo Search.* Consequently the discussion is held on the issue of testing the optimization algorithms through the use of the test function gallery. All algorithms are compared on the Travelling Salesman Problem as well.

In the second part of this work all above mentioned optimization algorithms are tested on 11 test functions and on three models of placement cities in Travelling Salesman Problem. Firstly, the experiments are carried out with unlimited number of accesses to the fitness function and secondly with limited number of accesses to the fitness function. All the data are processed statistically and graphically.

Keywords

Differential Evolution, Genetic Algorithm, Simulated Annealing, Taboo Search, optimization, optimization algorithms, evolutionary algorithms, test function gallery, fitness function, comparison of algorithms, population, mutation, crossing, evolution, generation, TSP, ERX, Travelling salesman problem, edge recombination crossover, relative position indexing

Bibliografická citace

HAUPT, D. *Evoluční algoritmy.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 69 s. Vedoucí bakalářské práce Ing. Petr Honzík, PhD.

P r o h l á š e n í

„Prohlašuji, že svou bakalářskou práci na téma "Evoluční algoritmy" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o ď ě k o v á n í

Děkuji tímto Ing. Petru Honzíkovi, PhD. za cenné připomínky a rady při vypracování semestrální práce.

V Brně dne : Podpis:

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

Zkratka/Symbol	Popis
N, Dim	Počet dimenzí
f_{cost}	Účelová funkce
A, B, C	Rodiče
P	Potomek
rand	Náhodné číslo z intervalu (0;1)
xi	původní hodnota genu před mutací
xi*	Nová hodnota zmutovaného genu
XMAX	Maximální možná hodnota proměnné *xi*
XMIN	Minimální možná hodnota proměnné *xi*
T	Maximální počet generací
B	Parametr nelinearity mutace
P	Pravděpodobnost přijetí nového řešení
GA	Genetický algoritmus
SA	Simulované žíhání
TS	Zakázané prohledávání
DE	Diferenciální evoluce
NP	Velikost populace
D	- dimenze
NP	- velikost populace
CR	- práh křížení (DE)
F	- mutační konstanta (DE)
GENERACE	- maximální počet generací pro ukončení běhu programu (DE a GA)
psi	- přesnost řešení pro rozdíl mezi dvěma po sobě jdoucími optimy (GA,DE,TS)
pocStejRes	- počet stejných po sobě jdoucích řešení pro ukončení programu (GA,DE,TS)
KK	- konstanta křížení (GA)

MK - mutační konstanta (GA)

B - parametr nelinearity dynamicke Michalewiczovy mutace (GA)

T - počáteční teplota žíhání (SA)

Tmin - konečná teplota žíhání pro zastavení běhu programu (SA)

alpha - koeficient snižování teploty po ukončení počtu transformací (SA)

kmax - maximální počet změn při určité teplotě (SA)

tmax - maximální počet transformací při určité teplotě (SA)

koefOkoli - udává velikost kroku při transformaci vzhledem k rozsahu definičního oboru (TS)

Tmax - maximální počet pohybů na prohledávané hyperploše vždy směrem k lepšímu řešení (TS)

kmax - velikost taboo listu (TS)

pocTrans - počet transformací v okolí jednoho bodu řešení (TS)

KGNJ - konstanta generace nahodneho jedince udava pst s jakou se uskutecni mutace pomoci generace nahodneho jedince a

1-KGNJ udava s jakou pravdepodobnosti dojde k mutaci typu vymena mest

KTS - konstanta typu selekce udava pst s jakou se v selekci bude vybirat z rodicu

a 1-KTS udava s jakou psti se bude vybirat z potomku

typ - udava který typ generovaní městu bude použitý

typ=0 nahodne rozmisteni mest vzhledem k osam x,y s omezenim

typ=1 rozdeleni mest do kruhu

typ=2 rozdeleni mest do tangenty pro rovnomerne x

typ=3 rozdeleni mest do tangenty pro nahodne x

omezeniXY - udava interval na ose X a Y v nemz se budou generovat mesta

D - počet měst

GTF - galerie testovacích funkcí

TSP - problém obchodního cestujícího

1. ÚVOD

V každodenním životě se setkáváme s řadou problémů, které jsme nuceni řešit. Naše řešení by mělo být dostatečně efektivní, abychom neztráceli zbytečně čas, energii, peníze atd. Příkladem by mohla být doprava autem do práce skrz ucpané město. Tato situace je závislá na mnoha faktorech (parametrech), které ovlivní celkový výsledek Musíme se spolehnout na vlastní inteligenci, zkušenost a intuici. Z toho plyne otázka zda jsme schopni využít moderní informační technologie k tomu, aby naše řešení bylo dle vybraného kritéria optimální.

V dnešní době výpočetní techniky se touto otázkou zabývá stále více lidí, ale pořád nejsme schopni řešit spoustu problémů, protože složitost v závislosti na počtu parametrů roste velmi rychle. Příklad úlohy o obchodním cestujícím nám to dokáže. Vybrat nejkratší cestu mezi např. 50 městy znamená vyčíslit $3{,}064*10^{64}$ různých kombinací cest a z nich vybrat tu nejkratší. I s dnešní rychlosti procesorů se při prohledání všech možných cest dostaneme k více než miliardám let výpočtů.

Člověk využívá mozek což znamená, že každé řešení je subjektivní. Subjektivita není v technických vědách oblíbená a vždy se snažíme jí vyhýbat a vše mít podložené exaktním řešením. Existují však případy, a není jich málo, kdy analytické řešení není možné. V těchto případech jsme nuceni hledat inspiraci např. v přírodě, v chování a vývoji zvířat, či v pochopení činnosti lidského mozku.

Skloubením možností počítačů (i když jsme ukázali, že jsou stále omezené), náhody a inspirace v netechnických vědách dostáváme nástroje pro hledání optima. Nejsou analyticky ověřitelné, ani nevedou vždy ke stejnému výsledku, ale fungují a většinou velmi dobře.

Cílem této práce je porovnat kvalitu optimalizačních algoritmů diferenciální evoluce a genetického algoritmu, jež se řadí mezi evoluční algoritmy. Pro pomocné srovnání jsou využity výsledky výpočtů algoritmů simulovaného žíhání, který je taktéž evolučního charakteru a algoritmu zakázaného prohledávání.

Výše zmíněné algoritmy jsou testovány na tzv. galerii testovacích funkcí, která obsahuje 11 vybraných testovacích funkcí. Testování je taktéž provedeno na problému obchodního cestujícího pro tři modely uspořádání 30 měst (rovnoměrně do

kruhu, rovnoměrně do tangenty, náhodně). Kvůli zjištění vlastností optimalizačních algoritmů a kvůli navržené modelové situaci (kap. 8.3.1) jsou algoritmy testovány jak pro neomezený počet přístupů k účelové funkci tak pro omezený počet přístupů k účelové funkci.

2. OPTIMALIZACE A OPTIMALIZAČNÍ ALGORITMY

2.1 OPTIMALIZACE

Optimalizací rozumíme proces hledání minimální, či maximální hodnoty dané účelové funkce.

2.1.1 Účelová funkce

Pod názvem účelová funkce (fitness function, cost function – anglické názvy) si lze představit jakoukoli funkci jejíž globální extrém je předmětem optimalizace a odpovídá řešenému problému. Jinými slovy je to taková funkce, která při určitých hodnotách parametrů poskytne svou funkční hodnotu k vyhodnocení, neboli udává kvalitu řešení optimalizace. Pomocí změny parametrů je hledána taková hodnota účelové funkce, která bude z daného hlediska optimální (většinou minimum, či maximum). Například úloha obchodního cestujícího má účelovou funkci takovou, jež udává vzdálenost ujetou mezi městy dle daného seřazení.

Na účelovou funkci lze také nahlížet jako na vyhodnocení geometrického problému. Výsledkem hledání hodnoty účelové funkce je její funkční hodnota na n-rozměrné hyperploše (2.1). Je-li n rovno osmi je hledáno optimum na osmi rozměrné ploše v devíti rozměrném prostoru. Pokud je použita analogie z dvourozměrného prostoru je návratová hodnota účelové funkce (2.2) výška (funkční hodnota) dvourozměrné plochy v zadaném bodě.

$$f_{cost} = g(x_1, x_2, ..., x_n) \tag{2.1}$$

$$z = f(x, y) \tag{2.2}$$

2.2 TYPY OPTIMALIZAČNÍCH ALGORITMŮ

Optimalizačních algoritmů lze najít velké množství [2], taktéž existují různá dělení. V této podkapitole je zmíněno pouze základní dělení.

2.2.1 Enumerativní

V tomto případě algoritmy vypočítávají všechny možné kombinace parametrů a následně naleznou optimum. Je to jediný typ algoritmu u kterého je jistota, že lepší řešení neexistuje (pokud zanedbáme možnou chybu při převodu iracionálních čísel do binární podoby). Velkou nevýhodou je obrovská časová náročnost výpočtu i za předpokladu použití nejmodernějších superpočítačů. Enumerativní algoritmy jsou použitelné pouze v případě malého počtu parametrů s velmi malou variabilitou, nejlépe diskrétního charakteru.

2.2.2 Deterministické

Skupina těchto algoritmů je založená na matematické přesnosti. Na stejný vstup reagují vždy stejným výstupem, nalézají vždy tatáž řešení. Pro jejich efektivní aplikaci musí řešený problém splnit řadu předpokladů [1].

2.2.3 Stochastické

Stochastické algoritmy využívají ke své funkčnosti náhodu. Jednotlivé hodnoty parametrů jsou pomocí jistého generátoru náhodných čísel doplněny a nakonec je vybráno nejlepší z náhodně prohledaných řešení. Skupina těchto algoritmů je využívána pouze k hrubému odhadu.

2.2.4 Smíšené

Tyto algoritmy jsou směsicí deterministických a stochastických algoritmů. Využívají náhody a parametry jsou podle předem daných pravidel upravovány. Mezi smíšené algoritmy patří všechny testované algoritmy v této publikaci. Algoritmus zakázaného prohledávání je deterministický, ale při transformaci je použita náhoda pro generování směru ohodnocených transformací.

2.3 OŠETŘENÍ PŘEKROČENÍ OMEZUJÍCÍCH PODMÍNEK ÚČELOVÉ FUNKCE

Účelová funkce může mít jisté požadavky na omezení, které plynou buď z fyzikální podstaty (záporná délka atd.) nebo z omezení požadovaného zadavatelem (ekonomické důvody, časová omezení atd.). Existuje několik způsobů jak nakládat s parametry, aby dodržely omezující podmínky. Používají se transformace a techniky, které znemožní parametrům nabytí zakázaných hodnot nebo, které převedou parametry ze zakázané oblasti zpět do povolených intervalů [1], [3].

Pokud jsou velmi dobře známy vlastnosti účelové funkce lze navrhnout takovou sadu transformací, aby nedošlo k překročení daného definičního oboru. Bohužel takových případů je v praxi minimum. Navíc tato metoda klade velký důraz na logiku, inteligenci a zkušenost programátora.

Často používaná je další metoda založená na pokutování funkcí, které překročily daný definiční obor. Překročení musí být rozpoznáno a velikost účelové funkce je pak degradována. Degradace pomocí pokutové funkce může být málo efektivní a výsledky se mohou dostat do celkové statistiky řešení. Je proto nanejvýš žádoucí pokutovou funkci volit s rozmyslem.

Způsob hodně univerzální je založen na opravných algoritmech. Po rozpoznání překročení použijeme takovou transformaci, která zajistí návrat hodnoty do mezí přípustných řešení. Používá se výměna překračujících parametrů náhodným číslem z intervalu přípustných hodnot nebo se parametry vymění za hraniční hodnoty tohoto intervalu.

3. POPIS POUŽITÝCH OPTIMALIZAČNÍCH ALGORITMŮ

Použité optimalizační algoritmy jsou evolučního charakteru. V čase se vyvíjejí a deterministické transformace jsou ovlivněny náhodou. I deterministický algoritmus taboo search je v této práci ovlivněn náhodnými čísly. Oproti ostatním je použití náhody značně minimalizováno. Používá se pouze u výběru směrů transformace. Je to učiněno z důvodu složitosti určení přesných transformací pro velký počet parametrů, tak aby transformace pokryla velkou část prohledávaného prostoru. Základními algoritmy v této práci jsou diferenciální evoluce a genetický algoritmus. Jejich implementace se liší i když mají společný základ evolučního charakteru.

3.1 GENETICKÝ ALGORITMUS

Genetický algoritmus se snaží napodobovat přírodní evoluční výběr a je inspirován Darwinovou evoluční teorií. Slovo genetický poukazuje na implementaci, kde jsou parametry kódovány pomocí binárních čísel v tzv. chromozómu. Pro kódování jsou v této práci použita reálná čísla. Je takto učiněno z důvodu větší jednoduchosti a přehlednosti. Tato skutečnost má vliv na způsoby křížení a mutace, jejich větší variabilitu, nicméně je dokázáno, že binární i reálné kódování přinášejí téměř stejné výsledky [3].

3.1.1 Základní pojmy

Podrobný slovník základních pojmů najdete v [4], zde je zmíněno pouze několik pojmů týkajících se reálného kódování a implementace.

NP – velikost populace
Populace – množina jedinců
Jedinec – soubor parametrů prohledávaného prostoru a jejich ohodnocení
Dimenze – počet parametrů prohledávaného prostoru

Konstanta křížení – pravděpodobnost křížení dvou vybraných jedinců

Mutační konstanta – pravděpodobnost mutace jednoho parametru jedince

Parametr nelinearity mutace – číslo určující typ Michalewiczovy dynamické mutace, používaná hodnota *B=5* [3]

Maximální počet generací – maximální počet generací jdoucí po sobě, parametr slouží k zastavení běhu programu a pro velikost dynamické mutace

Elitismus – princip uchování nejlepšího jedince, nepodrobuje se křížení ani mutaci, pokud je v nové populaci lepší jedinec než Elitní je Elitní vyměněn

3.1.2 Princip genetického algoritmu

Genetický algoritmus se skládá z několika úseků. Můžeme je logicky seřadit takto :

1. *Tvorba počáteční populace* – vytvoření populace s náhodnými parametry a jejich ohodnocení
2. *Selekce* – vybranou metodou selekce se odstraní nebo redukují málo kvalitní jedinci
3. *Křížení* – vybraní jedinci se podrobí křížení z něhož vyjdou dva nebo tři noví jedinci tzv. potomci
4. *Mutace* – každý parametr každého jedince v populaci má šanci být zmutován, pokud je náhodné číslo v intervalu mutace, mutací se změní daný parametr dle vybrané metody
5. *Zkouška a ošetření omezení* – zjistí se zda nedošlo k překročení definičního oboru, či daného předpisu, pokud ano následuje oprava
6. *Vyhodnocení potomků* – zjištění kvality nových jedinců pomocí účelové fukce, zjištění zda stávající Elitní jedinec je stále nejlepší
7. *Kontrola ukončovacích podmínek* - zvýšení počtu generace o 1, kroky 2-6 se znovu opakují pokud ukončovací podmínka není platná, pokud ano dochází k zastavení programu

3.1.3 Kódování pomocí reálných čísel

Princip genetických algoritmů je založen na kódování pomocí binárních čísel jako paralela s genetickým kódem. Chromozom (sekvence symbolů) se skládal z jednotlivých genů (bitů). Kvůli větší přehlednosti, názornosti a jednodušší práci je stále častěji používáno kódování pomocí reálných čísel. Toto kódování dává větší možnost v kreativitě křížení, mutace atd.

Křížení lze vidět jako geometrický problém hledání např. zlatého řezu mezi jednotlivými parametry rodičů, nebo čistě náhodný způsob smíchání jednotlivých parametrů. K mutaci je třeba přistoupit obezřetněji. Mutační konstanta pro reálná čísla se tvoří pomocí výpočtů z pravděpodobnosti mutace jednoho genu v chromozomu [4] .

Rozdíly mezi binárním a reálným kódováním testovalo již několik nezávislých vědců a nikdo nepřišel s výsledky, které by byly vysoce odlišné viz.[3][5]

3.1.4 Křížení

Křížení u kódování pomocí reálných čísel nemá stejnou podobu jako u kódování pomocí binárních čísel, protože je to neefektivní oproti jiným způsobům křížení. Naproti tomu se používá řada křížících postupů, které se dají najít ve velkém množství literatury např. [4].

Křížící postup z rovnic (3.1, 3.2, 3.3) vytvoří namísto klasických dvou potomků tři. Všichni jsou ohodnoceni a do nové populace postupují pouze dva nejlepší.

$$\vec{P}_1 = \frac{\vec{A} + \vec{B}}{2} \tag{3.1}$$

$$\vec{P}_2 = \frac{3\vec{A} - \vec{B}}{2} \tag{3.2}$$

$$\vec{P}_3 = \frac{-\vec{A} + 3\vec{B}}{2} \tag{3.3}$$

Křížení se uskuteční pouze při splnění podmínky pro křížení. Pokud bude náhodné číslo vygenerované počítačem v intervalu, jež určuje *konstanta křížení*. Rodiče jsou taktéž náhodně vybírání z celé populace, aby bylo dosaženo větší variability.

