Vzorové testovací problémy v evolučních algoritmech

Bc. Pavel Novák

Bakalářská práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2005/2006

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Pavel NOVÁK**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Vzorové testovací problémy v evolučních algoritmech**

Zásady pro vypracování:

Cílem práce je systematicky shrnout současné testovací problémy z oblasti evolučních technik. Tyto metody by měli obsahovat jak triviální testovací problémy, tak problémy typu NP apod.:

1. vypracovat přehled problematiky testovacích problémů
2. vybrat vhodné již řešené příklady
3. vypracovat řešení pomocí některého z algoritmů SOMA, DE, SA a GA
4. provést závěr

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

[1] ZELINKA, Ivan, Umělá inteligence I. Volume 1. Zlín: Vutium, Brno, 1998. 126 p.ISBN 80-214-1163-5.
[2] ZELINKA, Ivan, Umělá inteligence / kap.6 "Diferenciální evoluce", Academia, 33p
[3] Kvasnička v., Pospíchal J., Tiňo p. 2000, Evolučné algoritmy, STU Bratislava, ISBN 85-246-2000, 2000
[4] ZELINKA, Ivan, New Optimitization Techniques in Engineering / kap.7 "SOMA -Self Organizing Migrating Algoritm",Springer-Verlag
[5] Koza J. R. 1998, Genetic Programing, MIT Press, ISBN 0-262-11189-6, 1998
[6] Koza J. R., Bennet F.H., Andre D., Keane M. 1999, Genetic Programing III, Morgan Kaufnamm pub., ISBN 1-55860-543-6, 1999

Vedoucí diplomové práce: **doc. Ing. Ivan Zelinka, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **14. února 2006**

Termín odevzdání diplomové práce: **26. května 2006**

Ve Zlíně dne 14. února 2006

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
①

prof. Ing. Vladimír Vašek, CSc.
pověřený děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato diplomová práce uvádí čtenáře do problematiky testovacích a optimalizačních problémů na jejichž úspěšné vyřešení se dají použít evoluční algoritmy. V teoretické části je probrána teorie složitosti a základní problémy a typy úloh které s ní souvisí. V praktické části jsou ukázány vzorové problémy. Je zde uveden jak jejich matematický zápis, tak algoritmy které byli použity k jejich řešení. Ve většině případů je ukázán i grafická vizualizace daného problému.

Klíčová slova: Testovací problémy, účelová funkce, optimalizační problémy, multikriteriální optimalizace

ABSTRACT

This diploma theme features reader to the basic benchmark and optimization problems on whose successful resolution give use evolution algorithms. In theoretic parts is examine theory complexity and basic problems and type exercise which with by bears. In practical parts are shown exemplary problems. Is here state how their mathematical entry, so algorithms which were used to their solving. In most cases is shown and graphic visualization given to problem.

Keywords: Test function, objective function, multi-objective function, multiobjective problem, globaly optimization

Tímto bych chtěl poděkovat vedoucímu mé diplomové práce doc. Ing. Ivanu Zelinkovi, Ph.D. , za vedení a cenné připomínky při zpracování jejího obsahu.

Ve Zlíně …….

Jméno diplomanta

OBSAH

ÚVOD

V dnešní době se velmi často objevuje problém vícekriteriálního rozhodování za omezené dostupnosti potřebných informací. Právě tyto typy problémů jsou však základním zdrojem dalšího růstu nových a lepších algoritmů.

Proto stále vzniká mnoho nových metod a postupů, které se snaží tento problém uspokojivě řešit. Jednou z těchto staronových metod je využití evolučních algoritmů. Starých proto, že příroda je úspěšně užívá pro řešení podobných problémů již několik miliard let a nových proto, že si jejich sílu uvědomujeme teprve teď.

Je velmi důležité si uvědomit, že nový algoritmus nemusí být vždy nejlepší, záleží ještě taky na typu problému. Proto je pro nás důležitější schopnost vyhledat řešení s možnost sledovat a řídit průběh tohoto hledání podle aktuálních podmínek, a k tomu jsou určeny testovací funkce nichž známe polohu globálního minima a potom se už jenom na ohodnocení účelové. Tento faktor nám říká jak je algoritmus na daný problém učinný.

I. TEORETICKÁ ČÁST

1 TEORIE SLOŽITOSTI

1.1 Optimalizační a rozhodovací úlohy

Řada úloh z reálného prostředí jsou tak zvané optimalizační úlohy. To znamená úlohy, kdy hledáme řešení, které je podle nějakého kritéria nejlepší (nejlevnější, nejdražší, nejkratší apod.). Teorie složitosti, a zejména pojmy P a NP jsou studovány pro tak zvané rozhodovací úlohy. Rozhodovací úloha je úloha, jejímž řešením je odpověď „ano“ nebo odpověď „ne“. Ke každé optimalizační úloze existuje její rozhodovací verze. Názorná ukázka je vidět na příkladu úlohy obchodního cestujícího. Úloha je abstrakcí tohoto problému. Obchodní cestující má navštívit všechna města $M_1,\ldots, M_n$ a vrátit se do výchozího města tak, aby ujel nejkratší možnou vzdálenost. Řešením je pořadí měst, ve kterém je obchodní cestující má navštívit.

1.1.1 Úloha obchodního cestujícího

Je dána množina měst $\{M_1,\ldots, M_n\}$. Pro každou dvojici měst M_i, M_j je dána přímá vzdálenost $d(M_i, M_j)$ měst M_i, M_j. **Trasa** (tour) je posloupnost měst $M_{\pi(1)}, M_{\pi(2)}, \ldots, M_{\pi(n)}$, kde π je permutace čísel $1,\ldots, n$. Délka této trasy je

$$\sum_{i=1}^{n-1} d(M_{\pi(i)}, M_{\pi(i+1)}) + d(M_{\pi(n)}, M_{\pi(1)})$$

Optimalizační verze: Najděte trasu nejmenší možné délky.

Rozhodovací verze: Je dáno navíc číslo K. Rozhodněte, zda existuje trasa délky menší nebo rovná K.

Na předchozím příklad je vidět, že optimalizační úloha hledá řešení, které je „nejlepší“, kdežto u rozhodovací verze je součástí instance ještě číslo K a otázka zní, zda existuje řešení, jež má hodnotu „ne horší“ než je dané číslo K.

Je zřejmé, že optimalizační verze úlohy není lehčí než její odpovídající rozhodovací verze. Najdeme-li „nejlepší“ řešení, víme, zda je aspoň tak dobré jako K. Na druhou stranu lze dokázat, že je-li rozhodovací verze řešitelná polynomiálně, pak je řešitelná polynomiálně i optimalizační verze. Je dobré si uvědomit, že i zde hraje roli fakt, že jsme si vybrali třídu všech polynomů. Z předchozího tvrzení je důležité, že skládáním, násobením a sčítáním polynomů dostaneme opět polynomy.

1.2 P-úlohy a NP-úlohy

1.2.1 P-úloha

P-úlohy jsou ve třídě všech rozhodovacích úloh *U*, pro něž existuje polynomiální algoritmus, který řeší *U*, tj. algoritmus, který má složitost $O(p(n))$ pro nějaký polynom $p(n)$.

Ne všechny rozhodovací úlohy jsou ve třídě P. Například pro rozhodovací verzi obchodního cestujícího není dosud znám polynomiální algoritmus, který by řešil tuto úlohu. Existuje algoritmus, nazývaný „algoritmus hrubé síly“, který úlohu řeší, ale stačí prohlédnout všechny trasy, tj. všechny permutace čísel $\{1,\ldots, n\}$ a pro každou spočítat její délku. Tím zjistíme, zda existuje trasa délky nejvýše *K*. Ovšem tento postup vyžaduje v nejhorším případě (totiž v případě, kdy jediná trasa kratší než *K* je ta poslední) čas $O(n!)$. Tím se tento algoritmus stává prakticky nepoužitelný již pro $n = 15$.

Naše úloha má však jednu zajímavou vlastnost: Jestliže pro instanci obchodního cestujícího existuje trasa délky nejvýše *K* a my ji náhodou odhadneme, pak na základě tohoto odhadu jsme velmi rychle schopni zdůvodnit, že správná odpověď pro danou instanci je „ano“. Neexistuje-li však pro naši instanci trasa délky nejvýše *K*, žádné „rychlé zdůvodnění“ nemusí existovat. NP-úlohy jsou právě úlohy, kdy odpověď „ano“ je možné (máme-li ještě nějakou dodatečnou informaci) rychle ověřit, ale odpověď „ne“ může být složitější.

Mějme rozhodovací úlohu *U*. Její instance rozdělíme do dvou tříd. ANO-instance jsou ty instance, na které je odpověď „ano“; NE-instance jsou instance, na které je odpověď „ne“. Zhruba řečeno, NP-úloha je taková rozhodovací úloha, pro kterou platí: pro každou ANO-instanci existuje objekt vhodného tvaru (nabízené řešení), pro který je v polynomiálním čase možné ověřit správnost odpovědi „ano“. Pro žádnou NE-instanci takový objekt neexistuje. Vlastnímu objektu se často říká „certifikát“ a proceduře, která objekt generuje, „oraculum“.

V případě úlohy obchodního cestujícího je tímto objektem trasa délky nejvýše *K*.

1.2.2 Np-úloha

NP-úloha je rozhodovací úloha, pro níž existuje nedeterministický algoritmus pracující v polynomiálním čase.

Do třídy NP patří ty úlohy, pro něž je lehké ověřit správnost řešení (v případě obchodního cestujícího to je trasa odpovídající délky). Navíc toto řešení musí mít nejvýše polynomiální velikost vzhledem k velikosti vstupu. Odpověď ne, tedy zjištění, že žádné takové řešení neexistuje, může být daleko obtížnější.

$$P \subseteq NP.$$

Každý deterministický algoritmus A můžeme považovat za nedeterministický algoritmus. První fázi buď vynecháme nebo vygenerujeme prázdný řetězec. Druhá fáze je pak vlastní deterministický algoritmus, který s řetězcem *s* vůbec nepracuje.

Otázka, zda P = NP je nejdůležitější dosud nezodpovězená otázka v teoretické informatice. Vzhledem k podstatě úloh, které jsou NP a pro něž není znám polynomiální algoritmus, se má za to, že rovnost neplatí. Ovšem zatím se to nikomu nepodařilo dokázat.

1.2.3 NP-úplné úlohy

Rozhodovací úloha *U* je NP-úplná (NP-complete), jestliže je NP-úlohou a každá NP-úloha se na *U* polynomiálně redukuje. Třídu všech NP-úplných úloh označujeme NPC. Jinými slovy, NP-úplné jsou ty úlohy, které jsou nejtěžší mezi všemi NP-úlohami.

Máme rozhodovací úlohy *U, V* a *W*. Jestliže úloha *U* se polynomiálně redukuje na úlohu *V* a úloha *V* se polynomiálně redukuje na úlohu *W*, pak se U polynomiálně redukuje na *W*, (tj. relace $\propto_p$ je transitivní).

Toto tvrzení vyplývá z faktu, že složení dvou polynomů je opět polynom. Přesněji: protože $U \propto_p V$, existuje polynomiální algoritmus *A*, který pro každou instanci *I* velikosti *n* zkonstruuje instanci *A(I)* úlohy V velikosti $q(n)$ pro nějaký polynom $q(n)$. Dále platí $V \propto_p W$, tj. existuje polynomiální algoritmus *B*, který pro instanci *A(I)* úlohy *V* zkonstruuje instanci *B(A(I))* úlohy *W* v čase $p(q(n))$ pro nějaký polynom $p(m)$. Navíc *I* je ANO-instance úlohy *U* právě tehdy, když *B(A(I))* je ANO-instance úlohy *W*.

Máme-li NP-úlohy *U* a *V*, takové, že $U \propto_p V$. Pak platí:

1) jestliže *U* je NP-úplná úloha, pak i *V* je NP-úplná úloha;
2) jestliže *V* je P-úloha, pak i *U* je P-úloha.

Do dnešní doby je známo více než dva tisíce NP-úplných úloh. My se zde omezíme na ty nejznámější z nich. U úloh, pro které existují i optimalizační verze, tyto verze uvedeme.

1.2.3.1 Splnitelnost Booleovských formulí (satisfiability)

Je dána formule v CNF (v konjunktivním normálním tvaru). Rozhodněte, zda je tato formule splnitelná, tj. rozhodněte, zda existuje pravdivostní ohodnocení logických proměnných tak, aby v něm byla daná formule pravdivá.

Splnitelnost Booleovských formulí, je NP-úplný problém, což dokázal Cook v roce 1971.

1.2.3.2 Barevnost

Je dán neorientovaný graf G s množinou vrcholů V a množinou hran E. Obarvení grafu G je zobrazení b: $V \rightarrow B$, kde B je konečná množina tzv. *barev*, takové, že pro každou hranu $e \in E$ jsou oba její krajní vrcholy obarveny různými barvami. (Jinými slovy $b(u) \neq b(v)$, kdykoli u a v jsou krajní vrcholy nějaké hrany $e \in E$.)

Řekneme, že graf G je k-barevný, jestliže jej lze obarvit k barvami (tj. množina B z definice má k prvků).

Optimalizační verze:

Je dán neorientovaný graf G. Určete nejmenší počet barev, kterými jej lze obarvit (tj. určete jeho barevnost).

Rozhodovací verze:

Je dán neorientovaný graf G a číslo $k \geq 3$. Rozhodněte, zda G je k-barevný.

1.2.3.3 Nezávislé množiny

Mějme neorientovaný graf G s množinou vrcholů V a množinou hran E. Množina vrcholů $N \subseteq V$ je nezávislá množina, jestliže žádná hrana z E nemá oba krajní vrcholy v množině N. Jinými slovy, jestliže úplný podgraf indukovaný množinou N neobsahuje žádnou hranu.

Optimalizační verze. Je dán neorientovaný graf G. Najděte nejpočetnější nezávislou množinu grafu G.

Rozhodovací verze. Je dán neorientovaný graf G a číslo $k \geq 3$. Rozhodněte, zda v G existuje nezávislá množina o aspoň k vrcholech.

1.2.3.4 Nejkratší cesty v grafech se zápornými cykly

Je dán orientovaný graf G s vrcholy V a hranami E, dále je dáno ohodnocení c hran celými čísly (tj. c: $E \rightarrow \mathbf{Z}$).

Optimalizační verze. Najděte nejkratší cesty mezi všemi dvojicemi vrcholů (délka cesty je rovna součtu ohodnocení všech hran, ze kterých se cesta skládá).

Rozhodovací verze. Je navíc dáno číslo k. Rozhodněte, zda mezi každými dvěma vrcholy existuje cesta délky nejvýše k.

1.2.3.5 Nejdelší cesty v grafech s cykly kladné délky

Úloha je obdobná jako v minulém případě, pouze hledáme cesty co nejdelší délky.

1.2.3.6 Existence hamiltonovské cesty, kružnice

Je dán neorientovaný graf G s vrcholy V a hranami E. Kružnice (cesta) v G se nazývá hamiltonovská, jestliže obsahuje všechny vrcholy (tj. všechny vrcholy právě jednou).

Je dán neorientovaný graf G. Rozhodněte, zda v něm existuje hamiltonovská kružnice (hamiltonovská cesta).

1.2.3.7 Celočíselné lineární programování

Jsou dána celá čísla a_{ij}, c_j a b_i, $1 \leq i \leq m$, $1 \leq j \leq n$.

Optimalizační verze: Najděte celá čísla $x_1,\ldots, x_n$, pro něž je lineární funkce největší

$$\sum_{j=1}^{n} c_j x_j$$

za podmínek

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \,, \quad 1 \leq i \leq m \,.$$

Rozhodovací verze: Navíc je dáno číslo *B*. Rozhodněte, zda existují celá čísla $x_1,\ldots, x_n$, splňující uvedené podmínky a pro něž je lineární funkce

$$\sum_{j=1}^{n} c_j x_j \leq B .$$

1.2.3.8 Problém batohu

Je dáno *n* předmětů $p_1,\ldots, p_n$, každý předmět má váhu $w(p_i)$ a cenu $c(p_i)$, pro $i = 1,\ldots, n$. Dále je dáno přirozené číslo *K*.

Optimalizační verze: Najděte podmnožinu předmětů tak, aby jejich váha nepřesáhla *K* a přitom jejich cena byla největší možná. (Jinými slovy, najděte podmnožinu $I \subseteq \{1,\ldots,n\}$ takovou, že je největší možná.)

$$\sum_{i\in I} w(p_i) \leq K \quad \text{a} \quad \sum_{i\in I} c(p_i)$$

Rozhodovací verze: Navíc je dáno číslo *C*. Rozhodněte, zda lze vybrat předměty tak, aby součet jejich vah nepřesáhl *K* a součet jejich cen byl alespoň *C*. (Jinými slovy, rozhodněte, zda existuje podmnožina $I \subseteq \{1,\ldots,n\}$ taková, že

$$\sum_{i\in I} w(p_i) \leq K \quad \text{a} \quad \sum_{i\in I} c(p_i) \geq C .)$$

1.2.3.9 Dělení kořisti

Je dáno *n* předmětů $p_1,\ldots, p_n$, každý předmět má cenu $c(p_i)$, pro $i = 1,\ldots, n$. Rozhodněte, zda lze rozdělit předměty na dvě skupiny tak, aby součet cen předmětů v obou skupinách byl stejný. Jinými slovy, rozhodněte, zda existuje podmnožina $I \subseteq \{1,\ldots,n\}$ taková, že

$$\sum_{i\in I} c(p_i) = \sum_{i\notin I} c(p_i) .$$

1.2.3.10 Třírozměrné párování

Jsou dány tři množiny K, L a M o n prvcích. Navíc je dána množina X trojic (k, l, m), kde $k \in K$, $l \in L$ a $m \in M$. Rozhodněte, zda existuje $Y \subseteq X$ taková, že žádné dvě trojice z Y se neshodují v žádné složce a Y obsahuje právě n trojic.

1.2.4 NP-těžké úlohy

Řekneme, že rozhodovací úloha U je NP-těžká (NP-hard), jestliže se na ni polynomiálně redukuje každá NP-úloha.

Z tranzitivity redukce úloh dostáváme, že NP-těžké jsou ty úlohy, na které se polynomiálně redukuje některá NP-úplná úloha.

To v jistém smyslu odpovídá představě, že NP-těžká úloha je „aspoň tak těžká jako všechny NP-úplné úlohy“. Kdybychom totiž uměli polynomiálně rychle vyřešit optimalizační verzi některé NP-úplné úlohy, uměli bychom polynomiálně vyřešit všechny NP úlohy. Ovšem náš pojem NP-těžké úlohy je širší, obsahuje i rozhodovací úlohy, které nejsou ve třídě NP. Uveďme příklad NP-těžké úlohy, která není ve třídě NP.

Je dán graf G a číslo L. Rozhodněte, zda existuje přesně L různých obarvení grafu G barvami $\{1,\ldots, k\}$, pro $k \geq 3$.

Je zřejmé, že pro tuto úlohu nemáme možnost jednoduše ověřit odpověď ano. I kdyby někdo vypsal L obarvení grafu G barvami $\{1,\ldots, k\}$, neměli bychom jistotu, že jiná obarvení neexistují. Na druhé straně, kdybychom měli polynomiální algoritmus pro tuto úlohu, uměli bychom polynomiálně řešit i úlohu, zda daný graf je trojbarevný. To by znamenalo, že P = NP.

1.3 Další třídy úloh podle složitosti

Stručně se zmíníme o dalším třídění úloh, pro něž existuje algoritmus, který je řeší, byť je třeba v praxi nepoužitelný. Vycházíme zde z předpokladu, že třídy P a NP nejsou stejné.

Jestliže P ≠ NP, pak žádná NP-úplná úloha není polynomiálně řešitelná. Můžeme se tedy ptát, zda existují NP-úlohy, které nejsou ani v P, ani nejsou NP-úplné. Jinými slovy, zda existují úlohy, které jsou svou obtížností „mezi“ P-úlohami a NP-úplnými úlohami.

Jestliže pro dvě rozhodovací úlohy $U \in P$ a $V \in N$ platí, že $U \propto_p V$, a neplatí, že $V \propto_p U$, pak existuje rozhodovací úloha W s vlastnostmi:

1) $U \propto_p W \propto_p V$;

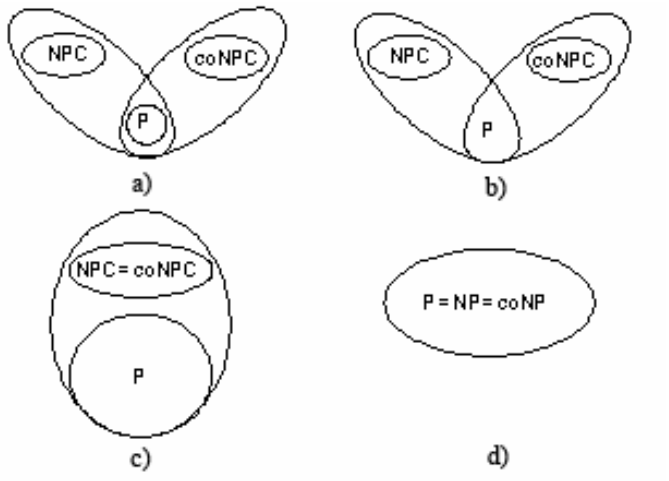
2)neplatí ani $W \propto_p U$ ani $V \propto_p W$.

Jestliže je tedy úloha U lehčí než úloha V, pak najdeme úlohu W, která je svojí obtížností mezi obtížnostmi úloh U a V.

Toto tvrzení má řadu důsledků. Například za předpokladu, že P ≠ NP, existuje mezi P-úlohami a NP-úplnými úlohami velmi bohatá škála obtížnosti úloh. Totiž, je-li úloha U ve třídě P a úloha V ve třídě NPC, pak existuje úloha U_1 obtížností mezi U a V. Dále existuje úloha U_2 obtížností mezi úlohami U a U_1, úloha U_3 obtížností mezi úlohami U a U_2 atd.

Podotkněme ovšem, že důkaz předchozího tvrzení se nezabývá úlohami „ze života“. Úloha se zde chápe jako jazyk nad danou abecedou (tj. jako množina slov dané abecedy) a v důkazu se vytváří jazyk W, který leží „mezi“ jazyky U a V. Proto i přes dokázaný fakt (i jeho důsledek) není jasné, zda existuje NP-úloha „ze života“, která by nebyla ani ve třídě P, ani ve třídě NPC.