3.1.5 Mutace

Mutací se rozumí náhodná změna náhodně vybraného parametru vybraného jedince. Při procesu mutace může být k parametru přičtena či odečtena pevně daná hodnota, nebo náhodné číslo z předem daného intervalu. To vyžaduje znalost rozsahů hodnot v nichž se budeme pohybovat. Proto se používá tzv. Michalewiczova dynamická mutace. Velikost přičítané nebo odečítané hodnoty se s časem mění podle předpisu (3.5). Z toho lze usoudit, že zpočátku je mutace velká a dopomáhá k prohledání velkého prostoru. Jakmile počet generací roste, vliv mutace je menší a dopomáhá k prohledání malé oblasti parametru.

$$x_i^* = \begin{cases} x_i + \Delta(t, X_{MAX} - x_i) \\ nebo \\ x_i + \Delta(t, x_i - X_{MIN}) \end{cases} \tag{3.4}$$

$$\Delta(t, y) = y\left(1 - r^{\left(1-\frac{t}{T}\right)^B}\right) \tag{3.5}$$

3.1.6 Selekce

Selekcí se rozumí způsob výběru kvalitních jedinců. Méně kvalitní jedinci jsou po selekci redukováni. Zato kvalitní jedinci mohou být zastoupeni i ve více kopiích.

Selekcí je několik druhů. Ale ne všechny jsou vhodné pro všechny typy řešených problémů [3]. Jedním typem je *vážená ruleta.* Každému jedinci je přidělena pravděpodobnost výběru podle jeho kvality podělená součtem kvalit všech jedinců. Potom jsou generována náhodná čísla a dle pravděpodobnosti jsou vybíráni jedinci, kteří projdou selekcí. Z toho vyplývá, že lépe hodnocení jedinci mají větší šanci

projít selekcí. Tato metoda není příliš vhodná kvůli tendenci zůstávat v lokálním extrému. Dalším problémem je ošetření záporných čísel [3] .

Problém vážené rulety upadat do lokálních extrémů řeší metoda *poziční selekce*. Principiálně je stejná jako vážená ruleta, ale namísto výpočtu pravděpodobnosti z kvality jedince je použito seřazení jedinců dle kvality (sestupně). Pravděpodobnost se potom počítá jako pořadí děleno součtem všech pořadí.

Tyto metody jsou implementačně složitější než další z používaných metod, kterou je *turnaj*. Dva náhodně vybraní jedinci jdou do turnaje a jako vítěz je brán ten s lepším ohodnocením. Ten také postupuje selekcí dále.

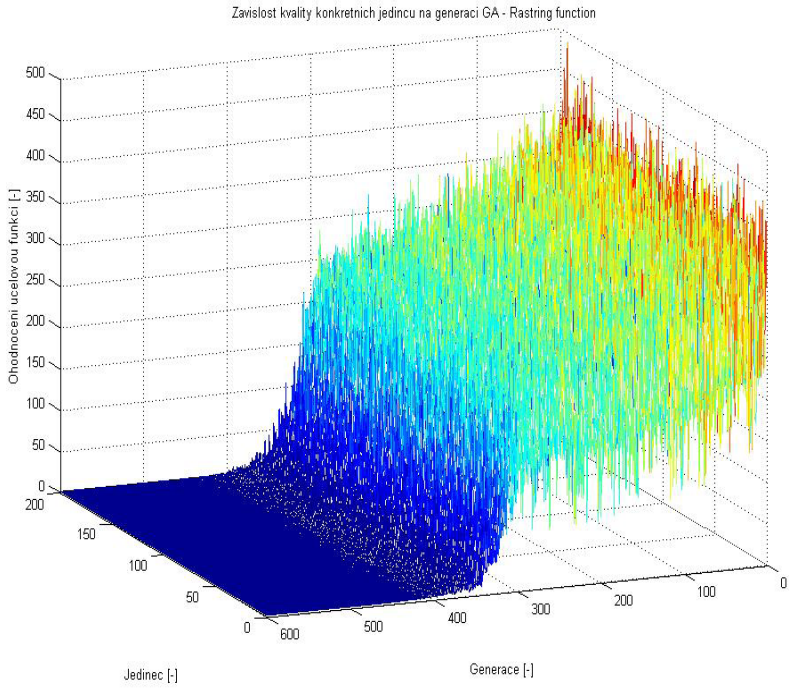
Aby se předcházelo ztrátám nejlepšího zatím nalezeného řešení křížením, či mutací používá se metoda Elitismus. To znamená, že nejlepší doposud nalezený jedinec je před selekcí, křížením a mutací uložen a po těchto operacích vyměněn za nejhoršího jedince v populaci.

3.1.7 Náhradová strategie

Možností jak vybírat kvalitní jedince se zabývá selekce, ale možností z čeho a kdy vybírat jedince pro selekci se zabývá náhradová strategie. Postupů je mnoho, jedince pro selekci lze vybírat jen z nové populace. Také se používá strategie při níž se populace prokříží a zmutuje a jedinci pro selekci se vyberou jak z původní populace (rodičů) tak z populace potomků. Selekcí v generačním kole může být i více, dají se umístit po jednotlivých úsecích např. po křížení atd.

3.1.8 Vliv maximálního počtu generací na dynamickou Michalewiczovu mutaci

Jak bylo uvedeno v kapitole 3.1.5 Michalewiczova mutace je vhodným řešením. Nicméně skrývá úskalí v podobě znalosti alespoň orientačního maximálního počtu generací, tedy v rovnici (3.5) parametru T. Pokud bude T příliš velké oproti tomu, kdy dochází k zastavení běhu programu pomocí jiných ukončovacích podmínek, nemusí dojít k prohledání blízkého okolí parametrů a úzký globální extrém může zůstat neodhalen.



3-1 Závislost kvality 200 jednotlivých jedinců na počtu generací

Jak můžeme vidět na obr. 3.1 jsou změny způsobené mutací a křížením zpočátku velké a hodně rozdílné. Zhruba od 300 generace dochází k mírné stabilizaci, zmenšení vlivů mutace (prohledávání menších okolí) a celá populace směřuje ke globálnímu extrému funkce Rastring jímž je 0. Pokud jiná zastavovací podmínka ukončí běh programu např. ve 250 generaci nedojde k prohledání malých okolí a z větší pravděpodobností nebude nalezen globální extrém.

3.2 DIFERENCIÁLNÍ EVOLUCE

Diferenciální evoluce je mladý optimalizační algoritmus (1995). Je evolučního charakteru a má podobné rysy jako genetický algoritmus. Oproti genetickému algoritmu je jednodušší v implementaci. Lze jej naprogramovat i v tabulkových procesorech. Noví jedinci se tvoří ze 4 rodičů a ne ze 2 jako v genetickém algoritmu. [6]

3.2.1 Základní pojmy

Diferenciální evoluce má několik stěžejních parametrů, jejich vzájemné provázanosti jsou diskutovány a testovány v [1], kde taktéž naleznete doporučení pro jejich nastavování. Zde je zmíněn jen jejich základ.

CR – práh křížení, pravděpodobnost s jakou bude přijat parametr ze šumového vektoru

F – mutační konstanta

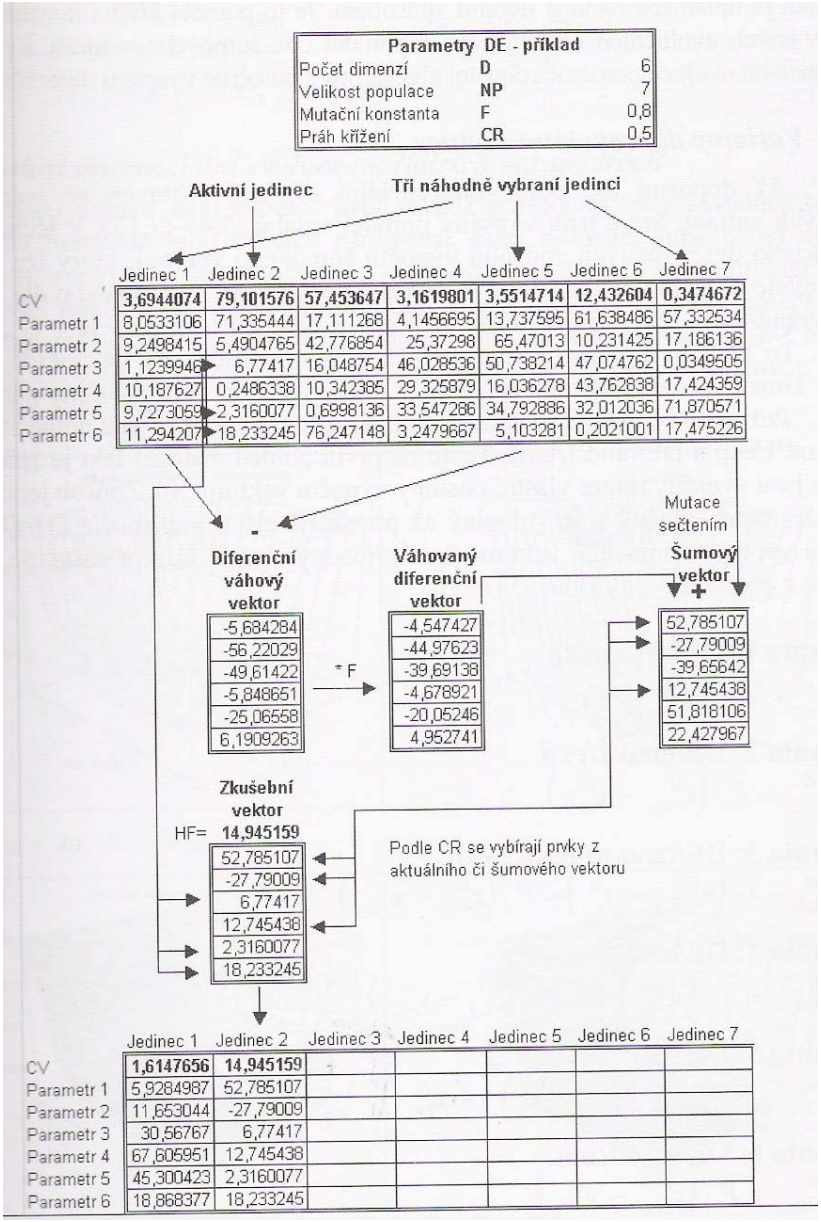
D – dimenze, počet parametrů prohledávaného prostoru

NP – velikost populace

3.2.2 Princip diferenciální evoluce

Diferenciální evoluce je velmi jednoduše implementovatelný algoritmus jak lze vidět z obrázku obr. 3.2.

1. Na počátku jsou vygenerováni jedinci s náhodnými parametry a ohodnoceni.
2. Vybere se první (aktivní) jedinec z populace a náhodně další tři.
3. Parametry prvních dvou náhodně vybraných jedinců se od sebe odečtou a tím vznikne diferenční váhový vektor.
4. Všechny parametry se vynásobí mutační konstantou F, tak vznikne váhový diferenční vektor
5. K parametrům třetího náhodně vybraného jedince se přičte váhový diferenční vektor a vznikne vektor šumový.
6. Následuje tvorba zkušebního vektoru. Pro každý parametr je generováno náhodné číslo, pokud je náhodné číslo menší než CR uloží se do daného parametru stejný parametr z vektoru šumového. Pokud je náhodné číslo větší než CR uloží se do daného parametru stejný parametr z aktivního jedince.
7. Zkušební vektor se ohodnotí a porovná se s aktivním jedincem. Do další generace pokračuje vektor s lepším ohodnocením ze zmíněných dvou.
8. Vybere se druhý (aktivní) jedinec a náhodně další tři a kroky 3-7 se opakují pro celou populaci. Takto se pokračuje v tvorbě nových generací až do zastavení programu.



3-2 Princip diferenciální evoluce [1]

3.2.3 Varianty diferenciální evoluce

Způsob tvorby šumového vektoru není v diferenciální evoluci striktně dán. Jako příklad byla použita varianta 7 (DE/rand/1/bin) [1], která je hodnocena jako nejkvalitnější [1].

3.2.4 Stagnace

Je nežádoucí jev provázející diferenciální evoluci. Při stagnaci dochází k zastavení vývoje hodnoty účelové funkce směrem k optimu ještě před jeho nalezením. Je to způsobeno malou populací a neschopností zkušebních vektorů probojovat se do nové populace, tudíž operace reprodukce je schopna vytvořit jen malý počet zkušebních řešení [1].

3.3 SIMULOVANÉ ŽÍHÁNÍ

Simulované žíhání je optimalizační algoritmus inspirovaný žíháním oceli. Dalo by se říci, že s postupem času jsou změny čím dále menší a tolerance horších řešení taktéž klesá (analogie s klesající teplotou při žíhání). Tímto způsobem se může algoritmus jednoduše dostat z lokálního extrému [7].

3.3.1 Základní pojmy

T – teplota, s časem se snižuje a ovlivňuje další parametry

Tmin – minimální teplota, pokud T bude menší než Tmin zastaví se program

alpha – koeficient zmenšování teploty

Metropolisovo kritérium - určuje pravděpodobnost nahrazení starého řešení novým

$$P(x \rightarrow x') = \begin{cases} 1 & \text{pro } f(x') \leq f(x) \\ e^{\frac{f(x')-f(x)}{T}} & \text{pro } f(x') > f(x) \end{cases} \quad (3.6)$$

Pokud je nové řešení lepší než předešlé je jeho pravděpodobnost přijetí 1. Ve druhém případě se počítá dle druhého řádku vztahu (3.6), kde f(x) je hodnota funkce pro dané parametry **x** (původní) a f(x´) je hodnota funkce pro nové parametry **x´.**

3.3.2 Princip simulovaného žíhání

1. Je vygenerován vektor s náhodnými parametry a ohodnocen

2. Vektor je uchován a projde transformací
3. Původní a přetransformovaný vektor jsou porovnány a je spočítána pravděpodobnost přijetí horšího řešení podle (3.6)
4. Náhodně generované číslo určí, který z vektorů bude pokračovat jako základní
5. Body 2-4 se stále opakují než je překročen maximální počet transformací při dané teplotě nebo maximální počet akceptovaných horších řešení
6. Pomocí koeficientu *alpha* se zmenší teplota (podle Metropolisova kritéria klesne pravděpodobnost akceptace horšího řešení)
7. Nejlepší nalezená hodnota se ukládá a po zmenšení teploty je ona výchozím bodem pro nové transformace
8. Body 2-7 se opakují tak dlouho, než je dosažena minimální teplota nebo ukončí běh programu jiná ukončovací podmínka.

3.3.3 Simulované žíhání s délkou kroku transformace závislou na teplotě

Transformace u simulovaného žíhání mohou být různorodé. Nežádoucí je jen překročení definičního oboru. Transformace může být navržena tak, aby jej nepřekročila nebo použijeme některou z oprav zmíněných v podkapitole 2.3.

Transformaci můžeme vytvořit i dynamickou (podobnost s Michalewiczovou dynamickou mutací u genetických algoritmů). Délka nebo krok možné transformace lze ovlivňovat teplotou. S klesající teplotou je dobré snižovat i maximální délku transformace, čímž způsobíme prohledávání menších okolí ke konci běhu programu. Tato skutečnost znamená velké transformační změny na počátku a malé na konci běhu programu.

3.4 ZAKÁZANÉ PROHLEDÁVÁNÍ

Zakázané prohledávání nebo-li taboo (tabu) search spadá do kategorie deterministických gradientních algoritmů. Hledá si cestu největšího spádu a hlídá si, aby v několika krocích za sebou nedoputoval znovu do stejného místa. K tomu využívá krátkodobou paměť do níž ukládá inverzní transformace k těm předešlým.

Tímto způsobem předchází nebo spíše eliminuje možnost uváznutí v lokálním extrému.

Existuje velké množství inovací tohoto algoritmu jak je zmíněno v [7].

3.4.1 Základní pojmy

Zakázaný seznam – seznam inverzních transformací k již použitým transformacím s počtem prvků rovným *k*, seznam musí být kratší než maximální možný počet transformací. Můžeme jej taktéž chápat jako zásobník typu FIFO.

Koeficient okolí – určuje délku kroku transformace, může být taktéž měněn podle dosažených výsledků [7]

Počet transformací – počet transformací při hledání lokálního minima, počet hodnot ze kterých se určí lokální minimum

Maximální počet kroků – po prohledávané hyperploše, zastavující podmínka

3.4.2 Princip zakázaného prohledávání

1. Je vygenerován vektor s náhodnými parametry a ohodnocen
2. Tento vektor je brán jako střed okolí na němž se bude hledat lokální optimum, okolí je definováno *koeficientem okolí*
3. Provede se daný počet transformací na daném okolí a určí se lokální optimum
4. Transformace pro nalezené lokální optimum se porovná s transformacemi v zakázaném seznamu. Pokud není v seznamu obsaženo jsou parametry lokálního optima brány jako střed pro další transformace. Pokud je transformace obsažena v zakázaném seznamu vybere se ze seznamu transformací lokální optimum, které není obsaženo v zakázaném seznamu.
5. Transformace vybraného lokálního optima je zapsána do zakázaného seznamu na nezaplněnou pozici, nebo na pozici nejstarší zapsané transformace
6. Pokud je vybrané lokální optimum lépe ohodnoceno než prozatímní globální optimum je považováno za prozatímní globální optimum

7. Kroky 3-6 se opakují tak dlouho než je dosažen maximální počet kroků po prohledávané hyperploše nebo program nezastaví jiná zastavovací podmínka

Transformace u zakázaného prohledávání je deterministická. Jsou předem dány postupy výpočtu okolí. Tato skutečnost je tím více složitější čím je větší počet parametrů. Vytvořit takovou sadu transformací např. pro 50 parametrů tak, aby bylo co nejefektivněji pokryto prohledávané okolí (nejjednodušší okolí je 50 rozměrná koule), je velmi složitou záležitostí s mnoha výpočty. Z toho důvodu je v implementaci použitého programu použita náhoda k určení transformací, avšak s danou délkou transformace. Tato skutečnost posouvá deterministický algoritmus mezi algoritmy smíšené.

Stěžejním parametrem v algoritmu zakázaného prohledávání je koeficient okolí. Správné nastavení jeho velikosti může být rozhodujícím faktorem. Pokud bude velký může přeskakovat lokální optima (což je na první pohled žádoucí), ale stejně může přeskočit globální optimum. Naopak pokud bude malý, je rozhodující délka zakázaného seznamu. Algoritmus se může zacyklit v lokálně optimálním „údolí“ a pokud bude malý seznam zakázaných transformací už se z něj nedostane a globální optimum taktéž nebude nalezeno.

4. GALERIE TESTOVACÍCH FUNKCÍ

Pod názvem galerie testovacích funkcí si lze představit sadu funkcí, s libovolným počtem parametrů, jež mají velmi různorodé vlastnosti. Různorodost těchto vlastností dokáže velmi dobře prověřit chování a efektivitu optimalizačního algoritmu, neboť téměř vždy je globální extrém schován ve změti velmi složitých tvarů, jak bude dále zřejmé z obrázků ukázkových funkcí pro 2 parametry. Funkce budou demonstrovány na 2 parametrech, protože tři dimenzionální prostor jsme schopni vykreslit a představit si jej. Samozřejmě složitost těchto funkcí s rostoucím počtem parametrů je zachována a zřejmě hodně prohloubena.

Nespornou výhodou testovacích funkcí je znalost globálního extrému pro jakýkoli počet parametrů. To umožňuje použít absolutní měřítko hodnocení namísto pouhého srovnávání výsledku různých algoritmů s různými nastaveními, kde není jistota, že nalezené řešení je opravdu globálním extrémem.

4.1 VÝPOČET GLOBÁLNÍCH EXTRÉMŮ

Jak již bylo zmíněno v úvodu této kapitoly je velkou výhodou možnost přesného určení globálních extrémů funkcí pro jakýkoli počet parametrů. Některé funkce z galerie testovacích funkcí obsahují pouze možnost určení jednoho z globálních extrémů. Většinou se jedná o minimu což je kvůli praktickému použití výhodnější. Téměř vždy hledáme minimum např.nejkratší cestu, či nejmenší náklady. Funkce jsou koncipovány na složitost okolí globálního minima, proto výpočetní nalezení globálního maxima je o mnoho jednodušší a zabere méně strojového času.

4.1.1 Postup výpočtu globálních extrémů

Výpočet globálních extrémů některých funkcí z uvedené galerie testovacích funkcí je velmi jednoduchý. K určení globálního extrému dané funkce v *n*-rozměrném prostoru nám postačí znalost globálního extrému v prostoru jedné proměnné. Ten lze získat analyticky na určitém intervalu (což může být v některých případech značně složité), nebo vykreslením průběhu funkce pomocí nějakého

programu s dostatečnou jemností kroku. Následně nalezený globální extrém funkce jedné proměnné vynásobíme *n* (počet parametrů dimenze pro níž chceme znát globální extrém) a okamžitě dostáváme globální extrém dané funkce n proměnných [1][8].

Funkce z použité galerie testovacích funkcí u nichž funguje výše uvedený způsob výpočtu globálních extrémů :

1. 4th De Jong $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim} i \cdot x_i^4 \quad (4.1)$$

2. 1st De Jong $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim} x_i^2 \quad (4.2)$$

3. 3rd De Jong $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim} |x_i| \quad (4.3)$$

4. Rastring function $$f(x_1, x_2, ..., x_{Dim}) = 10 \cdot Dim + \sum_{i=1}^{Dim} (x_i^2 - 10 \cdot \cos(2\pi x_i)) \quad (4.4)$$

5. Schwefel function $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim} -x_i \cdot \sin(\sqrt{|x_i|}) \quad (4.5)$$

6. Michalewicz function $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim} -\sin(x_i) \cdot \sin\left(\frac{i \cdot x_i^2}{\pi}\right)^{20} \quad (4.6)$$

7. Easom function $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim-1} -\cos(x_i) \cdot \cos(x_{i+1}) \cdot e^{-((x_i-\pi)^2 + (x_{i+1}-\pi)^2)} \quad (4.7)$$

4.1.2 Omezení metody výpočtu globálních extrémů

Funkce z použité galerie testovacích funkcí u nichž nefunguje výše uvedený způsob výpočtu globálních extrémů :

1. Rosenbrock saddle $$f(x_1, x_2, ..., x_{Dim}) = \sum_{i=1}^{Dim-1} (100 \cdot (x_i^2 - x_{i+1}^2) + (1 - x_i)^2) \quad (4.8)$$

2. Grienwangks function $$f(x_1, x_2, ..., x_{Dim}) = 1 + \sum_{i=1}^{Dim} \frac{x_i}{4000} - \prod_{i=1}^{Dim} \left(\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) \quad (4.9)$$

3. Salomon function (4.10)

$$f(x_1, x_2, ..., x_{Dim}) = -\cos\left(2\pi \cdot \sqrt{\sum_{i=1}^{Dim} x_i^2}\right) + 0.1 \cdot \left(\sqrt{\sum_{i=1}^{Dim} x_i^2}\right) + 1$$

4. Ackley function (4.11)

$$f(x_1, x_2, ..., x_{Dim}) = -20 \cdot e^{-0.2 \cdot \sqrt{\frac{\sum_{i=1}^{Dim} x_i^2}{Dim}}} - e^{\left(\frac{\sum_{i=1}^{Dim} \cos(2\pi x_i)}{Dim}\right)} + 20 + e$$

Pro tyto čtyři funkce neplatí předpis pro výpočet globálních extrémů zmíněný v podkapitole 4.1.1. Ovšem všechny jsou sestaveny takovým způsobem, že globální minimum je vždy v 0. To je velmi cenná informace, protože ať je počet parametrů jakýkoli globální minimum vždy zůstane v 0.

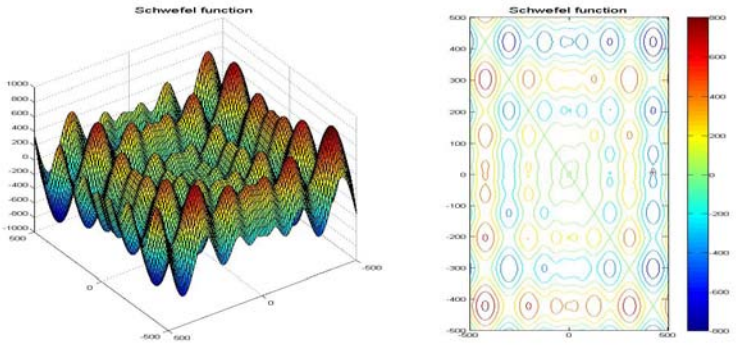
Hledání globálního maxima je složitější. Testovací funkce jsou koncipovány k hledání globálního minima. Díky této skutečnosti a díky zobrazení trojrozměrného grafu jsme schopni usoudit, že složitost funkce v okolí globálního maxima je zanedbatelná oproti složitosti v okolí globálního minima. Použitím algoritmu diferenciální evoluce a nastavení jejich parametrů tak jak bude ukázáno v kapitole 6, dojdeme k vyčíslení globálního maxima, pro určitý počet parametrů. Tato skutečnost byla vyzkoušena na čtyřech výše zmíněných funkcích během deseti nezávislých běhů algoritmů. Globální maximum bylo nalezeno vždy stejné a prohlášeno za nejvyšší hodnotu dané funkce 20 parametrů, kvůli použití ve výpočtech.

Veškeré hodnoty globálních extrémů (20 parametrů), definičních oborů na kterých byly funkce používány a zobrazení v trojrozměrném prostoru se nachází v příloze č.1.

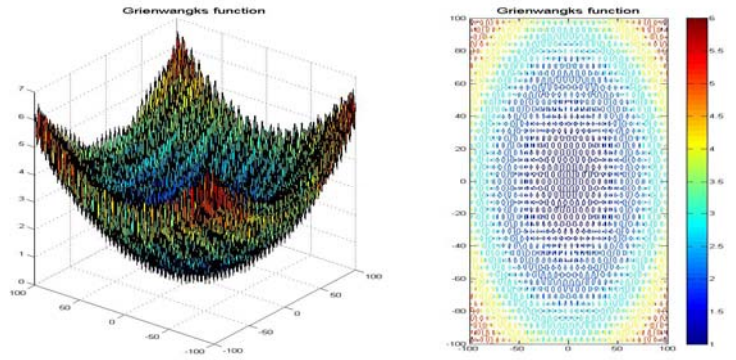
4.2 UKÁZKY TESTOVACÍCH FUNKCÍ

Následujících několik obrázků má přiblížit vlastnosti a složitost funkcí ze sestavené galerie testovacích funkcí. Funkce je vždy zobrazena na daném intervalu v trojrozměrném prostoru a taktéž je u každé funkce její vrstevnicové zobrazení. Celá

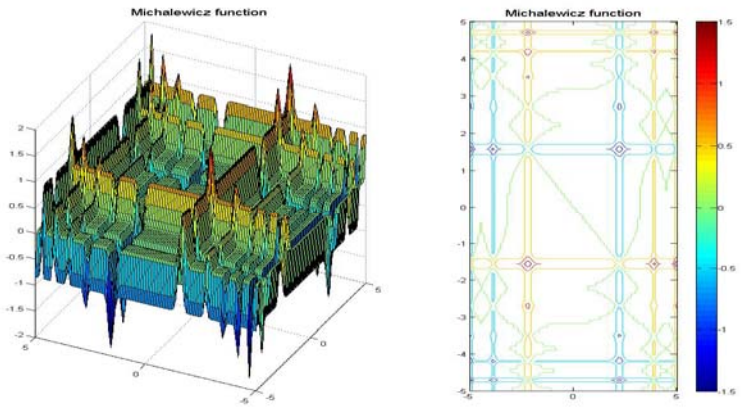
galerie testovacích funkcí s patřičnými parametry, funkčními předpisy atd. je zobrazena v příloze č.1.



4-1 Příklad funkce Schwefel



4-2 Příklad funkce Grienwangks



4-3 Příklad funkce Michalewicz

5. PROBLÉM OBCHODNÍHO CESTUJÍCÍHO A JEHO ŘEŠENÍ POMOCÍ GENETICKÉHO ALGORITMU A DIFERENCIÁLNÍ EVOLUCE

5.1 KOMBINATORICKÝ PROBLÉM

Kombinatorická optimalizace je silně omezená úloha, která pracuje s konečným počtem diskrétních stavů. Tato úloha je často porovnávána s diskrétní optimalizací s konečným počtem stavů. Z tohoto důvodu je kombinatorický problém rozdělen do dvou typů.

Prvním typem je *wide sense combinatorial problem* [9] (volně přeloženo – kombinatorický problém v širokém slova smyslu nebo pojetí). Z důvodu kvantování a následného numerického zpracování je spojitá funkce rozdělena na konečný počet dílků. Řešení probíhá výběrem diskrétní hodnoty daného parametru a to pro všechny parametry zvláště. Jako dobrý příklad se jeví problém výběru nejvhodnější čočky do teleskopu. I když optické parametry čočky jsou spojité veličiny, je třeba vybírat z konečného počtu čoček, které jsou komerčně dostupné [9]. Dalším příkladem může být optimalizace funkce z galerie testovacích funkcí jak je uvedeno v kapitole 4.

Druhým typem je *strict sense combinatorial problem* [9] (volně přeloženo – přísný kombinatorický problém). V tomto případě lze řešení problému vidět v konkrétním přeuspořádání prvků. Dobrým příkladem je problém obchodního cestujícího nebo problém batohu.

5.1.1 Problém obchodního cestujícího

Problém obchodního cestujícího je praktický příklad optimalizačního problému permutačního charakteru. Jedná se o seřazení cílových měst, které má obchodní cestující navštívit, takovým způsobem, aby výsledná cena za cestování mezi těmito městy byla optimální (v tomto případě minimální). V této práci předpokládáme symetrickou úlohu obchodního cestujícího. To znamená, že cena cesty z města A do města B je stejná jako cena cesty z města B do A. Ceny mají tedy

charakter vzdálenosti [3]. Dalším předpokladem je cesta z města A do B pouze po přímce.

Tento problém lze také interpretovat matematicky jako problém nalezení nejlevnějšího hamiltonovského cyklu v úplném ohodnoceném grafu s M uzly [3]. Takových cest existuje $M!$ z čehož vyplývá permutační charakter úlohy.

Posloupnost měst je zakódována jako vektor velikosti M, kde na první pozici je číselné označení města, které je navštíveno jako první. Na druhém místě vektoru je číslo města, které je navštíveno jako druhé atd. Ještě před vytvořením těchto vektorů je nutné města vhodným způsobem očíslovat nebo označit.

5.2 ŘEŠENÍ PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO POMOCÍ GENETICKÉHO ALGORITMU

Řešení problému obchodního cestujícího pomocí genetického algoritmu se svou technikou značně liší od klasické optimalizace účelové funkce pomocí tohoto algoritmu. Hlavním důvodem je číslování měst pomocí celých čísel. Jestliže by byl použit klasický genetický algoritmus docházelo by po křížení a mutaci k neceločíselným označením měst. Tato skutečnost vede k další otázce, jakým způsobem převézt označení měst opět na celočíselné hodnoty takovým způsobem, aby byla zachována efektivita algoritmu. V případě jiného označení měst než pomocí celých čísel by se klasické metody křížení a mutace taktéž neuplatnily.

Kvůli výše uvedeným důvodům byly vynalezeny speciální techniky křížení a mutace pro řešení problému obchodního cestujícího. Je třeba zmínit, že mutace se při řešení toho problému téměř nepoužívá [3]. Pokud ano, jedná se o mutaci typu náhodné výměny dvou měst, vygenerování nového jedince nebo výměny určitých úseků jedince. Technik křížení je také několik. Jedná se o *operátor křížení se zachováním pořadí – OX*, *operátor křížení s částečným zobrazením – PMX* a *operátor křížení s rekombinací hran – ERX* [3]. Posledně zmíněný, operátor ERX je použit v této práci neboť bylo experimentálně dokázáno, že tento operátor dává nejlepší výsledky, protože potomek přebírá od rodičů potenciálně dobré segmenty celé cesty [3].

ERX pracuje s tzv. *hranovou tabulkou.* Jedná se o tabulku v níž jsou ke každému městu přiřazeni jeho sousedé obou křížících se jedinců. Křížení probíhá vybíráním města s nejmenším počtem sousedů. Pokud je těchto měst více vybere se náhodně. Po výběru se město zařadí do vektoru potomka a vyškrtne ze všech sousedních míst v hranové tabulce. Takto se postupuje až do naplnění vektoru měst potomka. Zvláštnost operátoru ERX je, že ze dvou rodičů vytvoří jednoho potomka.

Techniky selekce, zastavovací podmínky, elitismus a struktura genetického algoritmu, použitého pro problém obchodního cestujícího, se oproti klasickému genetickému algoritmu nemění.