Víme, že NP-úlohy jsou ty rozhodovací úlohy, pro které se snadno ověří odpověď „ano“, máme-li k dispozici dodatečnou informaci, tzv. certifikát. V případě odpovědi „ne“, může být situace obtížnější. U NP-úplných úloh se zatím nezná podstatně lepší způsob než vyzkoušení skoro všech možností. Zaměníme-li u NP-úloh odpověď „ano“ a „ne“, dostaneme tzv. *co*NP-úlohy.



Obr. 1. Ukázky množin a podmnožin různých tříd

Rozhodovací úloha *U* je ve třídě *co*NP, jestliže existuje NP-úloha *V* taková, že pro každou instanci *I* úlohy *U* platí:

I je ANO-instance úlohy *U* právě tehdy, když *I* je NE-instance úlohy *V*.

Rozhodovací úloha *U* je *co*NP–úplná (*co*NP–complete), zkráceně *co*NPC, jestliže je *co*NP a každá *co*NP–úloha se na ni polynomiálně redukuje.

Ztotožníme-li rozhodovací úlohu *U* s jazykem L_U nad nějakou abecedou, (tj. do jazyka L_U dáme každou ANO-instanci úlohy *U*, zakódovanou jako slovo nad vhodnou pevnou abecedou) pak můžeme předchozí definici přeformulovat takto: Jazyk L_U je jazyk *co*NP-úlohy právě tehdy, když jeho doplněk je jazyk NP-úlohy.

Ani zde se neví, zda platí *co*NP = NP, nebo zda to neplatí. Protože třída P je podmnožinou tříd *co*NP a NP, nastává jedna z možností znázorněných na Obr. 1. Přitom více se věří variantám a či b.

1.3.1 Třída P SPACE

Až doposud byli úlohy tříděny podle časové náročnosti algoritmů, které pro danou úlohu existují. Nyní se zaměřme i na paměťové nároky algoritmů řešících danou úlohu.

Rozhodovací úloha U je ve třídě P*SPACE*, jestliže existuje (deterministický) algoritmus, který řeší U a který má paměťovou složitost nejhoršího případu $O(p(n))$ pro nějaký polynom $p(n)$.

Lze dokázat, že algoritmus, který je ve třídě P, se dá naprogramovat tak, aby měl také polynomiální paměťové nároky. Jinými slovy platí

$$\mathrm{P} \subseteq \mathrm{P}\,SPACE.$$

Podobně jako jsme definovali třídu NP, lze definovat i třídu NP *SPACE*. Rozhodovací úloha patří do třídy NP *SPACE* právě tehdy, když pro ni existuje nedeterministický algoritmus, který ji řeší, a má polynomiální paměťovou složitost.

$$\mathrm{NP}\,SPACE = \mathrm{P}\,SPACE$$

Rovnost je jedna z mála rovností, kterou se podařilo dokázat.

Odtud dostáváme

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{P}\,SPACE.$$

A tady nelze rozhodnout, zda $\mathrm{NP} \neq \mathrm{P}\,SPACE$ či $\mathrm{NP} = \mathrm{P}\,SPACE$. Pouze druhá z rovností pokládá za nepravděpodobnou.

1.4 Algoritmicky neřešitelné úlohy

Všechny úlohy, se kterými jsme se až dosud setkali, byly řešitelné algoritmicky, ať lehce, či obtížně. Pro ty nejobtížnější existoval algoritmus (také zvaný *metoda hrubé síly*), který vyžadoval čas v nejhorším případě úměrný 2^n či $n!$. Je pravda, že v praxi je takovýto algoritmus téměř nepoužitelný, ale algoritmus existuje. Nyní ukážeme, že úlohy mohou být ještě obtížnější – tak obtížné, že pro ně žádný algoritmus, který by je řešil, neexistuje. Takové úlohy se označují jako algoritmicky neřešitelné nebo jako úlohy nerozhodnutelné[1].

2 TURINGŮV STROJ

2.1 Klasický Turingův stroj

Motivací pro Turingův stroj se stal v roce 1900 tzv. *Entscheidungsproblem* (rozhodovací problém). V něm německý matematik David Hilbert formuloval problém, ve kterém se ptal, zda existuje mechanický proces, kterým je možno rozhodnout o pravdivosti libovolného matematického teorému nebo výroku. Hilbert se chtěl přesvědčit, zda je například možné vyjádřit kroky matematického důkazu spíše posloupností symbolů, než složitým matematickým aparátem. Toho se chopil Turing a navrhl stroj, který měl simulovat postup matematika při vytváření důkazu. Takový stroj měl několik základních vlastností:

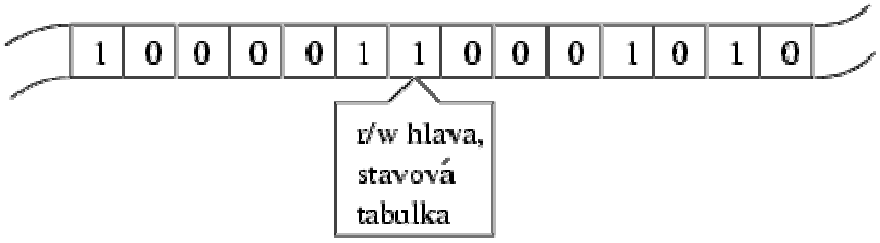
1) Musel nahradit složitou symboliku matematických kroků. V takovém případě šlo každou konečnou množinu symbolů nahradit pouze dvěma symboly (jako je 0 a 1) a prázdnou mezerou, která by oba symboly oddělovala.

2) Podobně jako si matematik zapisuje poznámky na papír, má Turingův stroj k zápisu nekonečnou pásku skládající se z buněk, do/ze kterých se symbol zapisuje/čte.

3) Nad touto páskou je možné provádět za pomocí čtecí hlavy operace čtení, zápisu a posunu pásky (read, write, shl, shr).

4) Protože je možné symboly číst, zapisovat nebo se posunovat po pásce, je pro Turingův stroj důležitý vnitřní stav, ve kterém operaci čtení provádíme (čtený symbol a stav tak určují další akci a přechod do dalšího stavu).

5) Protože se chování tohoto stroje vyvíjí podle tabulky přechodů, můžeme říci, že každý následující stav lze jednoznačně určit na základě čteného symbolu a aktuálního stavu. Jeho chování je proto *deterministické*.



Obr. 2. Model Turingova stroje

Výpočet začíná tak, že jsou na pásce uložena počáteční data (pokud jsou nějaká) a vlastní kód programu. Hlava je pak uvedena do stavu, který odpovídá načtení kódu programu a stroj tak započne výpočet, přechází mezi stavy a po skončení většinou zapíše výsledek. Vidíme, že takto se chovající model by se dal velmi dobře přirovnat k funkci dnešních počítačů. Přestože moderní technologie nevídaně od 30.let pokročily, Turingův model můžeme k popisu chování počítačů použít bez úprav i dnes. Pomocí Turingova stroje pak bylo dokázáno, že odpověď na počáteční Hilbertovu otázku je záporná. Neexistuje tak mechanický stroj, který by rozhodl pravdivost nebo nepravdivost libovolného matematického výroku.

2.2 Pravděpodobnostní Turingův stroj

Nyní si představme, že vývoj mezi stavy se řídí podle toho, jaký je výsledek nějakého náhodného jevu - například hodu kostkou. V takovém případě je následující stav určen *stochasticky* výsledkem hodu. Navíc s možností upřednostnit nebo potlačit některé směry vývoje pomocí váhových koeficientů. Na tomto základu je založen *pravděpodobnostní Turingův stroj (PTS).* PTS je schopen řešit všechny problémy deterministického Turingova stroje a často dokonce dojde k řešení rychleji. U obou typů strojů vlastně balancujeme mezi dvěma póly: jistota, že dojdeme k řešení (pokud existuje) u klasického Turingova stroje stojí proti možnosti, že najdeme správné řešení rychleji u PTS. Přestože je Turingův stroj matematickým modelem, nezbavil se všech zátěží klasické fyziky a jako takový vychází z jejích poznatků. Se znalostmi kvantové mechaniky však bylo možné ideu PTS upravit a definovat tak model, který popsal proces kvantového výpočtu.

2.3 Kvantový Turingův stroj

Hlavním impulzem pro vznik *kvantového Turingova stroje (KTS)*, bylo v roce 1973 potvrzení Charlese Bennetta, který dokázal jeho reverzibilitu. To nezůstalo bez povšimnutí Paulem Benioffem, kterého napadlo, že by šlo napodobit vývoj reverzibilního kvantového systému na Turingově stroji. Na práce Benioffa a později i Richarda Feynmana navázal prvním popisem opravdového KTS v roce 1985 David Deutsch. Kvantová verze měla tyto hlavní charakteristiky a odlišnosti od klasického stroje:

1) Čtení, zápis a posun pásky se odehrával pomocí kvantově-mechanických interakcí.

2) Místo čisté 0, 1 a mezery se nyní setkáváme v každé buňce pásky se superpozicí stavů 0 a 1, což si můžeme představit jako vektor v jednotkové kouli; odklon od svislé osy zde určuje podíl zastoupení 0 a 1.

3) Superpozice zde umožňuje využít kvantový paralelismus - možnost provádět jednu operaci nad více daty současně.

Nejlépe si lze KTS představit jako kvantové zobecnění PTS. Pokud necháme po určitou dobu takový KTS vyvíjet (nebudeme jej měřit), můžeme stav vyjádřit jako součet pravděpodobností výpočetních cest, kterými může stroj projít. U KTS se v jednom kroku nebere pouze jedna náhodně zvolená cesta (jeden následující stav, jako je tomu u PTS), ale výpočet pokračuje v duchu kvantového paralelismu všemi možnými cestami naráz. Se vznikem KTS bylo možné zaměřit úsilí na otázky vypočitatelnosti, složitosti a univerzálnosti kvantových počítačů. Vypočitatelnost je spojena s otázkou, zda je možné o daném problému rozhodnout (nebo nalézt k němu řešení) v konečném čase. Pokud algoritmus, terý by těmto požadavkům vyhověl, neexistuje, nazývá se problém nevypočitatelný. Hledání algoritmu k řešení problému vede i k tomu, že například pro generování náhodného čísla žádný klasický algoritmus neexistuje (není možné vygenerovat opravdu náhodné číslo, protože všechny klasické generátory musí být založeny na výpočtu vstupů dle daného předpisu), kdežto u kvantových počítačů takový algoritmus existuje. Vychází totiž ze základní vlastnosti přírody - okamžiku náhodného kolapsu vlnové funkce na vlastní stav při měření kvantového systému. Složitost neboli komplexita je druhou vlastností, která nás u kvantových počítačů zajímá. Souvisí s ní efektivita kvantových algoritmů a následně i vý-

kon kvantových počítačů. Aby se dokázalo, že výzkum kvantových počítačů může mít v budoucnosti i praktický dopad, začala se během 90.let vynořovat řada možných aplikací, které využívají superpozice kvantových systémů k uplatnění kvantového paralelismu a tím i vylepšení složitostí dosavadních algoritmů. Jako příklad uveďme známý problém s faktorizací velkých celých čísel na součin prvočísel. Nejlepší známý klasický algoritmus dosahuje exponenciální složitosti, kterou lze vyjádřit jako:

$$O\left(e^{\left(L\frac{1}{3}\ln\frac{2}{3}L\right)}\right)$$

Kde $L=ln\ n$ a n je celé číslo, jehož rozklad hledáme. Vidíme, že časová náročnost roste velmi rychle s velikostí vstupu. Tento problém je ze skupiny NP problémů, u nichž existuje efektivní nedeterministický algoritmus, jehož řešení v polynomiálním čase ověřit lze. Řešení takového problému deterministickým algoritmem není kvůli počtu výpočetních kroků schůdné. Nicméně u faktorizačního problému se zatím nepodařilo dokázat, že nemá efektivní algoritmus běžící v třídě složitosti P, tj. v polynomiálním čase, a proto je možné, že bude takový algoritmus ještě objeven. V roce 1994 se Peteru Shorovi z AT&T Bell Labs podařilo vymyslet kvantový faktorizační algoritmus, jehož složitost je na kvantovém počítači polynomiální. Takové řádové vylepšení znamenalo průlom ve vývoji kvantových algoritmů. Podobně jako se dají roztřídit algoritmy u klasické a pravděpodobnostní verze Turingových strojů, lze definovat i kvantové složitostní třídy. Pro přehlednost byly všechny třídy seřazeny do následující tabulky:

Tab. 1. Tabulky tříd problémů

Třída	Popis
P	schůdné algoritmy běžící nejhůře v polynomiálním čase (dále jen PT), příklad: násobení
NP	neschůdné algoritmy; pouze správnost řešení lze ověřit v PT, příklad: faktorizace
NP-úplný	NP problémy vzájemně mapovatelné v PT, příklady: SAT, obchodní cestující, plánování
ZPP	problémy řešitelné s jistotou v průměrně PT na PTS
BPP	problémy řešitelné s p > 2/3 v nejhůře PT na PTS
QP	problémy řešitelné s jistotou v nejhůře PT
ZQP	problémy řešitelné s jistotou v průměrném PT
BQP	problémy řešitelné s p > 2/3 v nejhůře PT

Univerzálnost je schopnost efektivně simulovat jeden Turingův stroj druhým. Turinga napadlo, že když vytvoří transformační pravidla jednoho Turingova stroje jako program (posloupnost bitů) pro jiný stroj, bude tento stroj schopný simulovat první stroj. Tak vznikla myšlenka, že je možné vytvořit programovatelný počítač - Turingův stroj, jehož programem je nějaký algoritmus. Až v roce 1996 potvrdili Seth Lloyd a Christof Zalka, že univerzálnost platí také u kvantových počítačů. To znamená, že jeden kvantový systém dokáže simulovat jiný[2].

3 PARETO OPTIMUM

3.1 Charakteristika multikriteriálního optimalizačního problému

Optimalizační multikriteriální problém je popsán dvěma nebo více učelovými funkcemi a má za úkol nalézt minimum každé zadané funkce.

$$F_1(\{x_i\}), F_2(\{x_i\}), \ldots\ldots, F_m(\{x_i\})$$

Optimální řešení odpovídající nezávislým optimalizacím jednotlivých účelových funkcí jsou zpravidla rozdílná a často bývají ve vajemném protikladu. Obecné řešení multikriteriálního problému je tzv. Pareto Množina optimálních řešení.

3.2 Matematická formulace multikriteriálního problému

Máme najít minimum funkce:

$$\min f(\{x_i\})$$

kde

$$f(\{x_i\}) = [F_1(\{x_i\}), F_2(\{x_i\}), \ldots\ldots, F_m(\{x_i\})]^T$$

při splnění následujících podmínek kde

$$g_j(\{x_i\}) \leq 0$$

znamená přípustnou oblast

$$h_k(\{x_i\}) = 0$$

vazbové podmínky

$$x_{i,min} \leq x_i \leq x_{i,max}$$

přirozené podmínky

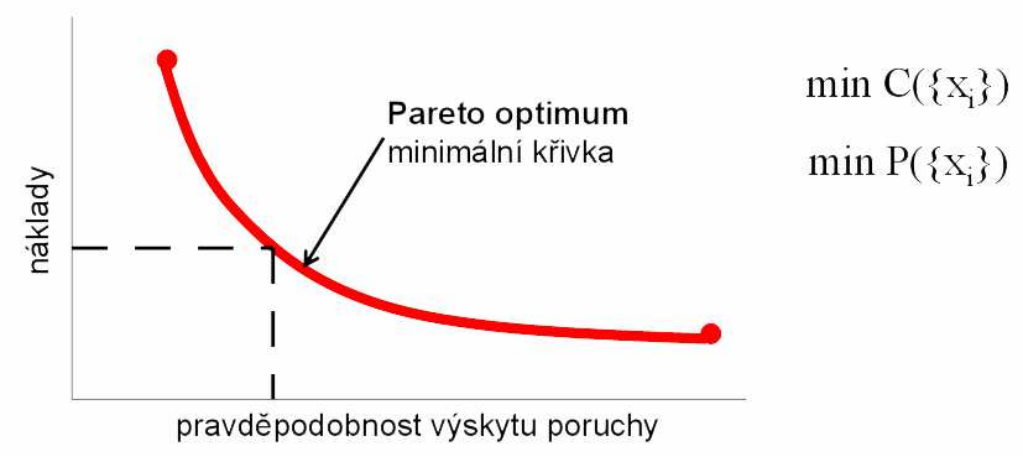
Cílem multikriteriální optimalizace $\min f(\{x_i\})$ je nalézt minimální křivku která se nazývá Pareto Optimum. Nalezení se děje v prostoru všech účelových funkcí a odpovídající hodnoty optimalizačních proměnných v prostoru přípustné oblasti.

Pareto optimum je silně nedominantní řešení, tzn. Že žádná z účelových funkcí se nemůže zmenšovat bez zapříčinění současného zvýšení nejméně jedné další účelové funkce.

Jestliže ale složky vektoru účelových funkcí f($\{x_i\}$) nejsou plně nezávislé, potom zpravidla neexistuje j

Pareto optimum – množina optimálních řešení

Zde uvedený příklad je pro dvě učelové fukce a mají minimalizovat náklady a pravděpodobnost výskytu poruchy.



Obr. 3. Minimalizace nákladů a pravděpodobnosti výskytu poruchy

Konce minimální křivky představují dvě absolutní minima obou účelových funkcí uvažovaných samostatně[3].

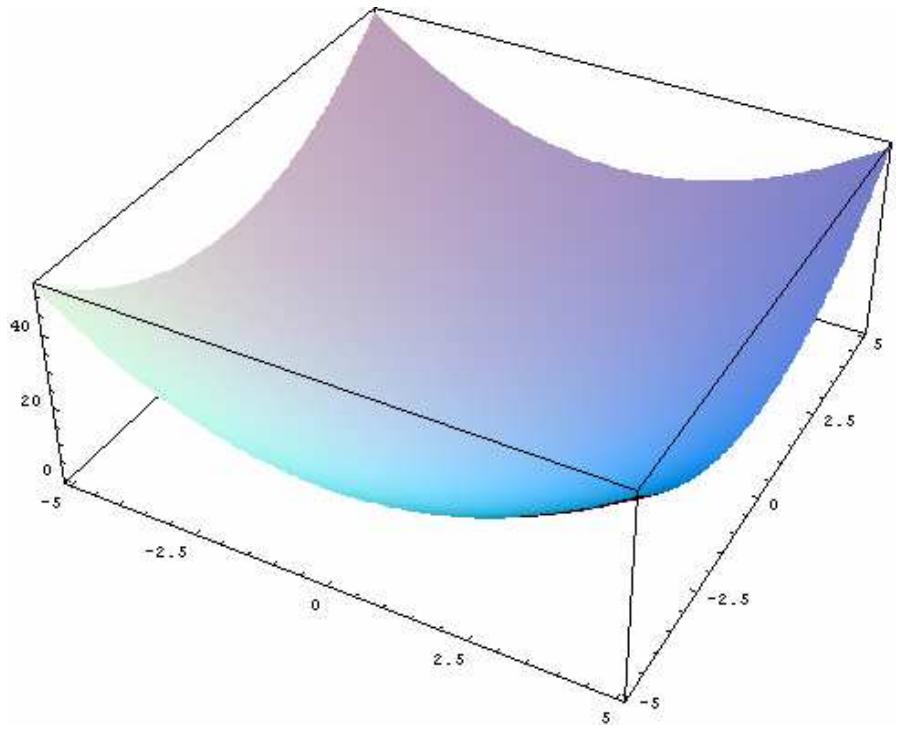
PRAKTICKÁ ČÁST

4 TESTOVACÍ PROBLÉMY

4.1 De Jong's function 1

Matematický zápis:

$$f(x) = \sum_{i=1}^{n} x_i^2$$
$$-5{,}12 \le x_i \le 5{,}12$$
$$n = 1,\ldots,n$$



Obr. 4. 3D-zobrazení De Jong's function 1, n = 2

Globální minimum f (x_1,x_2)= 0, souřadnice [0,0], při dimenzi n = 2.

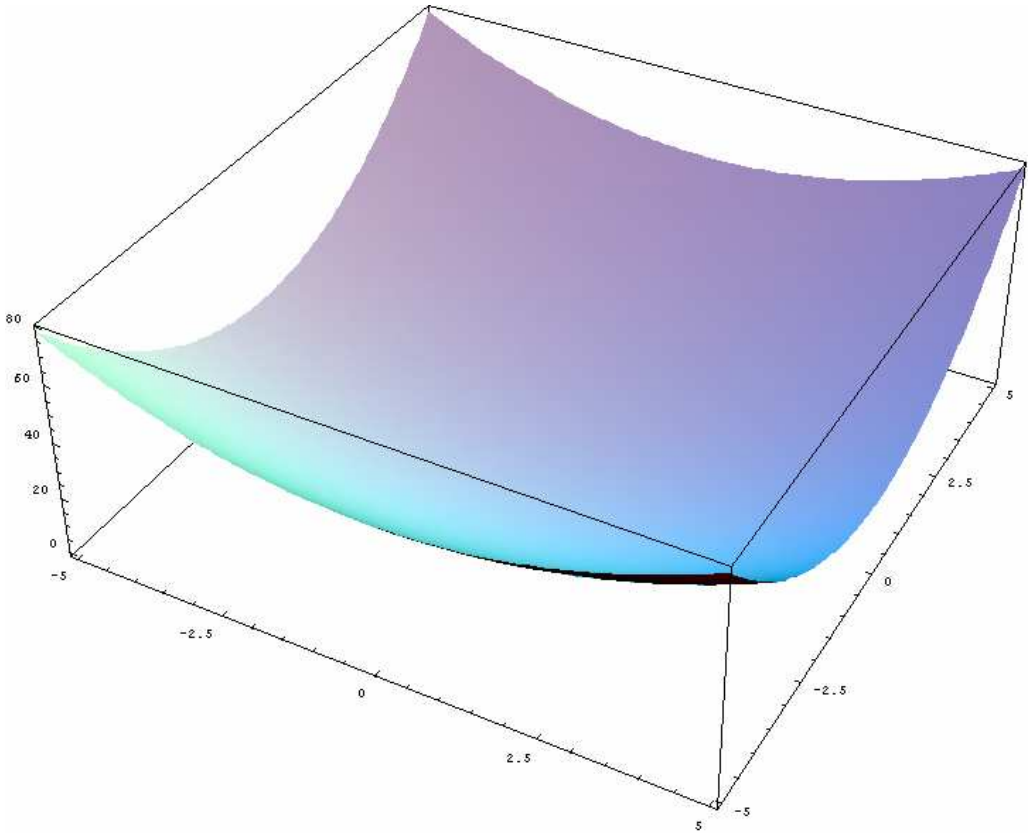
Na řešení této testovací funkce byli použity následující algoritmy:

FDR-PSO(112)(fitness-distance ratio – particle swarm optimization), FDR-PSO(111), FDR-PSO(102), FDR-PSO(012), FDR-PSO(002) [4], [6].