5.3 ŘEŠENÍ PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO POMOCÍ DIFERENCIÁLNÍ EVOLUCE

K řešení problému obchodního cestujícího pomocí diferenciální evoluce bylo také vypracováno několik speciálních technik řešení. Je zapotřebí zmínit techniku pomocí *permutační matice* dále pomocí *matice dosažitelnosti* a nakonec *relativní indexování pozicí* [9]. Bohužel výsledky těchto úprav algoritmu diferenciální evoluce nejsou příliš dobré. První dvě zmíněné techniky spíše připomínají vlastní algoritmus řešení problému obchodního cestujícího než modifikaci algoritmu diferenciální evoluce. Z toho důvodu je v této práci použita technika *relativního indexování pozicí.*

Tato metoda je velmi podobná klasické diferenciální evoluci, nicméně je vynechán předposlední krok, tudíž výběr prvku podle CR z aktuálního, či šumového vektoru. Důležité je, aby města byla označena čísly nebo, aby vektory jedinců obsahovaly čísla pozic v předem definovaném vektoru označení. Vektory tří náhodně vybraných jedinců jsou poděleny celkovým počtem měst. Dále je klasicky od prvního vektoru odečten druhý. Výsledný vektor je vynásoben mutační konstantou F a tento nově vzniklý vektor je sečten s vektorem třetího vybraného jedince. Nyní je nejvyšší číslo ve vektoru prohlášeno za poslední město v daném značení, druhé nejvyšší číslo za předposlední město atd. Následně se nově vzniklý vektor ohodnotí a porovná s vektorem ve staré populaci, který je právě na řadě. Jestliže je ohodnocení

lepší zkopíruje se nově vzniklý vektor do nové populace, pokud je horší postupuje do nové populace vektor ze staré populace, který byl právě na řadě [9].

6. IMPLEMENTACE ALGORITMŮ A POPIS POUŽITÝCH FUNKCÍ PRO TESTOVÁNÍ NA GALERII TESTOVACÍCH FUNKCÍ

Veškeré optimalizační algoritmy, jejich transformace, pomocné funkce, kontrola definičního oboru i galerie testovacích funkcí jsou naprogramovány v prostředí Matlab a jsou součástí přílohy č.2. Kódování parametrů je vždy pomocí reálných čísel z důvodu větší přehlednosti a jednoduchosti. Konstanty se neměnily jen s výjimkou případů popsaných v kapitolách 6.1 a 6.3 Vysvětlení účelů všech použitých konstant v programech je na první straně přílohy č.1 nebo v seznamu zkratek.

Tabulka 6-1 Nastavení parametrů jednotlivých algoritmů

Tabulka nastavení konstant pro jednotlivé algoritmy			
DE	**GA**	**SA**	**TS**
D = 20;	D = 20;	D = 20;	D = 20;
NP = 10*D;	NP = 10*D;	T = 1000;	NP=1;
CR = 0.5;	GENERACE = 600;	Tmin=1;	koefOkoli=500;
F = 0.3;	KK=0.82;	alpha=0.9;	Tmax=6000;
GENERACE =600;	MK=0.16;	kmax=2000;	kmax=50;
psi=0.0001;	B=5;	tmax=10*kmax;	pocStejRes=50;
pocStejRes=50;	psi=0.0001;		pocTrans=20;
	pocStejRes=50;		eps=0.01;

Všem algoritmům jsou společné následující funkce:

FUNKCE (param,D) – obsahuje galerii 11 testovacích funkcí, vybírá se vždy jen jedna pomocí globální proměnné CISFCE, vstupem je vektor parametrů o velikosti *D* (určuje dimenzi), ohodnocuje jedince s parametry *param*

defObor (hranice,Population,NP,D) – hlídá, aby nedocházelo k překročení definičního oboru, pokud dojde automaticky parametr nahradí jednou z mezních hodnot definičního oboru, hranice je matice která v prvním sloupci obsahuje dolní hranici definičního oboru a v druhém sloupci horní hranici definičního oboru (použití pouze pro kompaktní intervaly omezení), *Population* je matice, která obsahuje v každém sloupci jednoho jedince, v prvním řádku je jeho ohodnocení účelovou funkci a v dalších jsou parametry, jejich počet je *D*, *NP* určuje kolik jedinců je posláno k vyhodnocení překročení omezení

genPopulation (NP,defOborXY,typ,D) – vygeneruje náhodnou populaci o velikosti *NP*, z intervalu který definuje matice deOborXY, typ určuje zda se jedná o GA (typ=1) nebo jiný algoritmus (typ=0)

6.1 GENETICKÝ ALGORITMUS

Pro implementaci genetického algoritmu byla zvolena jako selekční metoda poziční selekce. Křížení bylo provedeno dle tří vzorců (3.1, 3.2, 3.3) a mutace podle Michalewiczova předpisu pro dynamickou mutaci dle vzorců (3.4, 3.5). Náhradová strategie byla takového typu, že se náhodně vybrali dva rodiče dle pravděpodobnosti určené poziční selekcí. Podle náhodného čísla a konstanty křížení se určilo zda se bude křížit. Pokud ano oba jedinci projdou křížícím procesem, následně možnou mutací jednotlivých parametrů a nakonec jsou zařazeni do nové populace. Pokud ne jsou oba vybraní jedinci zkopírování do dočasné populace, projdou možnou mutací a pak jsou zařazeni do nové populace. Samozřejmě je použit princip elitismu.

Konstanty jsou po celou dobu stejné až na případ kdy se zabýváme možnostmi selhání genetického algoritmu v kapitole 8.3.3. U tohoto případu jsou vždy použité konstanty zobrazeny vedle patřičné tabulky v příloze č.1.

6.2 DIFERENCIÁLNÍ EVOLUCE

Implementace diferenciální evoluce je dle (DE/rand/1/bin) [1], princip tohoto druhu diferenciální evoluce je zobrazen na obrázku 3-2 Princip diferenciální evoluce.

6.3 SIMULOVANÉ ŽÍHÁNÍ

Simulované žíhání využívá k výpočtu transformace okolí funkci **annealing_transform (defOborXY,F,D,T,Tpom),** která pomocí omezení definičním oborem, původním vektorem *F*, náhody a teploty *T* vypočítá nový vektor řešení. Vzdálenost transformace je snižována s klesající teplotou.

Simulované žíhání používá techniku podobnou elitismu. Vždy po zvýšení teploty *T* je jako výchozí bod brán dosud nejlepší výsledek, abychom jej neztratili a dobře prozkoumali jeho okolí.

Konstanty byly změněné jen u testování pro neomezený počet přístupu k účelové funkci a funkci Schwefel. Funkce byla pro simulované žíhání tak složitá, že během jeho běhu nedošlo k využití zastavovací podmínky a volání účelové funkce se dostalo přes jeden milion. Matice pro 100 běhů programu se z toho důvodu nevešla do operační paměti. Proto byla konstanta *kmax* zmenšena z 2000 na 500, aby se výsledky daly zapsat do operační paměti testovací stanice.

Druhým případem změny konstant byl případ všech testování pro omezený počet přístupů k účelové funkci. Kdyby se neprovedlo omezení *kmax* na 500 a koeficientu snižování teploty *alpha* na 0,75, nedošlo by ani k jedné změně teploty díky omezení počtu přístupů k účelové funkci na 20400. Tato skutečnost by znamenala nevyužití vlastnosti simulovaného žíhání.

6.4 ZAKÁZANÉ PROHLEDÁVÁNÍ

Zakázané prohledávání je implementováno klasickým přístupem. Jedinou výtkou, která z deterministického algoritmu tvoří algoritmus smíšený je transformace okolí výchozího bodu. Transformace dle náhodného čísla buď zachová hodnotu parametru, nebo od něj odečte či přičte koeficient okolí. Tento postup byl zvolen z důvodů velké složitosti návrhu deterministické transformace okolí pro velký počet parametrů.

7. IMPLEMENTACE ALGORITMŮ A POPIS POUŽITÝCH FUNKCÍ PRO ŘEŠENÍ PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO

Optimalizační algoritmy diferenciální evoluce a genetický algoritmus, funkce pro výpočet vzdálenosti mezi městy, pro tvorbu nové populace a pro vygenerování polohy měst jsou naprogramovány v prostředí Matlab a jsou součástí přílohy č.2. Konstanty algoritmů jsou u všech simulací vždy stejné. Vysvětlení účelů všech použitých konstant v programech je na první straně přílohy č.1 nebo v seznamu zkratek.

Tabulka 7-1 Nastavení parametrů jednotlivých algoritmů

Tabulka nastavení konstant pro jednotlivé algoritmy - TSP	
D = 30;	D = 30;
NP = 3*D;	NP = 3*D;
CR = 0.5;	GENERACE = 600;
F = 0,85;	KK=0.85;
GENERACE =600;	MK=0.2;
KTS=0.45;	KTS=0.3;
KGNJ=0.25;	KGNJ=0.35;
pocStejRes=50;	pocStejRes=50;

Pro oba algoritmy jsou společné tyto funkce :

fitness(Cities,Population,NP,D) – funkce počítá celkovou vzdálenost mezi městy. Vzdálenost je měřena jako součet přímek vzdálenosti mezi jednotlivými, po sobě jdoucími městy uloženými v jednotlivých sloupcích matice *Population*. První položka v každém sloupci matice je celková vzdálenost a ostatní už je označení měst, kterými má projet v daném pořadí. Matice *Cities* obsahuje v prvním sloupci

souřadnici *x* města a v druhém sloupci souřadnici *y* města příslušného řádku. Funkce vrací ohodnocenou populaci a ohodnocuje pouze neohodnocené jedince, kteří mají na místě celkové vzdálenosti hodnotu inf.

genCities (D,typ,omezeniXY) – funkce, která vygeneruje rozložení měst v kartézských souřadnicích s omezením pomocí proměnné *omezeniXY*. Počet měst je *D* a typ označuje zda budeme generovat města s náhodným rozložením, rovnoměrně kruhovým rozložením, rovnoměrně tangenciálním rozložením nebo náhodně tangenciálním rozložením.

genRandPopulation (D,NP) - funkce vytvoří náhodnou populaci o *NP* jedincích a velikosti každého jedince *D*. Ohodnocení takto vzniklých nových jedinců je implicitně nastaveno na inf.

7.1 GENETICKÝ ALGORITMUS

Křížení v genetickém algoritmu je zvoleno ERX jak je popsáno v kapitole 5.2 a [3]. Mutace je taktéž použitá a pomocí náhodného čísla a konstanty *KGNJ* se rozlišuje mezi vygenerováním náhodného nového jedince a mezi výměnou dvou náhodně zvolených měst. Samozřejmě mutace i selekce má určité pravděpodobnosti vykonání jak dokládají hodnoty *MK* a *KK*. Náhradová strategie se řídí *KK* tzn., jestliže nedojde ke křížení náhodně se zvolí jeden z rodičů, který je následně zkopírován do nové populace. Selekce je zavedena na konec cyklu programu a je typu turnaj, přičemž je použit i elitismus.

7.2 DIFERENCIÁLNÍ EVOLUCE

Implementace algoritmu diferenciální evoluce se řídí popisem v kapitole 5.3 a [9]. Jedinou výjimku tvoří použití jakési náhrady selekce na konci cyklu algoritmu, kdy je na základě náhody a proměnné *KTS* vybírána nová populace buď z rodičů nebo potomků.

8. SROVNÁNÍ ALGORITMŮ

Cílem této práce bylo naprogramovat a porovnat algoritmy diferenciální evoluce a genetický algoritmus. Jako přidružené algoritmy k hlubšímu porovnání byly vybrány algoritmy zakázaného prohledávání a simulovaného žíhání, jež byly použity pouze v první části a to při testování na galerii testovacích funkcí.

V první části testování byla vytvořena galerie testovacích funkcí, která obsahuje funkce zmíněné v kapitole 4.1.1 a 4.1.2 Galerie byla sestavena z takových funkcí u nichž jsme schopni předem určit globální minimum. Dále byla požadována jistá různorodost co se týče tvarů funkcí. Složitost, velké množství lokálních extrémů, více globálních extrémů, pásy oblastí jako lokální extrém, či globální extrém jako díra v ploše, to vše mělo dostatečně prověřit kvalitu optimalizačních algoritmů.

Veškeré zdrojové kódy byly psány v jazyku Matlab a jsou obsahem přílohy č.2. Každá funkce byla 20 parametrová tzn., že jsme hledali globální minimum na 20 parametrové hyperploše v 21 rozměrném prostoru. Jednotlivé hledání globálního minima bylo omezeno definičními obory jako 20 parametrovou hyperkrychlí (tzn. všechny parametry měly stejné omezení). Hledání globálního minima bylo pro každou funkci opakováno 100krát nezávisle na předchozích běžích programu.

Abychom předešli zbytečnému volání účelové funkce byla kontrola definičního oboru volána vždy před vstupem k ohodnocení pomocí účelové funkce. Pro ošetření překročení omezujících podmínek účelové funkce byla použita funkce, která překročení detekovala a napravila. Oprava znamenala posunutí parametru, jež byl mimo obor platnosti na pomyslnou hranu omezující hyperkrychle. Zjednodušeně řečeno parametr byl nastaven na okraj své omezující podmínky.

Testování bylo nejprve provedeno pro neomezený přístup k účelové funkci. Následně bylo vycházeno z praktických požadavků na vyřešení určitého problému v daném časovém intervalu. Jako časově nejvytíženější byla zvolena operace výpočtu účelové funkce. Z těchto důvodů bylo provedena druhá sada testování se stejným nastavením parametrů a omezeným počtem přístupů k účelové funkci. Jednalo se o fiktivní případovou studii popsanou v kapitole 8.3.1.

V druhé části testování byl řešen problém obchodního cestujícího. V tomto případě byly použity pouze dva optimalizační algoritmy a to diferenciální evoluce a genetický algoritmus. Zdrojové kódy byly taktéž jako v první části testování programovány v prostředí Matlab. Bylo zvoleno 30 měst ve třech typech rozmístění. Prvním bylo náhodné rozmístění měst, druhým bylo rovnoměrné rozmístění do kruhu (u tohoto typu dokážeme jednoduše vypočítat minimální vzdálenost mezi městy) a třetím bylo rovnoměrné rozmístění měst do tangenty. Města byla generována v kartézských souřadnicích a očíslována od 1 do 30. Vzdálenost mezi dvěmi městy byla brána jako přímka tyto města spojující a celková vzdálenost byla součtem všech těchto přímek v daném pořadí.

Všechny simulace taktéž probíhaly 100krát a nejprve byly provedeny pro neomezený počet přístupů k účelové funkci se zastavující podmínkou 50ti po sobě jdoucích stejných nejlepších řešení a poté s omezeným počtem přístupů k účelové funkci s omezením na 20 000 přístupů.

Výsledná data byla graficky a tabelárně zpracována. Z důvodů závislosti algoritmů na náhodě byla data zpracována statisticky a byl kladen důraz na celkové výsledky, ne na výsledky jednoho konkrétního běhu programu.

8.1 TESTOVÁNÍ NA GALERII TESTOVACÍCH FUNKCÍ

Testovací funkce jsou různé složitosti, jak je možné vidět v příloze č.1. Tato různorodost může mít na svědomí různé pořadí kvality algoritmů.

Abychom byli schopni objektivně zhodnotit výsledky algoritmů musíme určit kritérium podle nějž budeme hodnotit. Jelikož známe hodnoty globálních extrémů nabízí se absolutní vzdálenost nalezeného minima od globálního minima. Tato metoda je dosti zavádějící, protože pokud je celý obor hodnot ve statisících a my nalezneme minimum pomocí některého algoritmu v řádově desítkách vzdálený od globálního minima, je výsledek velmi dobrý. Z toho důvodu se nabízí možnost vypočítat procentuální vzdálenost nalezeného minima od globálního minima. Vypočtené číslo nám podá lepší představu o výsledku a lepší intuitivní odhad kvality.

Srovnávané algoritmy jsou více či méně závislé na náhodě. Tento jev nám komplikuje hodnocení. Neexistuje analytický předpis nebo důkaz, kterým jsme

schopni dokázat proč evoluční algoritmy fungují (výjimku tvoří kontroverzní teorie schémat [3]). Ze zkušenosti naopak víme, že opravdu fungují a přinášejí uspokojivé výsledky. Z těchto důvodů je vhodnější výsledky zpracovat statisticky. Tímto získáme lepší představu o celkové schopnosti algoritmů řešit daný problém. Použití průměru může být značně zavádějícím kritériem v takto náhodou ovlivňovaných případech. Jedna velká změna výsledku může velmi ovlivnit průměrnou hodnotu. Vhodnější bude srovnání průměru s mediánem a výpočet směrodatné odchylky, jež nám dává informaci o různorodosti výsledků.

8.1.1 Výpočet procentuální vzdálenosti od globálního minima

Procentuální vzdálenost mezi nalezeným minimem a globálním minimem se spočte dle rovnice (8.1)

$$Dist_{\%} = 100 \frac{\left| Min_{globální} - Min_{nalezené} \right|}{\left| Max_{globální} - Min_{globální} \right|} \tag{8.1}$$

8.1.2 Popis tabulek v příloze č.1

Výsledky byly zapsány do tabulek v příloze č.1. Tabulka vždy obsahuje dvě optimalizační metody a jejich výsledky pro danou testovanou funkci. Ke každé testované vlastnosti je uvedena její minimální a maximální hodnota nalezená za 100 běhů programu. Její průměr, medián a směrodatná odchylka.