4.2 Axis parallel hyper-ellipsoid

Matematický zápis:

$$f(x) = \sum_{i=1}^{n} i \times x_i^2$$
$$-5{,}12 \le x_i \le 5{,}12$$
$$n = 1,\ldots,n$$



Obr. 5. 3D-zobrazení funkce Axis parallel hyper-ellipsoid, n = 2

Globální minimum f (x_1,x_2)= 0, souřadnice [0,0], při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

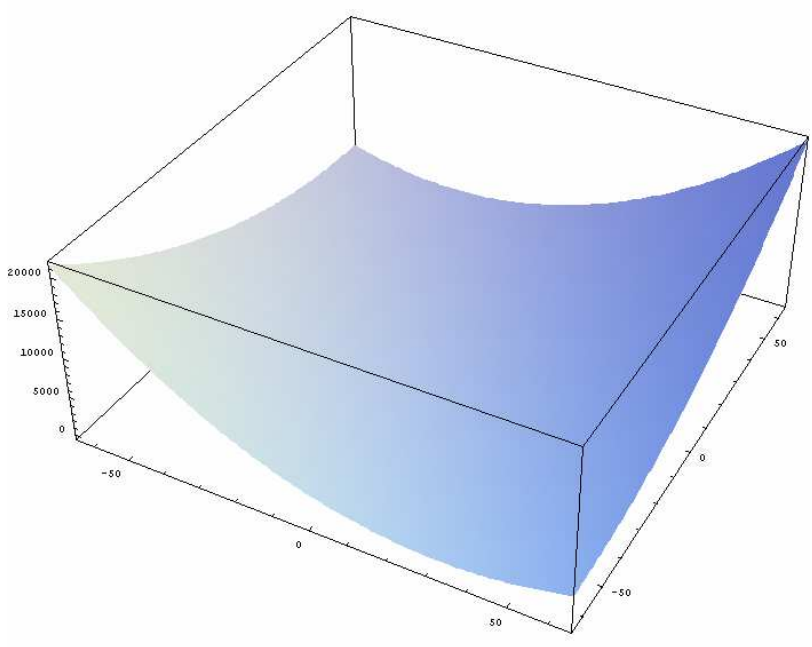
FDR-PSO(112)(fitness-distance ratio – particle swarm optimization), FDR-PSO(111), FDR-PSO(102), FDR-PSO(012), FDR-PSO(002) [4], [6].

4.3 Rotated hyper-ellipsoid

Matematický zápis:

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$$

$$-65{,}536 \leq x_i \leq 65{,}536$$



Obr. 6. 3D-zobrazení funkce Rotated hyper-ellipsoid, n = 2

Globální minimum f (x_1,x_2)= 0, souřadnice [0,0], při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

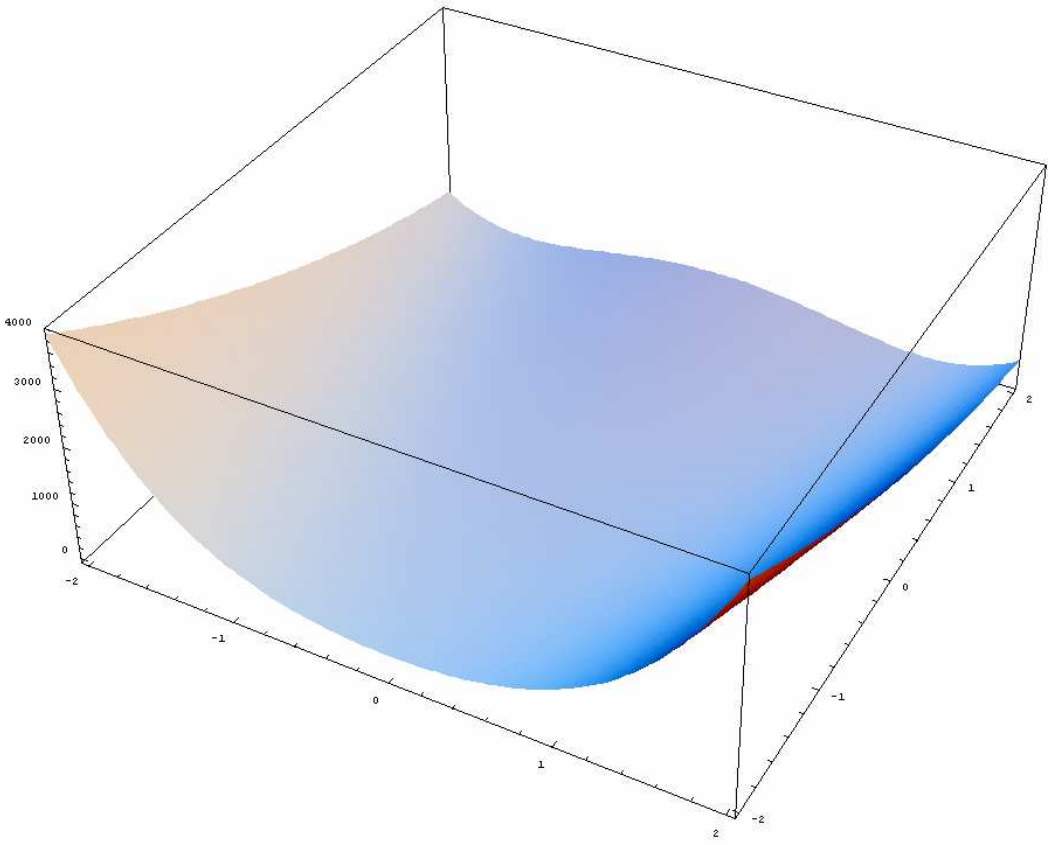
FDR-PSO(112)(fitness-distance ratio – particle swarm optimization), FDR-PSO(111), FDR-PSO(102), FDR-PSO(012), FDR-PSO(002) [4], [6].

4.4 Rosenbrock's Valley(Banana function)

Matematický zápis:

$$f(x) = \sum_{i=1}^{n} 100 \times \left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2$$

$$-2{,}048 \leq x_i \leq 2{,}048$$



Obr. 73D-zobrazení funkce Rosenbrock's Valley, n = 2

Globální minimum f (x_1,x_2)= 0, souřadnice [1,1], při dimenzi n = 2

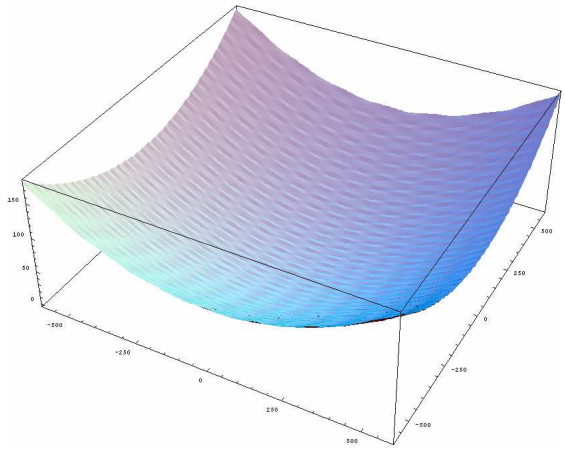
Na řešení této testovací funkce byli použity následující algoritmy:

FDR-PSO(112)(fitness-distance ratio – particle swarm optimization), FDR-PSO(111), FDR-PSO(102), FDR-PSO(012), FDR-PSO(002) [4], [6].

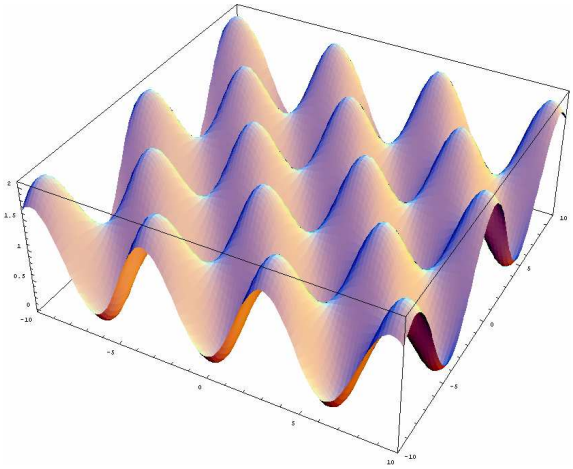
4.5 Griewangk's function

Matematický zápis:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
$$-600 \le x_i \le 600$$
$$i = 1,\ldots,n$$



Obr. 8. 3D-zobrazení funkce Griewangk's function, n = 2



Obr. 9. Výřez průběhu Griewangk's function

Globální minimum f (x_1,x_2)= 0, souřadnice [0,0], při dimenzi n = 2

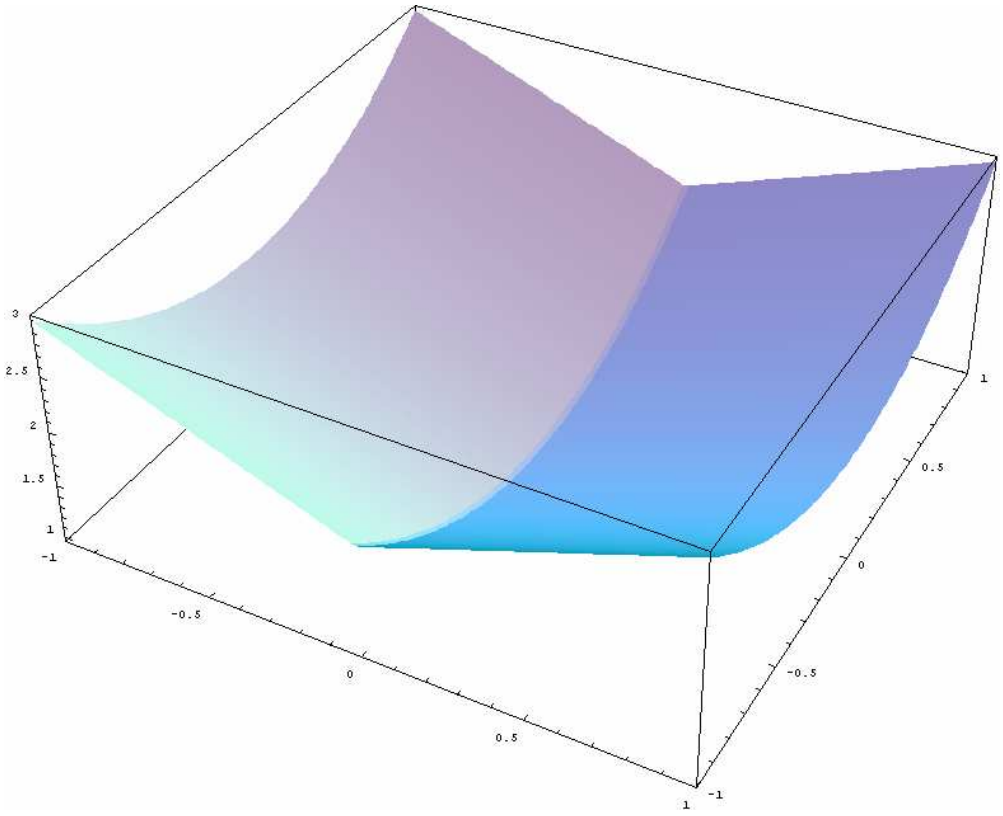
Na řešení této testovací funkce byli použity následující algoritmy:

FDR-PSO(112)(fitness-distance ratio – particle swarm optimization), FDR-PSO(111), FDR-PSO(102), FDR-PSO(012), FDR-PSO(002) [4], [6], [7].