Vlastnosti sloužící ke zhodnocení jsou:

Počet generací pro nalezení minima – u genetického algoritmu a diferenciální evoluce nám tato hodnota dává přehled kolik generací bylo potřeba k zastavení běhu programu (nalezení minima, které se po jistou dobu nezmění)

Procentuální vzdálenost od minima – procentuální vzdálenost od globálního minima spočtena dle (8.1)

Počet přístupu k účelové funkci – vypovídá o počtu volání účelové funkce, které bylo potřeba k zastavení běhu programu

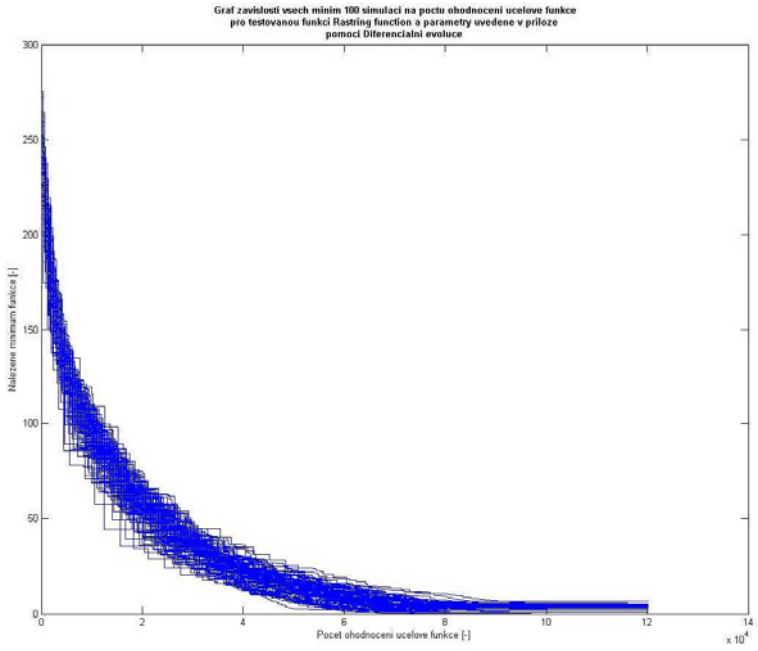
Hodnoty nalezených minim – konkrétní číslo minima nalezené algoritmem

Tabulka 8-1 Výsledky DE a GA pro neomezený počet přístupů k účelové funkci a testovanou funkci Grienwangks

Optimalizační metoda:	Diferenciální evoluce (DE)					Genetický algoritmus (GA)				
	min	max	průměr	medián	směrodatná odchylka	min	max	průměr	medián	směrodatná odchylka
Počet generací pro nalezení minima	198	268	216	212	14,17	63	540	465	523	145,76
Procentuální vzdálenost od minima [%]	$1{,}19*10^{-6}$	0,019	$3{,}91*10^{-6}$	$3{,}42*10^{-5}$	0,0024	$2{,}06*10^{-6}$	3,34	0,39	$1{,}4*10^{-5}$	0,946
Počet přístupu k účelové funkci:	39800	53800	43334	42600	2835	27876	238031	204360	230510	64126
Hodnoty nalezených minim	$6{,}1*10^{-7}$	$3{,}7*10^{-6}$	$1{,}4*10^{-6}$	$1{,}19*10^{-6}$	$4{,}7*10^{-7}$	$1{,}06*10^{-6}$	1,7	0,2	$8{,}44*10^{-6}$	0,4825

8.1.3 Popis grafů v příloze č. 1

Příloha č.1 obsahuje pro každou testovanou funkci 24 grafů. Jsou to tři druhy grafů pro čtyři optimalizační algoritmy a testování pro omezený a neomezený přístup k účelové funkci.

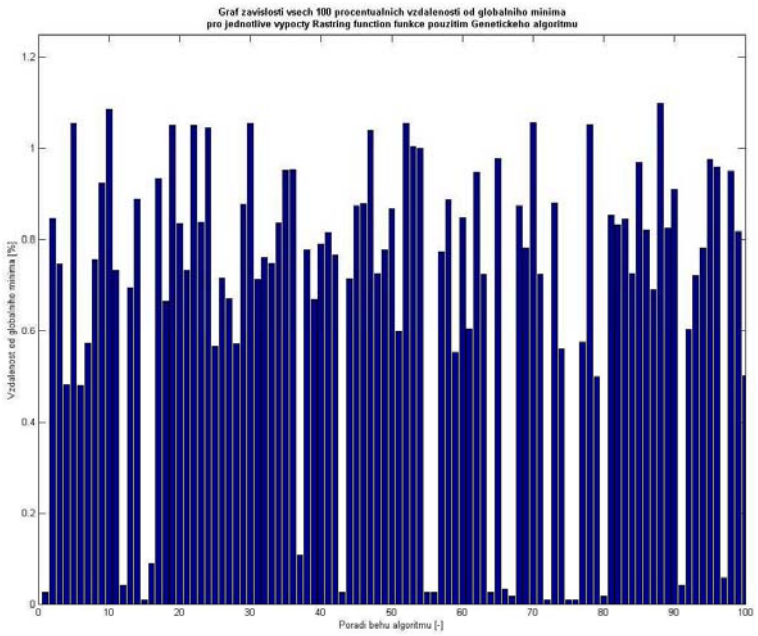


8-1 Závislost nalezeného minima na počtu ohodnocení účelové funkce pro 100 běhů DE

Na Obr. 8.1 vidíme závislost nalezeného minima na počtu volání účelové funkce. Jak se mění nalezené minimum populace se zvětšujícím počtem volání

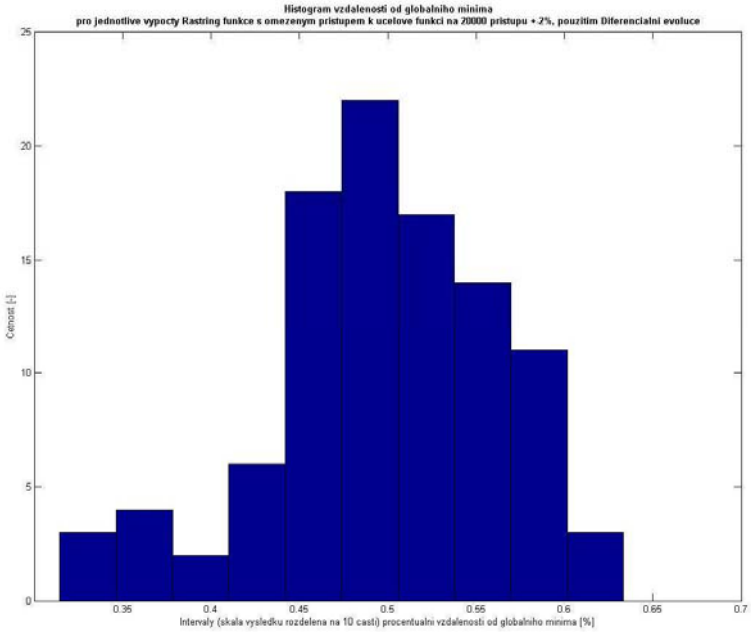
účelové funkce až k zastavení běhu programu. Do grafu je vyneseno všech 100 křivek, abychom měli představu o vlastnosti algoritmu na dané funkci. Křivky jsou po částech konstantní z důvodu zjišťování prozatímního minima až po vykonání všech transformací, křížení, mutací atd. (toto vede k volání účelové funkce NP-krát + volání při překročení mezí definičního oboru)

Obr. 8.2 ukazuje jak se měnila procentuální vzdálenost od globálního minima v průběhu všech 100 výpočtů pro danou funkci. Na tomto grafu lze dobře vidět různorodost nalezeného řešení během jednotlivých výpočtů.



8-2 Zobrazení procentuálních vzdálenosti nalezených minim od globálního minima pro všech 100 běhů programu

V histogramu z obrázku 8.3 je obsažena informace o četnosti jednotlivých procentuálních vzdálenostech od globálního minima v daných intervalech. Lze z toho vypozorovat rozložení výsledků a jejich ucelenost. Rozložení bylo vždy rozděleno do deseti intervalů.



8-3 Histogram procentuální vzdálenosti od globálního minima pro všech 100 běhů programu

8.2 TESTOVÁNÍ PRO NEOMEZENÝ POČET PŘÍSTUPŮ K ÚČELOVÉ FUNKCI NA GALERII TESTOVACÍCH FUNKCÍ

Neomezený počet přístupů k účelové funkci znamená, že počet maximálního počtu přístupu je nekonečně velký. Všechny algoritmy v tomto případě používaly zastavovací podmínku. Ta byla formulována jako 50 po sobě jdoucích generací (kol, sekvencí) stejného výsledku. Pokud bylo tohoto dosaženo výpočet se zastavil a veškerá data vypočtena do té doby byla brána jako výsledek.

Skutečnost, že se výpočet ukončil po splnění výše uvedené podmínky nahrávala uváznutí v lokálním extrému. Na druhou stranu nám dává informaci o tom jak „rychle“ je algoritmus schopný se vyprostit z toho extrému a efektivně pokračovat dále ve výpočtu. Druhá možnost zastavení běhu programu bylo dosažení maximálního počtu generací (kol, sekvencí).

Další z informací, které nám poskytlo testování pro neomezený počet přístupů k účelové funkci je přesnost výsledků jestliže nejsme časově omezeni (což se v praxi téměř nevyskytuje).

8.2.1 Vyhodnocení výsledků pro jednotlivé testovací funkce

Vyhodnocení bylo provedeno na základě procentuálního minima a na základě mediánu procentuálního minima. Vše je shrnuto v Tabulce 8.2 a Tabulce 8.3, kde nejsou zobrazeny konkrétní číselné výsledky (příloha č.1), ale pořadí algoritmů v pomyslné soutěži čtyř algoritmů. Celkové vyhodnocení je patrné ze součtu pořadí daného algoritmu pro všechny testovací funkci.

Tabulka 8-2 Výsledné pořadí algoritmů (minimum) pro neomezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na minimální nalezené procentuální vzdálenosti od globálního minima pro neomezený počet přístupů k účelové funkci				
funkce	**DE**	**GA**	**SA**	**TS**
4th De Jong	1	3	2	4
1st De Jong	1	2	4	3
Rosenbrock saddle	2	3	4	1
3rd De Jong	1	2	4	3
Rastring	1	3	2	4
Schwefel	1	4	3	2
Grienwangks	1	2	4	3
Michalewicz	1	2	4	3
Easom	1	3	2	4
Salomon	1	1	1	4
Ackley	1	2	3	4
Součet pořadí	**12**	**27**	**33**	**35**

Tabulka 8-3 Výsledné pořadí algoritmů (medián) pro neomezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na mediánu minimálních nalezených procentuálních vzdálenosti od globálního minima pro neomezený počet přístupů k účelové funkci				
funkce	**DE**	**GA**	**SA**	**TS**
4th De Jong	1	3	2	4
1st De Jong	1	2	4	3
Rosenbrock saddle	2	3	4	1
3rd De Jong	1	2	4	3
Rastring	2	3	1	4
Schwefel	1	4	2	3
Grienwangks	2	1	4	3
Michalewicz	1	2	4	3
Easom	1	3	2	4
Salomon	2	2	1	4
Ackley	1	2	3	4
Součet pořadí	**15**	**27**	**31**	**36**

Z výsledku zobrazených v Tabulce 8.2 vidíme, že si nejlépe vedl algoritmus diferenciální evoluce. Navíc jen jednou byl druhý. Výsledky genetického algoritmu byly různorodé, ale poslední skončil jen jednou. Třetí a čtvrtý skončily algoritmy simulovaného žíhání a taboo search se vzájemným minimálním odstupem.

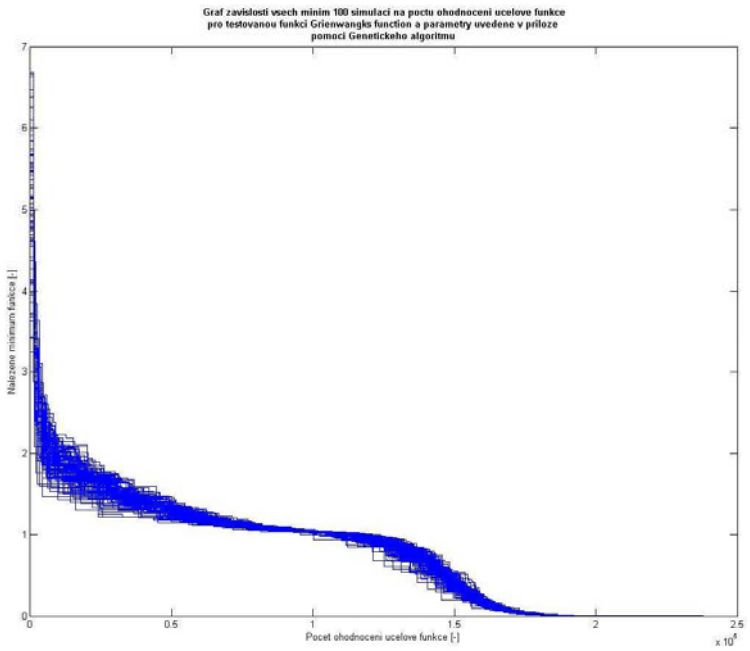
Výsledky v Tabulce 8.2 jsou velmi zatížené náhodou. U každé funkce se jedná o výběr pouze nejmenšího nalezeného optima ze 100 běhů. Lépe si výsledky představíme v Tabulce 8.3, kde jsou zpracovány mediány procentuálních minim, což je objektivnější kritérium. Nejlépe skončil algoritmus diferenciální evoluce, sice už ne tak suverénně jako v předchozím případě, ale stejně velmi dobře oproti ostatním. Druhý byl genetický algoritmus opět s jedním posledním místem. Dále simulované žíhání a nakonec taboo search.

V tomto srovnání skončil nejlépe algoritmus diferenciální evoluce. Genetický algoritmus pokulhával a skončil druhý. Přidružené algoritmy simulované žíhání a

taboo search skončily třetí a čtvrtý. Při pohledu na výsledky vyvstává otázka kolik času a kolik přístupu k účelové funkci dané algoritmy využily a za jak dlouho byly ukončeny podmínkami. Vše je podrobně popsáno v tabulkách v příloze č.1 a lze si z nich udělat objektivní názor.

8.2.2 Vlastnosti vybraných testovacích funkcí zobrazené ve výsledcích

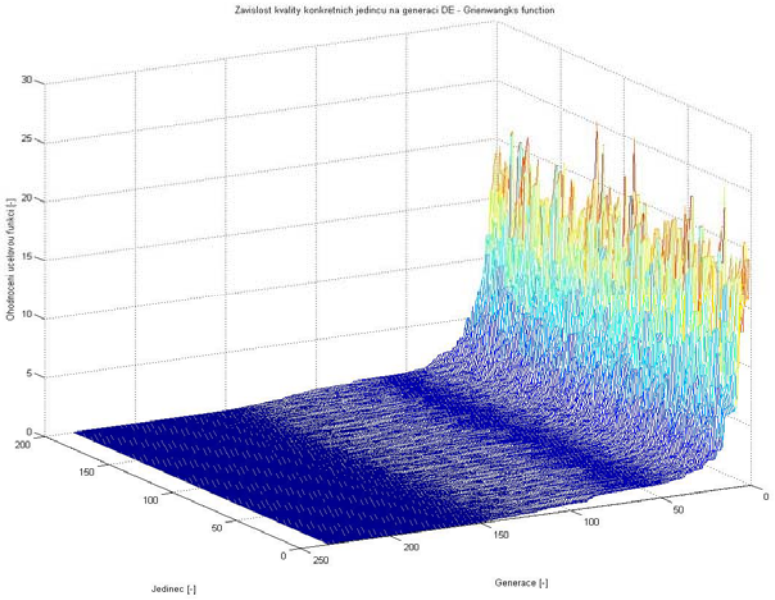
Testovací funkce jsou velmi různorodé jak již bylo uvedeno v kapitole 4 a v příloze č.1. Avšak je zajímavé ukázat si jak se tyto různorodosti projeví v grafických výsledcích testování.



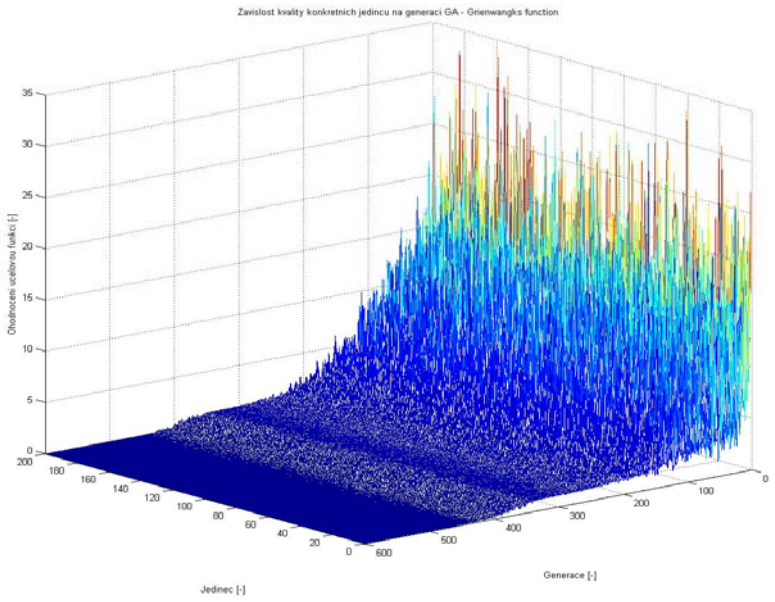
8-4 Závislost procentuálních minim na počtu ohodnocení účelové funce Grienwangks pomocí GA

Funkce Grienwangks (Obr.4.2) má velmi mnoho lokálních extrémů. V okolí globálního extrému na hodnotě 1 existují sedla, která jsou pro algoritmy velmi těžko překonatelná jak lze vidět z obr 8.4. Nicméně téměř vždy jsou překonána a evoluce pokračuje dále ke globálnímu minimu. Obdobné průběhy jsou patrné i u diferenciální evoluce a simulovaného žíhání. Naprosto stejný jev uvidíme

také na průběhů všech jedinců jednoho běhu v závislosti na generacích (obr 8.5 a obr 8.6). Pruh sedla, kdy všichni jedinci mají po určitou dobu téměř stejnou hodnotu.

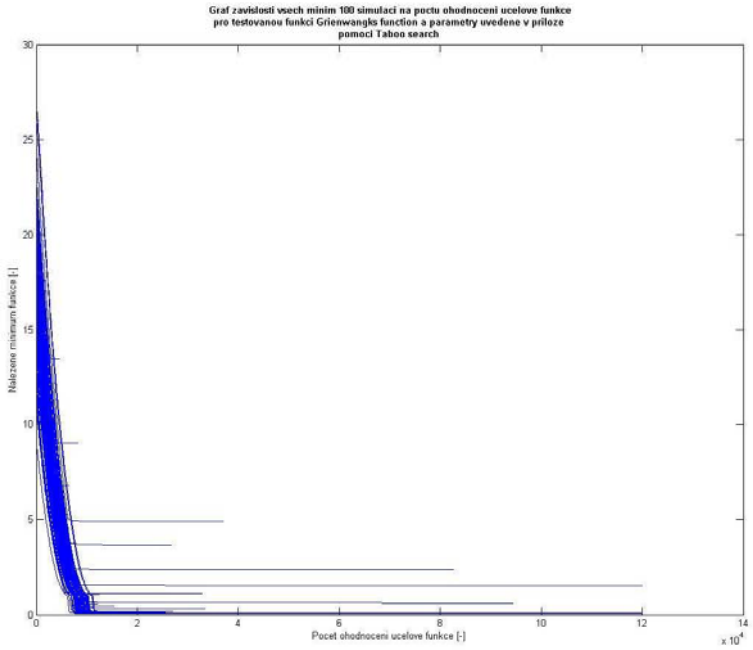


8-5 Graf kvality jednotlivých jedinců v závislosti na generacích, funkce Grienwangks a DE



8-6 Graf kvality jednotlivých jedinců v závislosti na generacích, funkce Grienwangks a GA

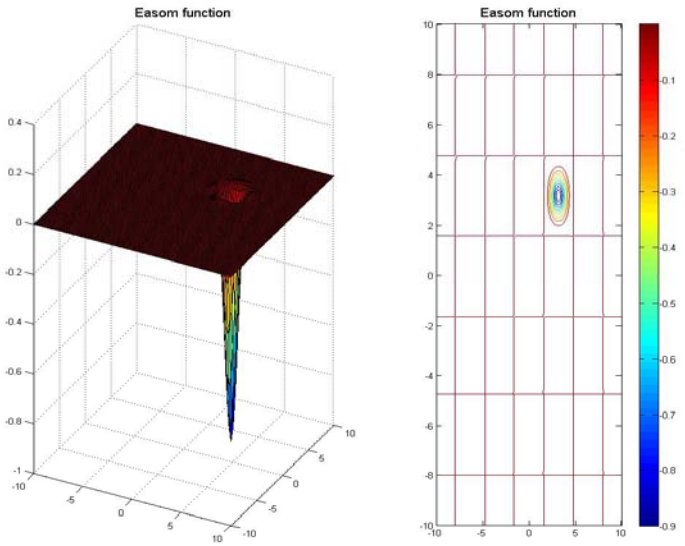
Zajímavá je situace, která nastává při použití taboo search. Na obr. 4.2 vidíme, že lokální extrémy jsou periodické a hluboké. Vhodnou volbou kroku transformace může nastat případ kdy se velmi složitá funkce Grienwangks, s nadsázkou řečeno, aproximuje útvarem podobajícím se *n*-rozměrnému eliptickému paraboloidu. Krok je dostatečně velký, aby lokální extrémy přeskakoval. Tato forma účelové funkce je pro gradientní typ optimalizační metody téměř vzorovým případem a řeší ji velmi efektivně jak lze vidět na obr. 8.7.



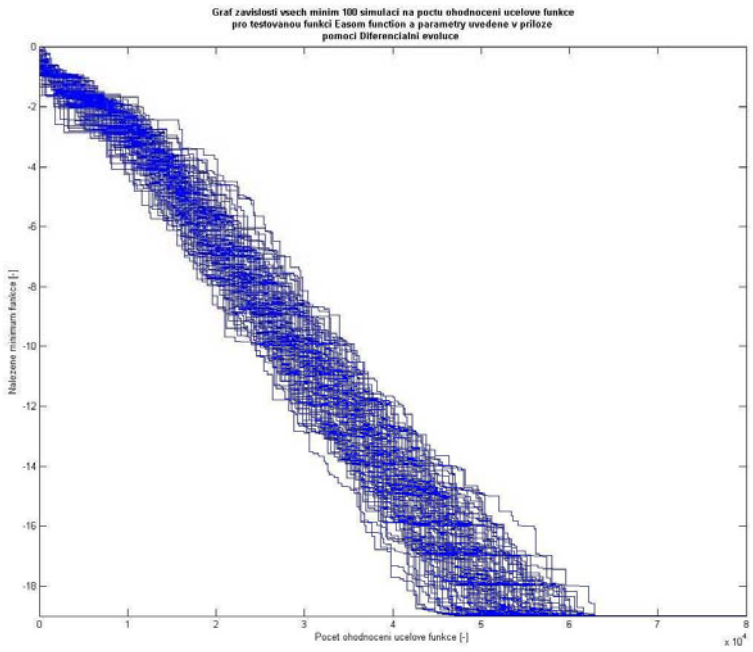
8-7 Závislost procentuálních minim na počtu ohodnocení účelové funkce Grienwangks a TS

Funkce Easom (obr.8.8) je velmi zrádná, má jediný globální extrém, který se nachází na rovině a vypadá jako díra. Nalezení globálního extrému je buď věcí náhody nebo dokazuje robustnost optimalizačního algoritmu. Krásný příklad síly diferenciální evoluce je vidět z průběhu na obr. 8.9. Průběh ukazuje postupné šlechtění populace v časovém horizontu téměř „lineárně“. Tento algoritmus také jako jediný nalezl globální extrém a to téměř vždy. Podobně lze pohlížet na průběh kvality jedinců pro jednotlivé jedince a generace. obr 8.10 ukazuje jak diferenciální

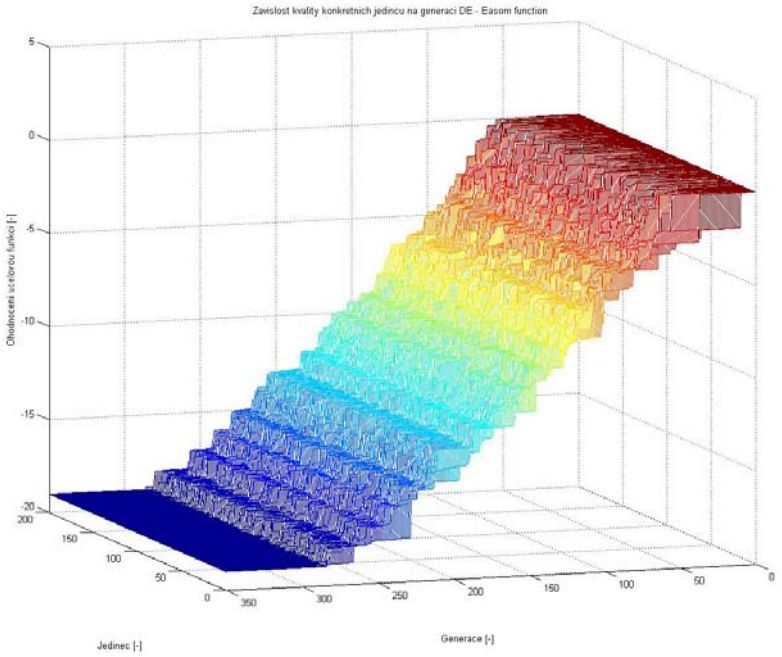
evoluce pracuje jako celek. Schodovitost naznačuje dosažení určité kvality minima všech jedinců v určitém časovém intervalu.



8-8 Funkce Easom



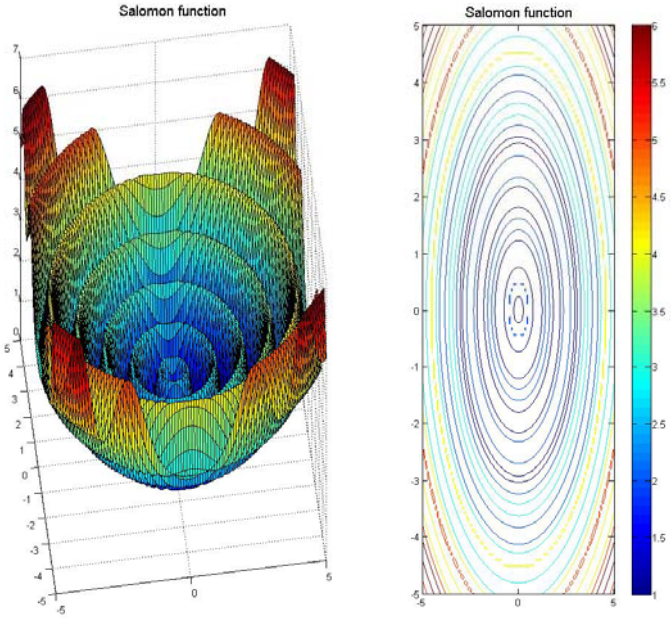
8-9 Závislost procentuálních minim na počtu ohodnocení účelové funkce Easom a DE



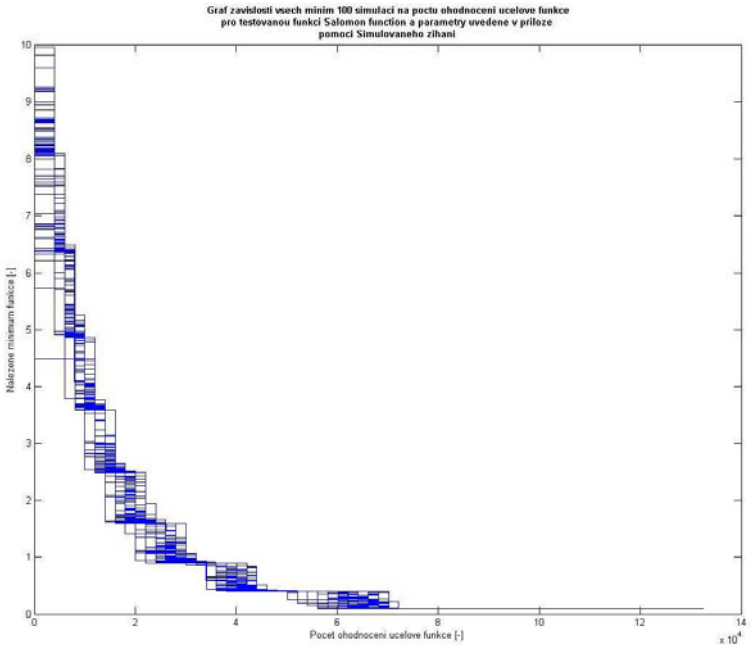
8-10 Graf kvality jednotlivých jedinců v závislosti na generacích, funkce Easom a DE

Funkce Salomon (obr. 8.11) má oblasti typu *n*-rozměrných kruhových ploch jako oblasti s lokálním extrémem. Již podle toho můžeme předvídat velkou možnost uváznutí v lokálním extrému. Na obr. 8.12 vidíme jak se algoritmus simulovaného žíhání (stejně DE a GA) na čas zachytí v oblastech lokálních extrémů a následně vývojem pokračuje dále..

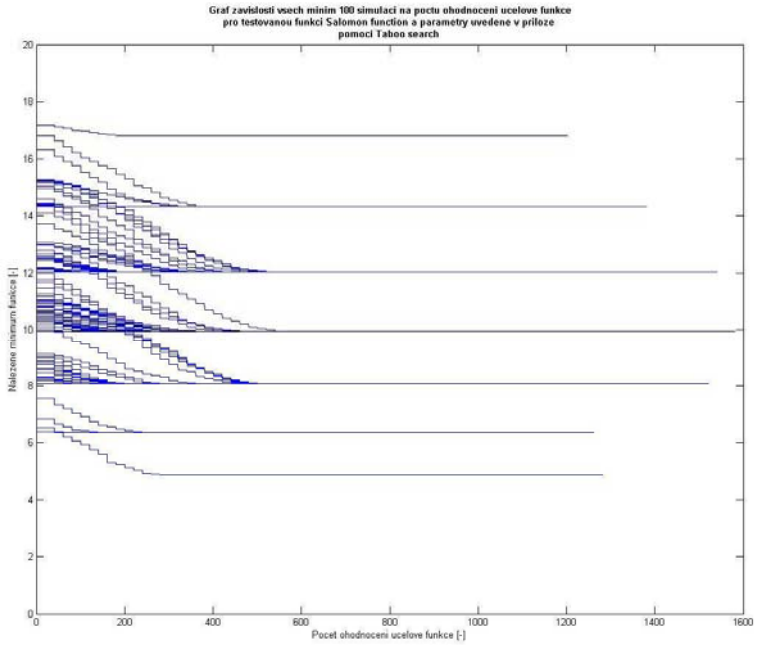
Stejně je tomu u průběhu jednoho cyklu, kdy vidíme kvalitu jednotlivých jedinců populace (obr. 8.14). Z obrázku lze také vypozorovat, že jedinci mají po určitou dobu stejné ohodnocení. To se dle implementace diferenciální evoluce může změnit až bude vytvořen lepší jedinec dle předpisu šlechtění. Také z tohoto průběhu vidíme, kdy se který jedinec zasekl v oblasti lokálních extrémů a jak rychle se mu jej podařilo opustit. Naopak taboo search na této funkci selhává. Náhodně zvolená počáteční pozice určuje pouze do kterého údolí lokálních extrémů se algoritmus dostane (obr. 8.13).



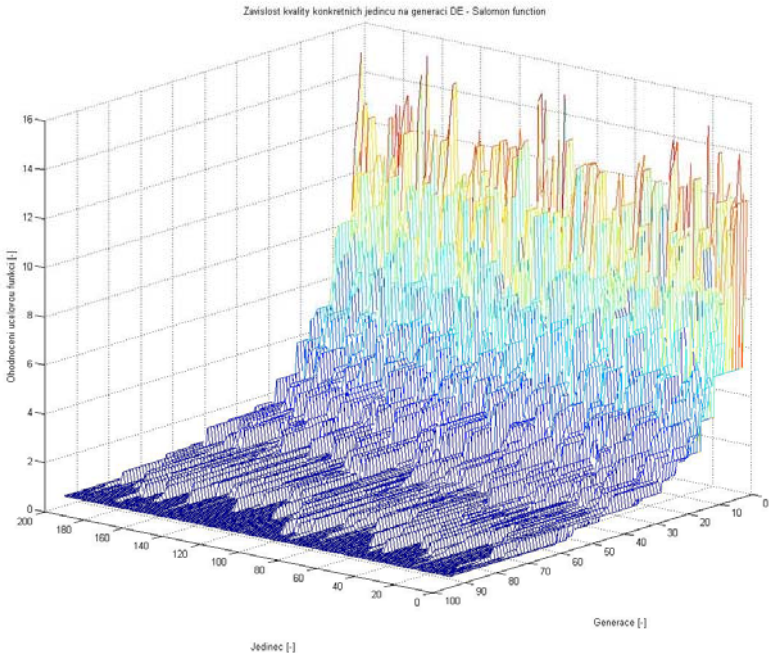
8-11 Funkce Salomon



8-12 Závislost procentuálních minim na počtu ohodnocení účelové funkce Salomon a SA



8-13 Závislost procentuálních minim na počtu ohodnocení účelové funkce Salomon a TS



8-14 Graf kvality jednotlivých jedinců v závislosti na generacích, funkce Salomon a DE

Složitosti také přináší vícenásobný výskyt stejného globálního extrému. Tuto skutečnost můžeme pozorovat např. u funkcí Rosenbrock saddle a Schwefel function.