4.6 Sum of different powers

Matematický zápis:

$$f(x) = \sum_{i=1}^{n} |x_i|^{i+1}$$
$$-1 \le x_i \le 1$$
$$i = 1,...,n$$



Obr. 10. 3D-zobrazení Sum of different powers, n = 2

Globální minimum f (x_1,x_2)= 0, souřadnice [0,0], při dimenzi n = 2

Na řešení této testovací funkce byli použity následující algoritmy:

FDR-PSO(112)(fitness-distance ratio – particle swarm optimization), FDR-PSO(111), FDR-PSO(102), FDR-PSO(012), FDR-PSO(002)[4].

4.7 Shekel Function

Matematický zápis:

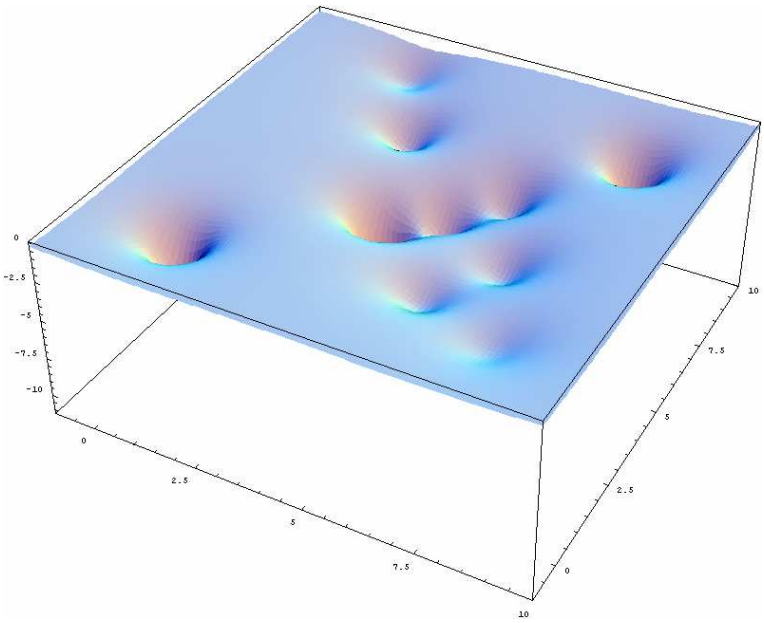
$$f(x) = \sum_{i=1}^{m} \frac{1}{c_i + \sum_{j=1}^{n} (x_j - a_{ji})^2}$$

$$c = [1 \quad 2 \quad 2 \quad 4 \quad 4 \quad 6 \quad 3 \quad 7 \quad 5 \quad 5]$$

$$a = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 5 & 1 & 2 & 3{,}6 \\ 4 & 1 & 8 & 6 & 3 & 2 & 3 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3{,}6 \end{bmatrix}$$

$$0 \le x_j \le 10,\ j = 1,2,3,4$$

$$i = 1,2,...,10$$



Obr. 11. 3D-zobrazení Shekel function, n = 2

Globální minimum f (x_1,x_2)= -10,5364 pro m = 10, souřadnice [4,4], při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

Continuous Genetic Algorithm (CGA), GA for Globally Minimizing Function (GAGMF), Enhanced Continuous Tabu Search (ECTS), Enhanced Simulated Annealing (ESA), Continuous Reactive Tabu Search (CRTS min) minimum, Continuous Reactive Tabu Search (CRTS ave) average, Taboo Search (TS), INTEROPT [8].

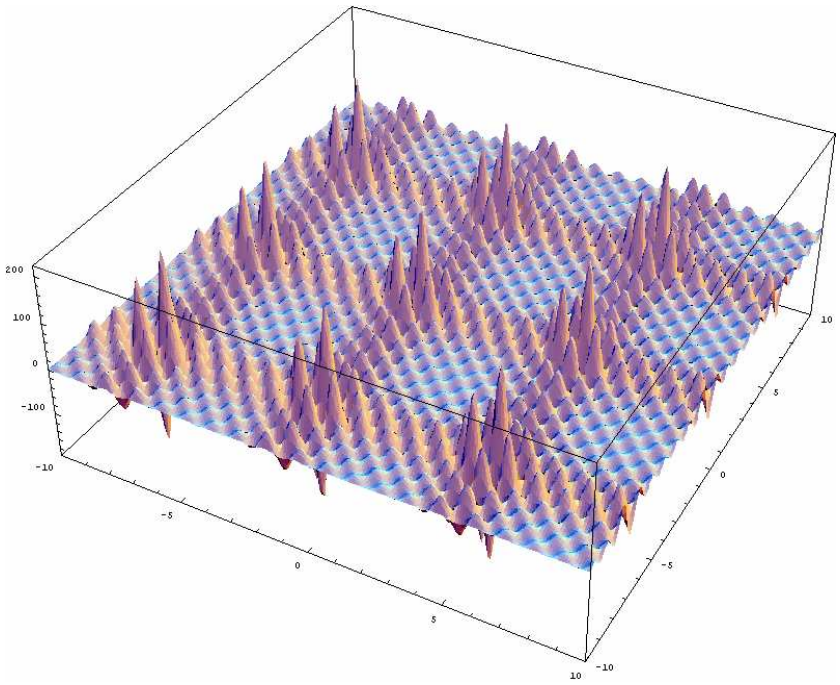
4.8 Levy function

4.8.1 Levy function No. 3

Matematický zápis:

$$f(x) = \sum_{i=1}^{m} i\cos\left((i+1)x_1 + i\right)\sum_{j=1}^{m} j\cos\left((j+1)x_2 + j\right)$$

$$-10 \le x_i \le 10, i = 1,2$$



Obr. 12. 3D-zobrazení Levy function No. 3, n = 2

Tato funkce má 760 lokálních minim a 18 globálních minimum. Hodnota globálního minima je f (x)= -176,542, pro m = 5, při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

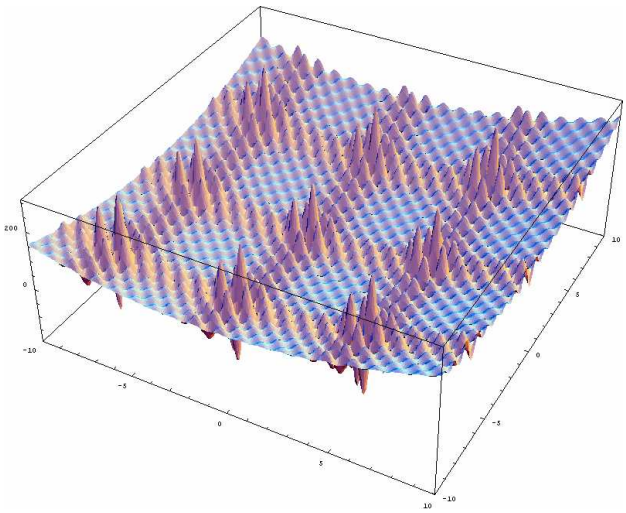
SimplexEvolution (SE), Annealed Nelder and Mead method (ANM) [9].

4.8.2 Levy function No. 5

Matematický zápis:

$$f(x) = \sum_{i=1}^{m} i\cos((i+1)x_1 + i)\sum_{j=1}^{m} j\cos((j+1)x_2 + j) + (x_1 + 1{,}42513)^2 + (x_2 + 0{,}80032)^2$$

$$-10 \le x_i \le 10, i = 1,2$$



Obr. 13. 3D-zobrazení Levy function No.5, n = 2

Tato funkce má 760 lokálních minim a 1 globální minimum. Hodnota globálního minima je f (x)= -176,1375 a souřadnice x [-1,3068 ; -1,4248], pro m = 5, při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

SimplexEvolution (SE), Annealed Nelder and Mead method (ANM) [9].

4.8.3 Levy function No. 8

Matematický zápis:

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_1 - 1)^2\left[1 + 10\sin^2(\pi y_i + 1)\right] + (y_n - 1)^2$$

$$y_i = 1 + \frac{x_i - 1}{4}, i = 1,...,n$$

$$-10 \le x_i \le 10, i = 1,2,...,n$$

Tato funkce má kolem 125 lokálních minim a 1 globální minimum o hodnotě f(x)=0 souřadnicích x [1;1;1].

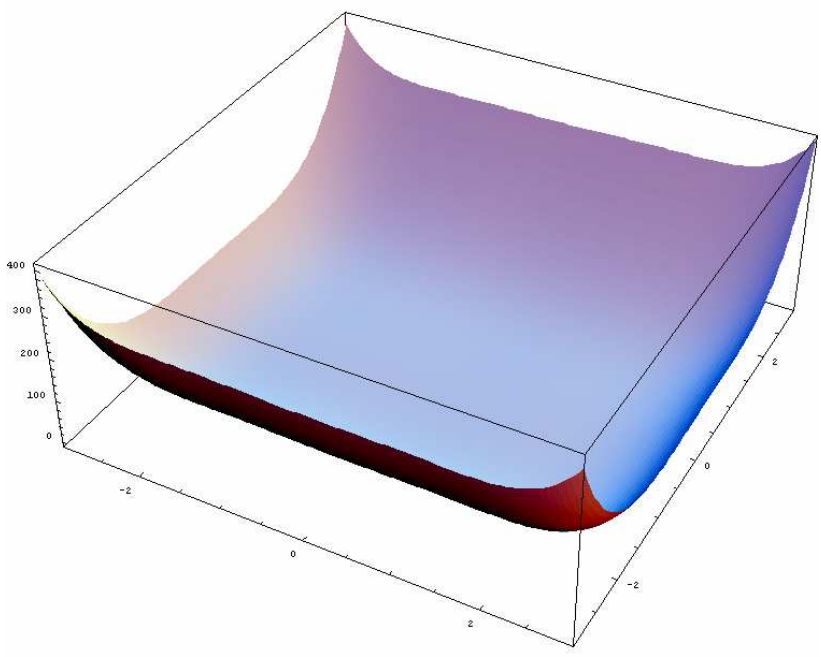
Na řešení této testovací funkce byli použity následující algoritmy: SimplexEvolution (SE), Annealed Nelder and Mead method (ANM) [9].

4.9 Six-hump camel back function

Matematický zápis:

$$f(x,y) = x^2\left(4 - 2{,}1x^2 + \frac{x^4}{3}\right) + xy + y^2\left(4y^2 - 4\right)$$

$$-3 \le x, y \le 3$$



Obr. 14. 3D-zobrazení Six-hump camel back function, n = 2

Globální minima jsou na souřadnicích [0,0898; -0.7126], [-0,0898; 0.7126] a hodnota f (x)= 0, při dimenzi n = 2.

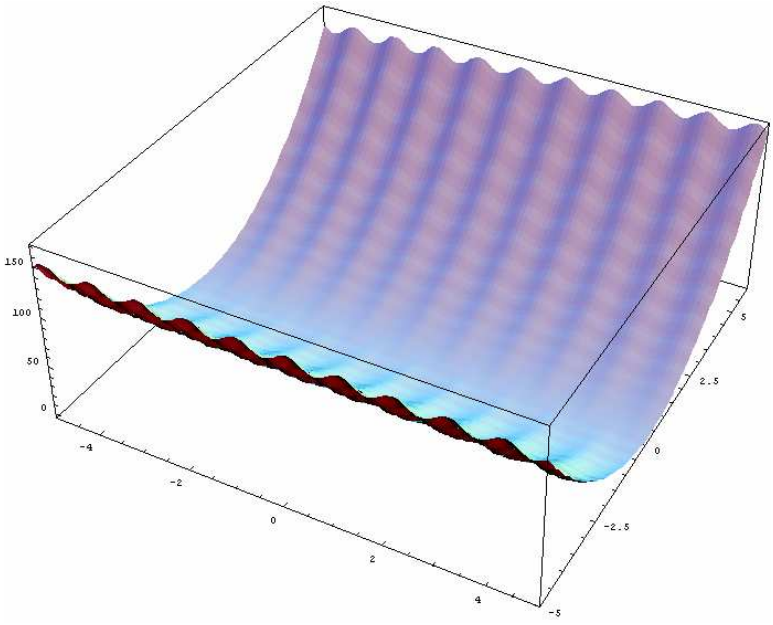
Na řešení této testovací funkce byli použity následující algoritmy:

Simultaneous Perturbation Stochastic Approximation (SPSA), Stochastic Approximation (SA), DIviding RECTangles (DIRECT), Efficient Global Optimization (EGO), Sequential Kriging Optimization (SKO) [10].

4.10 Branin function

Matematický zápis:

$$f(x) = \left(1 - 2x_2 + \frac{1}{20}\sin 4\pi x_2 - x_1\right)^2 + \left(x_2 - \frac{1}{2}\sin 2\pi x_1\right)^2$$



Obr. 15. 3D-zobrazení Branin function, n = 2

Globální minima jsou na souřadnicích x= [-π , 12.275], [π , 2.275], [9.42478, 2.475], hodnota f(**x**) = 0.397887, při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

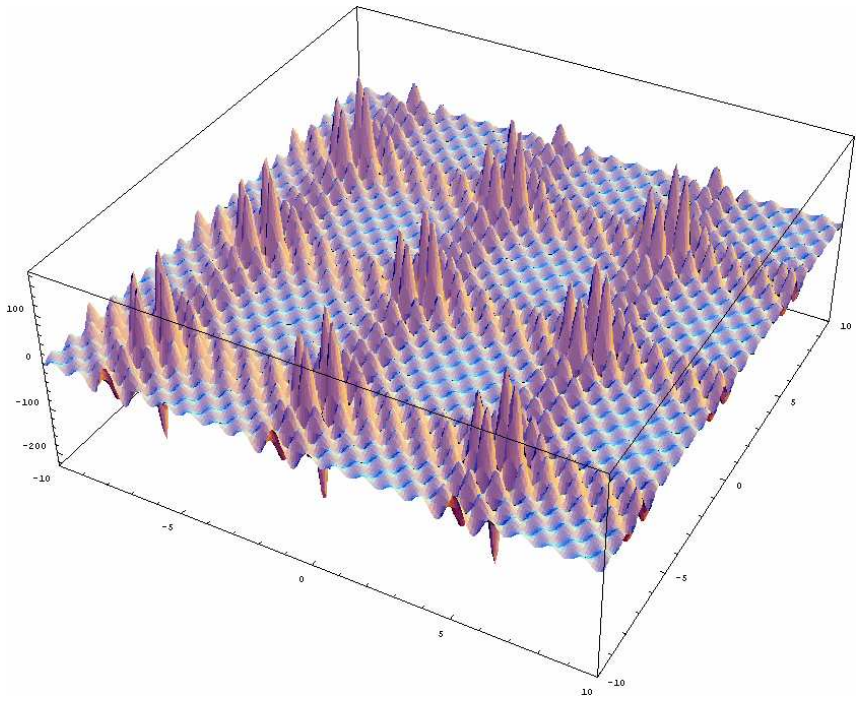
Pure Random Search (PRS), Simulated Annealing types 1 and 2 (SA1 and SA2, respectively), Tabu Search (TS)[6,11].

4.11 Shubert function

Matematický zápis:

$$f(x) = \prod_{i=1}^{n} \sum_{j=1}^{m} j \sin\left((j+1)x_i + j\right)$$

$$-10 \le x_i \le 10, i = 1,2,...,n$$



Obr. 16. 3D-zobrazení Shubert function, n = 2

Tato funkce má 18 globálních minim. Hodnota globálního minima je f (x)= -186,7309, pro m = 5, při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

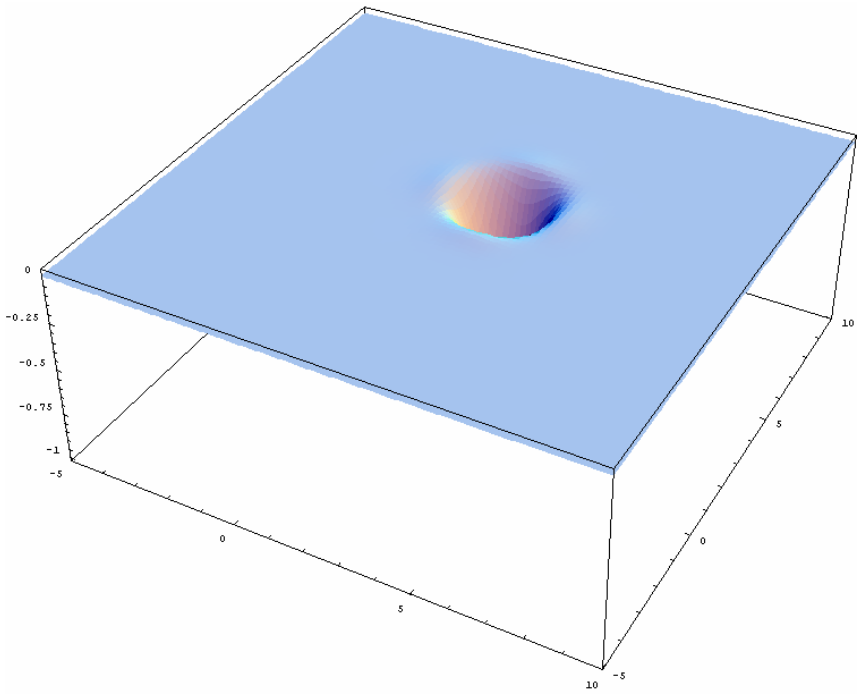
Pure Random Search (PRS), Simulated Annealing types 1 and 2 (SA1 and SA2, respectively), Tabu Search (TS)[6,11].

4.12 Easom function

Matematický zápis:

$$f(x) = -\left(\prod_{i=1}^{n} \cos(x_2)\right) \times \left(e^{\left(-\sum_{i=1}^{n} (x_i - \pi)^2\right)} \right)$$

$$-100 \le x_i \le 100$$



Obr. 17. 3D-zobrazení Easom function, n = 2

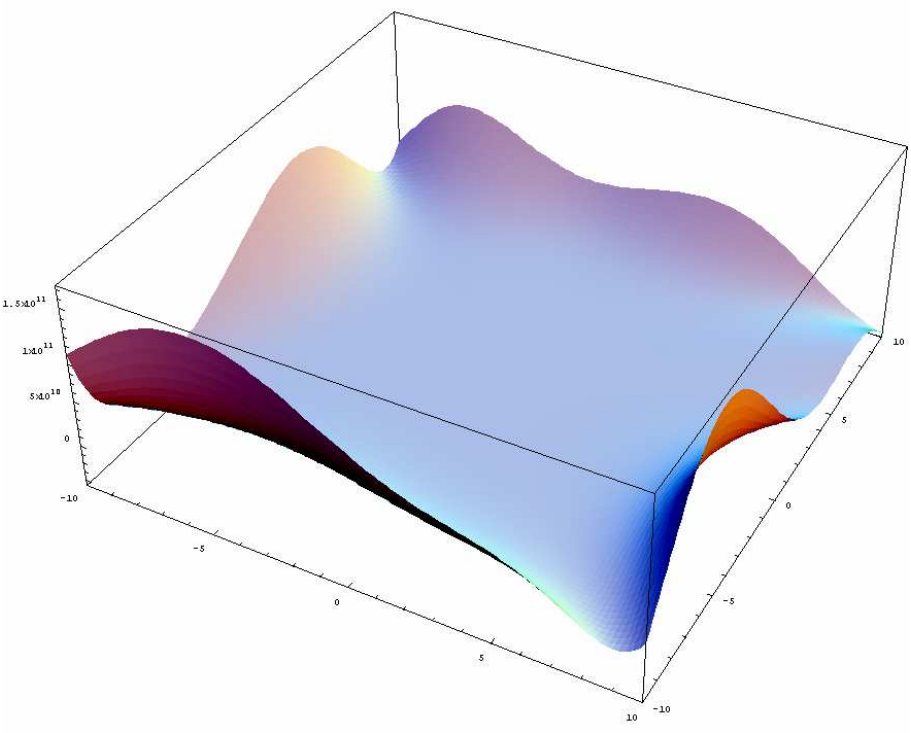
Globální minimum je na souřadnicích x= [π, π] a hodnota f(x) = - 1, při n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:
Enhanced Continuous Tabu Search (ECTS), Continuous Tabu Search (CTS), Continuous Reactive Tabu Search (CRTS min), Continuous Reactive Tabu Search (CRTS ave, Taboo Search (TS), Enhanced Simulated Annealing (ESA), INTEROPT [6,12].

4.13 Goldstein & Price Function

Matematický zápis:

$$f(x) = \left(1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right)$$
$$\times \left(30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$$



Obr. 18. 3D-zobrazení Goldstein & Price function, n = 2

Globální minimum je na souřadnicích x= [0, 1] a hodnota f(x) = - 3, při n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:
Enhanced Continuous Tabu Search (ECTS), Continuous Tabu Search (CTS), Continuous Reactive Tabu Search (CRTS min), Continuous Reactive Tabu Search (CRTS ave, Taboo Search (TS), Enhanced Simulated Annealing (ESA), INTEROPT [6,12].