8.3 TESTOVÁNÍ PRO OMEZENÝ POČET PŘÍSTUPŮ K ÚČELOVÉ FUNKCI NA GALERII TESTOVACÍCH FUNKCÍ

Praktické využití optimalizačních algoritmů je většinou spojeno s požadavky na čas výpočtu nebo na omezený přístup ke zdrojům. Z toho důvodu testování pro omezený přístup k účelové funkci vypovídá více o schopnostech optimalizačního algoritmu. Nalezené minimum za určitý čas nám ukazuje schopnost uplatnění algoritmu v časovém intervalu a také určuje rychlost konvergence k lepším hodnotám.

8.3.1 Formalizace problému omezeného počtu přístupu k účelové funkci

Představme si, že jsme postaveni před optimalizační problém, který musí být vyřešen (musí mít lepší výsledky než se stávajícím řešením a to řádově několikanásobně, aby bylo vynaložené úsilí doceněno). Řešení je třeba dosáhnout za časový interval dvou dnů. Časově nejnáročnější na výpočet je účelová funkce. Jedná se o velmi dlouhý a složitý výpočet, který na použitém výpočetním stroji trvá 8,64s. Oproti tomuto času jsou výpočty použitými algoritmy řádově v desítkách milisekund z čehož plyne,že je můžeme zanedbat.

Abychom dodrželi požadovanou lhůtu dvou dnů je maximální počet volání účelové funkce 20 000. Doba dvou dnů není naprosto striktní, proto můžeme počítat s odchylkou maximálně 2% což nám činí maximálně 20 400 přístupů k účelové funkci.

Takto nastavené podmínky budou pro testování pro omezený počet přístupů k účelové funkci směrodatné.

8.3.2 Vyhodnocení výsledků pro jednotlivé testovací funkce

Vyhodnocení bylo provedeno na základě procentuálního minima a na základě mediánu procentuálního minima. Vše je shrnuto v tabulce 8.4 a tabulce 8.5, kde nejsou zobrazeny konkrétní číselné výsledky (příloha č.1), ale pořadí algoritmů v pomyslné soutěži čtyř algoritmů. Celkové vyhodnocení je patrné ze součtu pořadí daného algoritmu pro určitou testovací funkci.

Tabulka 8-4 Výsledné pořadí algoritmů (minimum) pro omezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na minimální nalezené procentuální vzdálenosti od globálního minima pro omezený počet přístupů k účelové funkci (20000 přístupů s odchylkou max 2%)				
funkce	**DE**	**GA**	**SA**	**TS**
4th De Jong	3	4	2	1
1st De Jong	1	4	3	2
Rosenbrock saddle	2	4	3	1
3rd De Jong	1	4	3	2
Rastring	2	4	1	3
Schwefel	1	4	3	2
Grienwangks	2	4	3	1
Michalewicz	1	3	4	2
Easom	2	3	1	4
Salomon	1	3	1	4
Ackley	1	3	2	4
Součet pořadí	**17**	**40**	**26**	**26**

Tabulka 8.4 ukazuje velmi zajímavé výsledky i když se jedná o náhodou velmi zatížené kritérium vyhodnocení pomocí minimální procentuální vzdálenosti od globálního minima. Nejlépe dopadl algoritmus diferenciální evoluce následovaný shodným umístěním simulovaného žíhání a taboo search. Překvapivě poslední zůstává genetický algoritmus.

Tabulka 8.5 má naprosto stejné pořadí jako tabulka 8.4, akorát genetický algoritmus a diferenciální evoluce si polepšily v součtu pořadí. Naopak simulované žíhání a taboo search si mírně pohoršily, avšak stále mají totožné pořadí.

Celkový výsledek je dosti překvapivý. Srovnání genetického algoritmu a diferenciální evoluce vyšlo lépe pro diferenciální evoluci. Rozdílem oproti testování s neomezeným počtem přístupů k účelové funkci je pořadí simulovaného žíhání a taboo search. Tyto algoritmy nyní tvoří hranici mezi hlavními testovanými algoritmy. Jako zvláštnost v tomto testování je totální propad genetického algoritmu.

Tabulka 8-5 Výsledné pořadí algoritmů(medián) pro omezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na mediánu minimálních nalezených procentuálních vzdálenosti od globálního minima pro omezený počet přístupů k účelové funkci (20000 přístupů s odchylkou max 2%)				
funkce	**DE**	**GA**	**SA**	**TS**
4th De Jong	2	4	3	1
1st De Jong	1	4	3	2
Rosenbrock saddle	2	4	3	1
3rd De Jong	1	4	3	2
Rastring	2	3	1	4
Schwefel	1	4	3	2
Grienwangks	2	4	3	1
Michalewicz	1	2	4	3
Easom	1	3	2	4
Salomon	2	3	1	4
Ackley	1	3	2	4
Součet pořadí	**16**	**38**	**28**	**28**

8.3.3 Možné příčiny selhání genetického algoritmu při testování s omezeným počtem přístupu k účelové funkci

Jelikož je genetický algoritmus nepopsatelný analyticky a je ovlivněn náhodou, může být důvod např. ve špatném nastavení konstant algoritmu. Na druhou stranu již bylo provedeno testování s neomezeným počtem přístupů k účelové funkci a tam se stejné nastavení konstant osvědčilo. Tento názor má i opačnou stranu, neboť v prvním testování byla zastavovací podmínka 50 stejných řešení po sobě jdoucích generací (kol, sekvencí). Mnohdy algoritmy simulované žíhání a taboo search nedošly ani k pětině využití účelové funkce jaký měl genetický algoritmus. Z toho plyne, že buď nastavení genetického algoritmu je opravdu špatné (i když ze zkušenosti a po konzultaci s vedoucím bylo vybráno „dobré“ nastavení) nebo schopnost simulovaného žíhání a taboo search vymanit se z lokálního extrému nedokonalá.

Další z možností, která je logicky obhajitelná je nedokonalost Michalewiczovy dynamické mutace. Podíváme-li se na vzorce (3.4, 3.5) pozorně, zjistíme, že mutace je velkou měrou závislá na odhadu maximálního počtu generací. Jestliže bude předpokládaný maximální počet generací hodně velký oproti maximálnímu počtu generací, které opravdu dosáhneme (díky omezenému přístupu k účelové funkci), nedojde k řádnému prozkoumání (pomocí mutace) menších až velmi malých okolí jedinců. Tímto můžeme přijít o místa důležitá pro další křížení nebo spíše o konkrétní lokální či globální extrémy.

Třetí možnost je větší počet ohodnocení účelové funkce z důvodů vybraného způsobu křížení. Místo ohodnocení dvou potomků musí implementovaný genetický algoritmus ohodnotit tři potomky a následně vybrat dva nejlepší. Oproti diferenciální evoluci je genetický algoritmus v nevýhodě. Tato myšlenka je snadno napadnutelná názorem, že tímto způsobem křížení je prohledán větší prostor řešení a proto se o nevýhodu nejedná.

Z těchto tří důvodů je nejvíce pravděpodobné zachování velmi velkého maximálního počtu generací v dynamické Michalewiczově mutaci. Z průměrné hodnoty přístupu k účelové funkci během jedné generace byla vypočítána hodnota maximálního počtu generací během 20 400 přístupů k účelové funkci na 50. Test byl proveden se změněným parametrem maximální odhadované generace a jako výsledná hodnota byl brán medián z důvodu lepšího zhodnocení kvality algoritmu. Výsledná tabulka 8.6 ukazuje mírné zlepšení v celkovém výsledku. Jednotlivé výsledky (pro každou funkci) se ve většině případů velmi zlepšily jak lze vidět v příloze č.1, (tabulky pro změněné parametry lze porovnat s výsledky v tabulkách pro jednotlivé funkce).

Jelikož jsme experimentem ukázali, že vliv Michalewiczovy dynamické mutace má vliv na jednotlivé výsledky, ale na celkové výsledky působí jen nepatrně, zabývali jsme se možností změny parametru konstant genetického algoritmu. Konkrétně byly provedeny další čtyři testy s výsledkem mediánu. Jako měněné parametry byly *NP*, *KK*, *MK*.

Ani jeden z výsledku těchto čtyř testů nebyl lepší než předchozí test se stejnými parametry, ale změněným maximálním odhadovaným počtem generací na hodnotu 50. Přesné výsledky jsou v tabulkách pro změněné parametry v příloze č.1.

Nyní můžeme diskutovat vhodnost vybraných metod křížení, selekce, náhradové strategie a jiný způsob měnění konstant, protože jsme experimentálně ukázali, že nastavení maximálního počtu generací a námi zvolené obměny konstant neměly zásadní vliv na výsledky. Tato diskuze je kvůli své náročnosti vhodným námětem na další vědeckou práci.

Tabulka 8-6 Výsledné pořadí algoritmů (medián) pro změněnou maximální hodnotu počtu generací na 50

Tabulka pořadí algoritmů v závislosti na mediánu minimálních nalezených procentuálních vzdálenosti od globálního minima pro omezený počet přístupů k účelové funkci (20000 přístupů s odchylkou max 2%) a změněný maximální počet generací na 50 u Michalewiczovy mutace				
funkce	**DE**	**GA**	**SA**	**TS**
4th De Jong	2	4	3	1
1st De Jong	1	4	3	2
Rosenbrock saddle	2	4	3	1
3rd De Jong	1	4	3	2
Rastring	2	3	1	4
Schwefel	1	3	4	2
Grienwangks	2	4	3	1
Michalewicz	1	2	4	3
Easom	1	3	2	4
Salomon	3	1	1	4
Ackley	1	3	2	4
Součet pořadí	**17**	**35**	**29**	**28**

8.4 TESTOVÁNÍ NA PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO

Problém obchodního cestujícího byl rozebrán v kapitole 5. Rozmístění měst bylo vybráno náhodně, rovnoměrně do kruhu a rovnoměrně do tangenty. U náhodného a tangenciálního rozložení neznáme nejmenší vzdálenost mezi městy, proto můžeme porovnávat algoritmy pouze mezi sebou. Pro rozmístění rovnoměrně do kruhu známe nejkratší cestu mezi městy. Je to jednoduše obvod kruhu s přihlédnutím na to, že spojnice mezi městy jsou přímky. Z toho důvodu bude celková minimální délka trochu menší než obvod kruhu. Rozmístění měst je v kartézských souřadnicích pro x i y v rozmezí +10 až -10. Výpočtem dostáváme obvod kruhu 62,83 jednotek.

Popis tabulek i grafů v příloze č.1 je totožný jako v prvním testování pomocí galerie testovacích funkcí jak je uvedeno v kapitole 8.1.2 a 8.1.3. U tabulek se pouze nevyskytuje parametr vzdálenosti od globálního minima, ale nejkratší nalezená vzdálenost mezi městy. Mezi grafy je uveden pouze první typ grafů a to graf *závislosti procentuálních minim na počtu ohodnocení účelové funkce* a vždy jedno vybrané řešení problému obchodního cestujícího.

8.5 TESTOVÁNÍ PRO NEOMEZENÝ POČET PŘÍSTUPŮ K ÚČELOVÉ FUNKCI NA PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO

Testování pro neomezený počet přístupů k účelové funkci prověřuje přesnost algoritmu a schopnost dojít k řešení. Tento přístup využívá zastavovací podmínku 50 po sobě jdoucích stejných nejkratších řešení uspořádání měst. Druhou zastavovací podmínkou bylo dosažení 600 generace běhu algoritmu.

8.5.1 Vyhodnocení výsledků pro jednotlivé rozložení měst

Vyhodnocení bylo provedeno na základě minima a mediánu minim nalezení nejkratší cesty mezi městy. Vše je shrnuto v Tabulce 8.7 a Tabulce 8.8, kde nejsou zobrazeny konkrétní číselné výsledky (příloha č.1), ale pořadí algoritmů.

Tabulka 8-7 Výsledné pořadí algoritmů (minimum) pro neomezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na minimální nalezené vzdálenosti mezi městy pro neomezený počet přístupů k účelové funkci a v uvedených rozloženích měst		
Rozložení měst	**GA**	**DE**
Náhodně	1	2
Rovnoměrně - kruh	1	2
Rovnoměrně - tangens	1	2
Součet pořadí	**3**	**6**

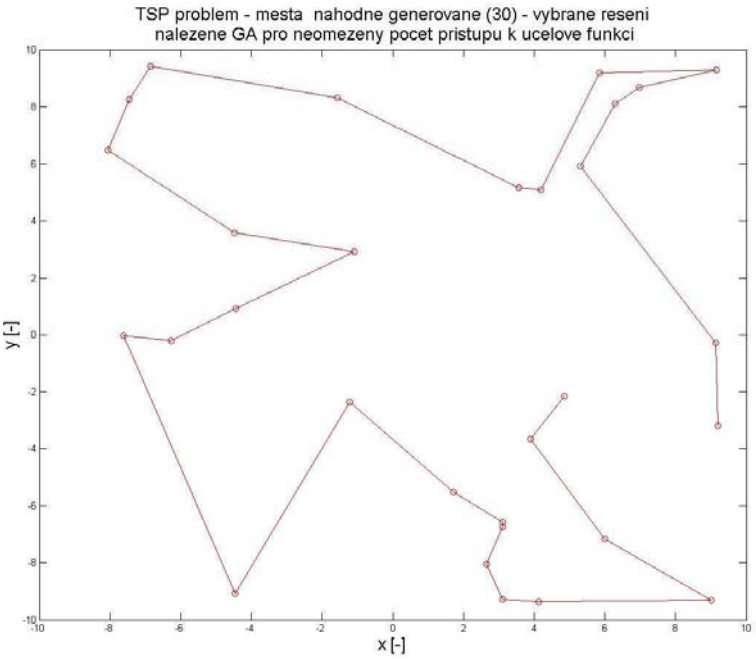
Celkové vyhodnocení je patrné ze součtu pořadí daného algoritmu pro všechny rozmístění měst.

Jak lze vidět z Tabulek 8-7 a 8-8 v případě úlohy o obchodním cestujícím pro neomezený počet přístupů k účelové funkci je výrazně lepší genetický algoritmus narozdíl od prvního testování na galerii testovacích funkcí. Výsledky byly různorodé, pro rovnoměrné rozložení měst do kruhu odhalil genetický algoritmus nejkratší cestu hned několikrát. Medián pro toto rozložení a genetický algoritmus byl 70,89 jednotek. Od nejkratší cesty se oproti diferenciální evoluci liší nepatrně. Ta v tomto testování selhala. Výsledky byly vždy téměř dvakrát horší než u genetického algoritmu (s výjimkou rozložení do tangenty) a to jak pro minimum tak pro medián minim. Hodnoty mediánu přístupů k účelové funkci byly téměř totožné až na případ náhodného rozložení měst, kdy medián vyhodnocení účelové funkce byl u genetického algoritmu 15818 a u diferenciální evoluce 7830, což je více než dvojnásobně.

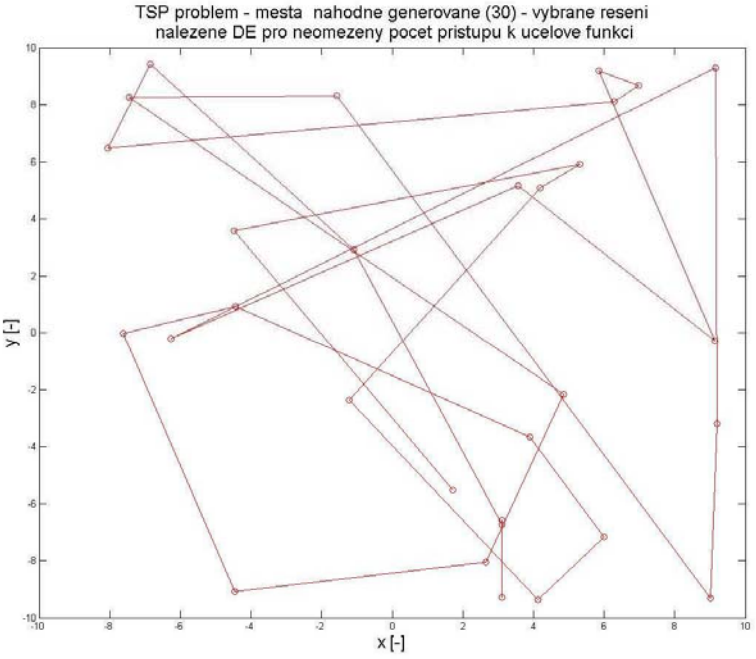
Tabulka 8-8 Výsledné pořadí algoritmů (medián) pro neomezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na mediánu minimální nalezené vzdálenosti mezi městy pro neomezený počet přístupů k účelové funkci a v uvedených rozloženích měst		
Rozložení měst	**GA**	**DE**
Náhodně	1	2
Rovnoměrně - kruh	1	2
Rovnoměrně - tangens	1	2
Součet pořadí	**3**	**6**

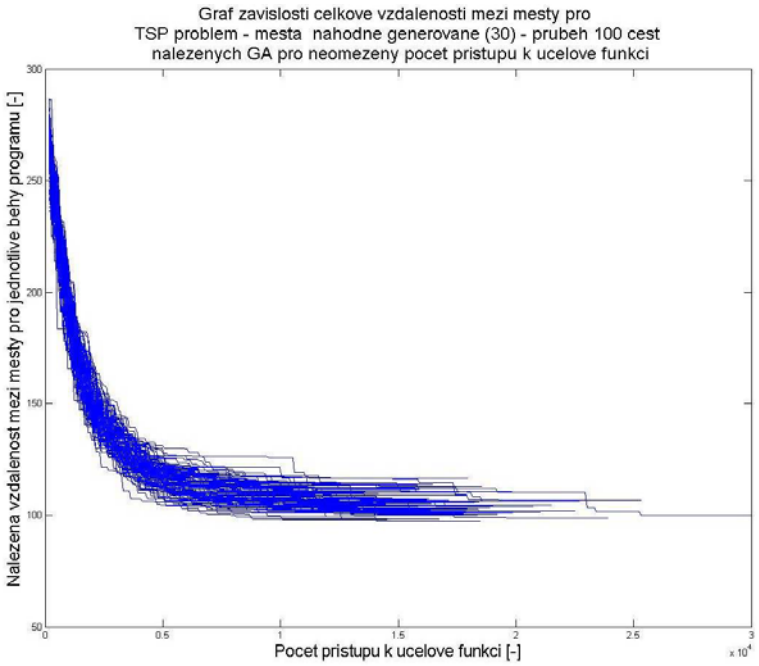
Z obrázků 8-15 a 8-16 lze vidět i pouhým okem značný rozdíl mezi celkovou vzdáleností mezi městy. Průběhy 8-17 a 8-18 ukazují velkou různorodost řešení pomocí diferenciální evoluce narozdíl od genetického algoritmu. Celé testování pro neomezený přístup k účelové funkci vyznělo jednoznačně lépe pro genetický algoritmus.



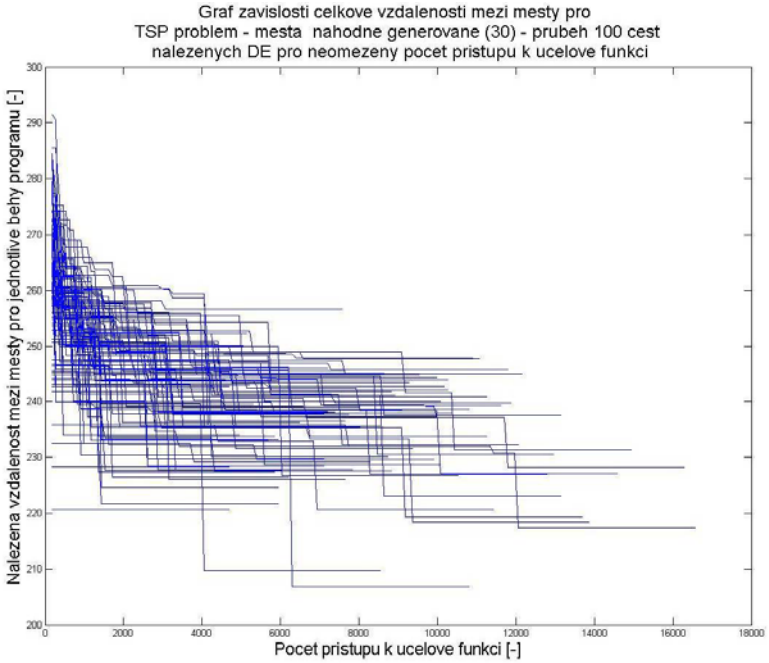
8-15 Vybrané řešení nalezené GA pro náhodné rozložení 30 měst a neomezený počet přístupů k účelové funkci



8-16 Vybrané řešení nalezené DE pro náhodné rozložení 30 měst a neomezený počet přístupů k účelové funkci



8-17 Závislost minimální cesty na počtu ohodnocení účelové funkce nalezené GA pro 30 měst a náhodné rozložení



8-18 Závislost minimální cesty na počtu ohodnocení účelové funkce nalezené DE pro 30 měst a náhodné rozložení

8.6 TESTOVÁNÍ PRO OMEZENÝ POČET PŘÍSTUPŮ K ÚČELOVÉ FUNKCI NA PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO

Na rozdíl od testování na galerii testovacích funkcí nebyla řešena formalizace problému s omezením přístupů, ale typ testování zůstal zachován a má simulovat omezení času pro plánování cesty mezi 30 městy. Počet přístupů k účelové funkci byl opět zvolen na maximálně 20 400.

8.6.1 Vyhodnocení výsledků pro jednotlivé rozložení měst

Vyhodnocení je provedeno stejně jako v prvním případě v kapitole 8.5.1 jen s tím rozdílem, že testování bylo provedeno pro omezený počet přístupů k účelové funkci.Veškeré tabulky a grafy jsou obsaženy v příloze č.1.

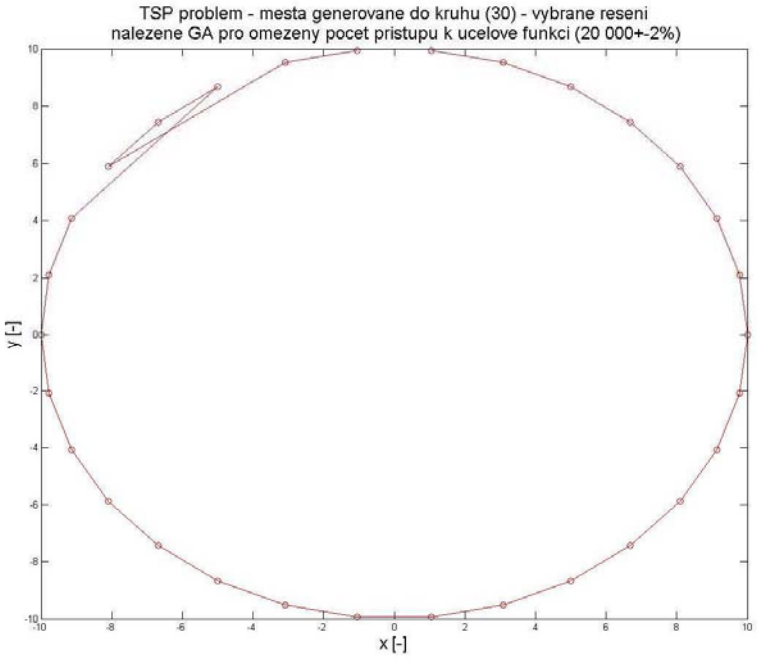
Tabulka 8-9 Výsledné pořadí algoritmů (minimum) pro omezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na minimální nalezené vzdálenosti mezi městy pro omezený počet přístupů k účelové funkci a v uvedených rozloženích měst		
Rozložení měst	**GA**	**DE**
Náhodně	1	2
Rovnoměrně - kruh	1	2
Rovnoměrně - tangens	1	2
Součet pořadí	**3**	**6**

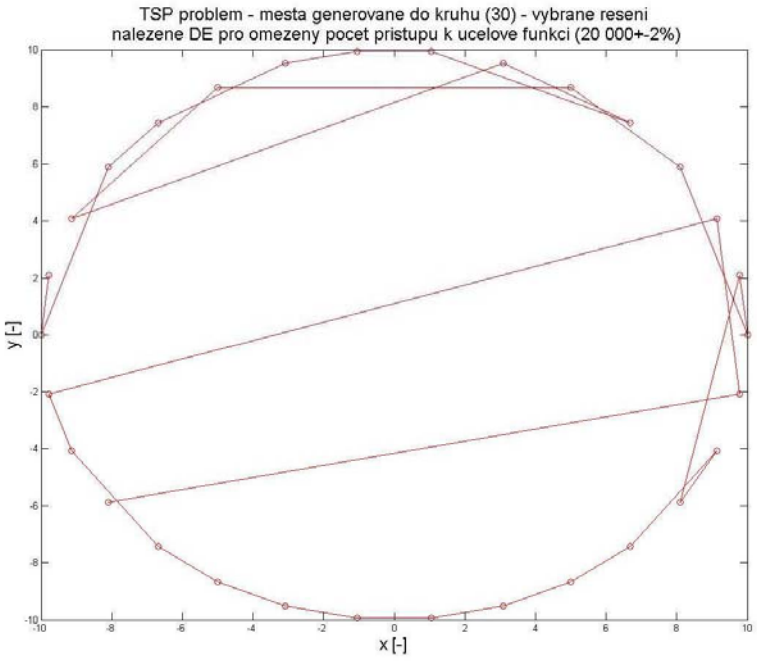
I v případě omezeného počtu přístupů k účelové funkci byl jednoznačně lepší genetický algoritmus. Medián jeho nalezených nejkratších cest byl opět téměř poloviční oproti diferenciální evoluci až na případ rozložení do tangenty. Zde měl medián genetického algoritmu hodnotu 136,75 a diferenciální evoluce 168,48. Ani v tomto případě nelze výsledné hodnoty srovnat s těmi nejkratšími, neboť nejkratší cesta je známa pouze pro rozložení do kruhu.

Tabulka 8-10 Výsledné pořadí algoritmů (medián) pro omezený počet přístupů k účelové funkci

Tabulka pořadí algoritmů v závislosti na mediánu minimální nalezené vzdálenosti mezi městy pro omezený počet přístupů k účelové funkci a v uvedených rozloženích měst		
Rozložení měst	**GA**	**DE**
Náhodně	1	2
Rovnoměrně - kruh	1	2
Rovnoměrně - tangens	1	2
Součet pořadí	**3**	**6**



8-19 Vybrané řešení nalezené GA pro rovnoměrné rozložení do kruhu 30 měst a omezený počet přístupů k účelové funkci



8-20 Vybrané řešení nalezené DE pro rovnoměrné rozložení do kruhu 30 měst a omezený počet přístupů k účelové funkci

8.7 MOŽNÉ DŮVODY SELHÁNÍ DIFERENCIÁLNÍ EVOLUCE NA PROBLÉMU OBCHODNÍHO CESTUJÍCÍHO

Jak je zmíněno v [3] metody použití genetického algoritmu na problému obchodního cestujícího jsou dosti propracované a prošly určitým vývojem. Namísto toho operátory pro diferenciální evoluci [9] spíše připomínají nové techniky řešení problému obchodního cestujícího než samotnou modifikaci tohoto algoritmu. Z toho důvodu je také v této práci zvolena technika *indexování pozicí*, která jako jediná připomíná klasickou diferenciální evoluci. Zdá se, že to je jeden ze stěžejních důvodů, proč diferenciální evoluce nedosahovala příliš uspokojivých výsledků. Možná dalším vývojem jejich technik nakonec předčí genetický algoritmus, ale v této práci je prokázáno, že k řešení toho problému se příliš nehodí.

9. ZÁVĚR

V teoretické části práce jsou shrnuty optimalizační algoritmy a podrobně vysvětleny principy čtyř optimalizačních algoritmů: genetického algoritmu, diferenciální evoluce, simulovaného žíhání a zakázaného prohledávání. Dále jsou popsány parametry jednotlivých algoritmů, jejich použití, možné modifikace a úskalí vybraných algoritmů.

Podrobně je vysvětlen postup testování optimalizačních algoritmů využitím galerie testovacích funkcí. V práci je konkrétní galerie sestavena z vybraných 11 testovacích funkcí. Funkce byly vybrány tak, aby v galerii byla obsažena co možná největší různorodost jejich vlastností. Mezi tyto vlastnosti patřila multimodalita, nelinearita, vícenásobné globální extrémy, celé oblasti lokálních extrémů či téměř konstantní funkce s velmi malým okolím globálního extrému.

V praktické části jsou zmíněné algoritmy testovány na galerii testovacích funkcí a na problému obchodního cestujícího vždy pro neomezený a omezený počet přístupů k účelové funkci.

Experimenty využívající galerie testovacích funkcí pro neomezený počet přístupů k účelové funkci vyzněly lépe pro diferenciální evoluci. Genetický algoritmus zaostával. Přidružené algoritmy měly ještě horší výsledky. Nejhůře dopadl algoritmus zakázaného prohledávání. Výsledky byly ovlivněny zastavovací podmínkou 50 stejných po sobě jdoucích minim. Tato podmínka mohla ukončit vykonávání programu dříve, což mohlo vést k ukončení programu v lokálním extrému. Výsledky vypovídají o efektivitě algoritmů pokud jsou časově neomezené a také jak rychle jsou algoritmy schopné vymanit se z lokálních extrémů (pokud jsou pomalé „pohltí“ je ukončovací podmínka). Na průbězích algoritmů v závislosti na počtu ohodnocení účelové funkce nebo na trojrozměrných průbězích kvality jedinců byly ukázány vybrané vlastnosti testovacích funkcí. Také bylo ukázáno jak si jednotlivé algoritmy s těmito vlastnostmi poradily.

Testování pomocí galerie testovacích funkcí pro neomezený počet přístupů vyznělo jednoznačně pro diferenciální evoluci. Simulované žíhání a zakázané prohledávání se shodně umístili na druhém místě a nejhůře dopadl genetický

algoritmus. Následně jsou experimentálně vyšetřovány možné příčiny selhání genetického algoritmu. Z toho také pramení mnoho námětů na další testování např. genetický algoritmus v závislosti na změnách parametrů, změnách metod selekce, křížení a mutace, testování na reálných datech a situacích namísto matematických funkcí.

Druhá skupina experimentů byla zaměřena na permutační úlohu obchodního cestujícího. Byla zvolena složitost 30 měst a ty byly rozmístěny náhodně, rovnoměrně do kruhu a rovnoměrně do tangenty. Testování probíhalo jak pro neomezený tak pro omezený počet přístupů k účelové funkci pro všechny tři typy rozmístění měst..

Výsledkem první části testování je poznatek, že algoritmus diferenciální evoluce je velmi robustní, schopný relativně rychle a efektivně řešit optimalizační problémy typu složitých funkcí. Navíc jeho implementace je velmi jednoduchá.. Dalším poznatkem je, že genetický algoritmus ve formě jaká byla zvolena, působí těžkopádně, potřebuje hodně využívat účelovou funkci a je složitější na implementaci (vztaženo k simulovanému žíhání a diferenciální evoluci).

Výsledek druhé části práce ukazuje na velmi dobrou schopnost řešit problém obchodního cestujícího pomocí genetického algoritmu a jeho speciálních operátorů. Tyto operátory jsou propracované a přinášejí dobré výsledky. Naproti tomu u diferenciální evoluce bylo praktickými výsledky poukázáno na nízkou kvalitu zatím vytvořených operátorů k řešení problému obchodního cestujícího, proto jí k řešení tohoto problému nelze doporučit a spíše nabádat k vytvoření nových, lepších technik.

Přidružené algoritmy simulovaného žíhání a zakázaného prohledávání obstály dobře. Každému algoritmu vyhovoval jiný typ funkce. Z toho plyne otázka jakým způsobem vybrat vhodný optimalizační algoritmus, aby rychle a efektivně přinesl lepší výsledky.

POUŽITÁ LITERATURA

[1] ZELINKA, I.: *Umělá inteligence v problémech globální optimalizace.* BEN – Technická literatura, Praha 2002, 189s.

[2] Kolektiv autorů/Wikipedia: *Optimalization Algorithms.* Dostupné na internetu: http://en.wikipedia.org/wiki/Category:Optimization_algorithms, aktualizováno dne 21.12.2008

[3] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: *Umělá inteligence (3), Kapitola 3 – Evoluční výpočetní techniky*. ACADEMIA, Praha 2001

[4] HONZÍK, P.: *Strojové učení, Kapitola 3 – Genetické algoritmy*. Skripta VUT Brno, Brno 2006

[5] HAUPT, R., HAUPT, S.: *Practical genetic algorithm*, Wiley-Interscience, New Jersey 2004

[6] Original homepage of DE http://www.icsi.berkeley.edu/~storn/code.html, aktualizováno dne 21.12.2008

[7] *Evoluční algoritmy a neuronové sítě*, dostupné z internetové stránky http://www.kiv.zcu.cz/studies/predmety/uir/texty/Chapter_09.pdf, aktualizováno dne 21.12.2008

[8] *GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with Matlab* - www.geatbx.com., http://www.geatbx.com/docu/fcnindex-01.html#P86_3059 , aktualizováno dne 21.12.2008

[9] PRICE, K., STORN, R., LAMPINEN, J.: *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin 2005 (1st edt.), ISBN:3540209506

SEZNAM PŘÍLOH