4.14 Hartman function

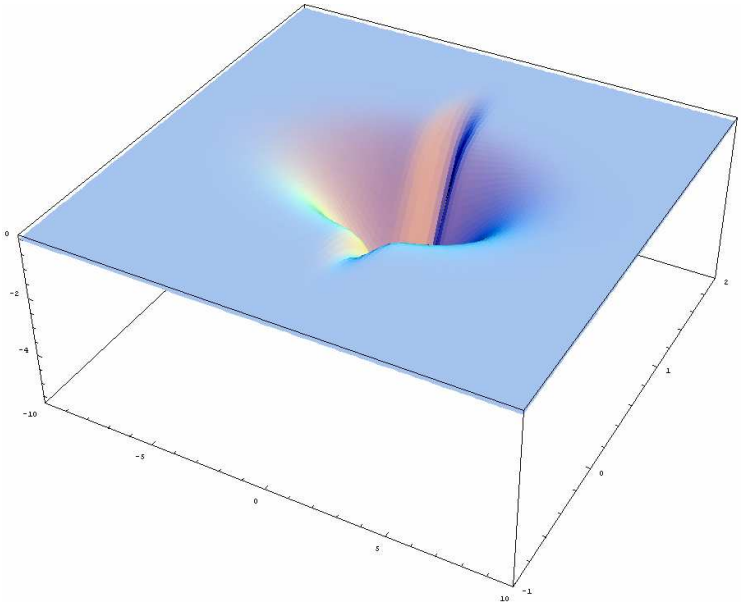
Matematický zápis:

$$f(x) = -\sum_{i=1}^{m} \alpha_i e^{\left[-\sum_{j=1}^{n} A_{ij}\left(x_j - P_{ij}\right)^2\right]}$$

$$\alpha = \begin{bmatrix} 1 & 1{,}2 & 3 & 3{,}2 \end{bmatrix}$$

$$A = \begin{bmatrix} 3 & 10 & 30 \\ 0{,}1 & 10 & 35 \\ 3 & 10 & 30 \\ 0{,}1 & 10 & 35 \end{bmatrix}$$

$$P = 10^{-4} \begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$$



Obr. 19. 3D-zobrazení Hartman function, n = 2

Globální minima jsou na souřadnicích **x** = [0.114, 0.556, 0.852], *f(x)* = -3.86
, při dimenzi n = 3.

Na řešení této testovací funkce byli použity následující algoritmy:

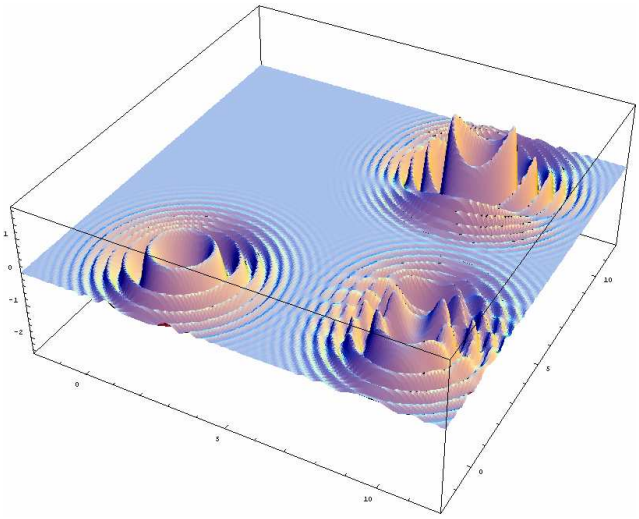
Simultaneous Perturbation Stochastic Approximation (SPSA), Stochastic Approximation (SA), DIviding RECTangles (DIRECT), Efficient Global Optimization (EGO), Sequential Kriging Optimization (SKO) [10].

4.15 Langerman function

Matematický zápis:

$$f(x) = -\sum_{i=5}^{m} c_i e^{-\frac{1}{\pi}\sum_{j=1}^{n}(x_j - a_{ij})^2} \cos\left(\pi \sum_{j=1}^{n}(x_j - a_{ij})^2\right)$$

Hodnoty pro proměnné a_{ij} a c_i jsou v příloze 1.



Obr. 20. 3D-zobrazení Langerman function, n = 2

Globální minimum má hodnotu f(x) =-1,0809 na souřadnicích x [9,681;0,6667] při m= 5 a při dimenzi n = 2.

Na řešení této testovací funkce byli použity následující algoritmy:

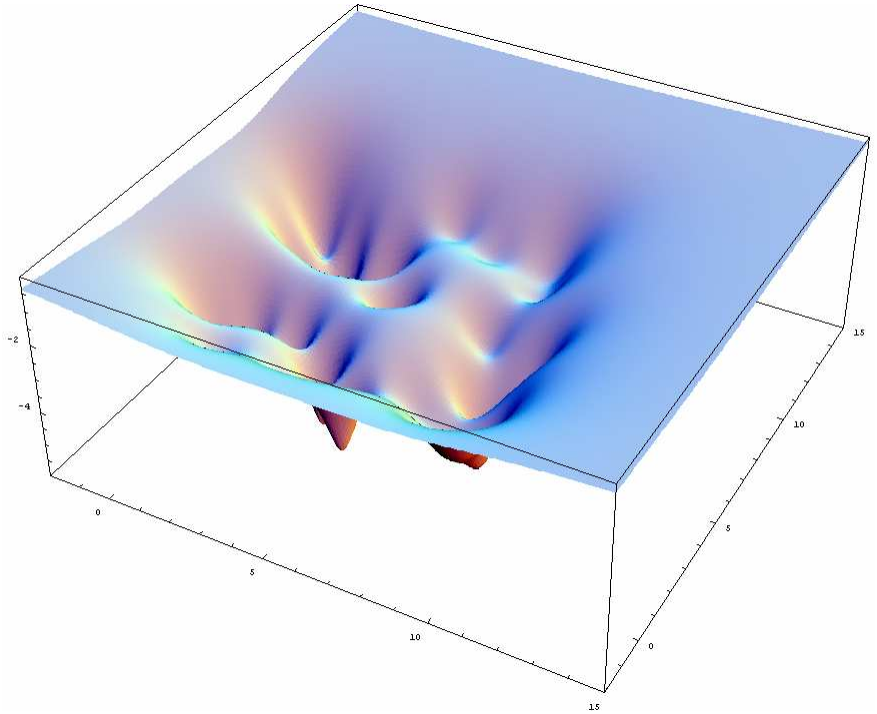
Steady-State Genetic Algorithm (SSGA), Population Training Algorithm (PTA) [7,13].

4.16 Modified Shekel´s Foxholes function

Matematický zápis:

$$f(x) = -\sum_{i=1}^{m} \frac{1}{\sum_{j=1}^{n} (x_j - a_{ij})^2 + c_i}$$

Hodnoty pro proměnné a_{ij} a c_i jsou v příloze 1.



Obr. 21. 3D-zobrazení Modified Shekel's Foxholes function, n = 2

Globální minimum f(x)= -10,406 na souřadnicích x [8,024;9,147] při dimenzi n = 2

Na řešení této testovací funkce byli použity následující algoritmy:

glb Solve, DIRECT [14, 15].

ZÁVĚR

Uvedené funkce jsou všechny příloze 1. a ta je na CD. Funkce jsou realizovány v programovém prostředí Mathematica a ve 3D-pohledu. U všech funkcí jsou uvedeny matematické formulace a globální minima. Věřím že tato práce může pomocí lidem kteří se zabývají designem evolučních algoritmů.

SEZNAM POUŽITÉ LITERATURY

[1] LAŽANSKÝ, Jiří, MAŘÍK, Vladímír, ŠTĚPÁNKOVÁ, Olga. *Umělá inteligence (3)*. 1. vyd. [s.l.] : Academia, 2003. 328 s. ISBN 80-200-0472-6.

[2] BASHAR, J.. *Kvantové teorie* [online]. 2001 [cit. 2006-04-20]. Dostupný z WWW: <http://cml.fsv.cvut.cz/~kupca/qc/node1.html>.

[3] HÁJEK, Petr. Multikriteriální optimalizace. *Přednášky* [online]. 2004 [cit. 2006-04-25], s. 1-9. Dostupný z WWW: <http://people.fsv.cvut.cz/~hajekp/vyuka/006DSP/2_OPT_multikrit.pdf>.

[4] Kalyan Veeramachaneni, Thanmaya Peram, Chilukuri K. Mohan, Lisa Osadciw "Optimization Using Particle Swarms with Near Neighbor Interactions", GECCO, July 11-16, 2003, Chicago, Illnois, Dostupný z WWW: <http://www.ecs.syr.edu/research/dreamsnet/Template/publications/Kalyan_GECCO_2003.pdf>.

[5] NEBRO, Antonio J.. *ESaM (Enumerative Search applied to Multi-objective optimization)* [online]. 2003-2004 , 25.6.2004 [cit. 2006-04-12]. Dostupný z WWW: <http://neo.lcc.uma.es/Software/ESaM/index.html>.

[6] HEDAR, Abdel-Rahman. *Global Optimization Test Problems* [online]. 2000. 1.1.2000 [cit. 2006-04-14]. Dostupný z WWW: <http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm>.

[7] POHLHEIM, Hartmut. *Documentation of the Genetic and Evolutionary Algorithm Toolbox for Matlab GEATbx - Start Page with overview of all documentation sections* [online]. 1 1994 , 2005 [cit. 2006-04-15]. Dostupný z WWW: <http://www.geatbx.com/docu/fcnindex-01.html#P95_3237>.

[8] CHELOUAH, R. A Continuous Genetic Algorithm Designed for the Global Optimization of Multimodal Functions . *Journal of Heuristics* [online]. 2000 [cit. 2006-04-26], s. 1-23. Dostupný z WWW: <http://ina2.eivd.ch/collaborateurs/rch/Pages_htm/..%5CDocumentations%5CArticles%5CArt_HEUR_00.pdf>.

[9] SOTIROPOULOS, D.G., PLAGIANAKOS, V.P., VRAHATIS, M.N. . An Evolutionary Algorithm for Minimizing Multimodal functions. *University of Patras* [online]. 1998 [cit. 2006-04-28], s. 1-7. Dostupný z WWW: <http://www.math.upatras.gr/~dgs/pdf/tr98-03.pdf>.

[10] HUANG, D., et al. Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. Scientific Forming Technologies Corporation [online]. 1995 [cit. 2006-04-22], s. 1-35. Dostupný z WWW: <http://www.mse.eng.ohio-state.edu/~huangd/writingSamples/Stochastic_Optimization_Sequential_Kriging.pdf>.

[11] SERRA, Pablo, et al. Pivot method for global optimization. *PHYSICAL REVIEW E* [online]. 1997 [cit. 2006-04-30], s. 1-4. Dostupný z WWW: <http://www.mat.univie.ac.at/~neum/glopt/mss/SerSK97.pdf>.

[12] CHELOUAH, Rachid , SIARRY, Patrick. Tabu Search applied to global optimization. *European Journal of Operational Research* [online]. 2000 [cit. 2006-05-04], s. 1-15. Dostupný z WWW: <http://ina2.eivd.ch/collaborateurs/rch/Pages_htm/..%5CDocumentations%5CArticles%5CArt_EJOR_00.pdf>.

[13] DE OLIVEIRA, Alexandre C. M., LORENA, Luiz Antonio Nogueira . Real-Coded Evolutionary Approaches to Unconstrained Numerical Optimization. *Evolutionary Approaches* [online]. 2000 [cit. 2006-05-16], s. 1-8. Dostupný z WWW: <http://www.lac.inpe.br/~lorena/alexandre/laptec-acmo-lorena.PDF>.

[14] BJORKMAN, Mattias , HOLMSTROM, Kenneth. Global Optimization Using the DIRECT. *Advanced Modeling and Optimization Volume 1* [online]. 1999 [cit. 2006-06-20], s. 1-21. Dostupný z WWW: <http://tomlab.biz/pubs/glbamo.pdf>.

[15] RONKKONEN , Jani. *Evolution function* [online]. 2004 [cit. 2006-05-21]. Dostupný z WWW: <http://www.it.lut.fi/ip/evo/functions/functions.html>.

Seznam použitých symbolů a zkratek

U	Úloha
A	Algoritmus
K	Celková vzdálenost
O	Složitost
p(n)	Polynom
KTS	Kvantový Turingův Stroj
m	Počet kroků
n	Dimenze

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

Na přilozeném CD je notebook do prostředí Mathematica, který je pojmenován funkce.nb