ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA STAVEBNÍ
KATEDRA STAVEBNÍ MECHANIKY

DIPLOMOVÁ PRÁCE

STOCHASTICKÁ OPTIMALIZACE NÁVRHU KONSTRUKCÍ

Vypracoval
Bc. Josef Nosek

Vedoucí práce
Ing. Matěj Lepš, Ph.D.

prosinec ’08

1 Obsah

2 Anotace

Diplomová práce se zabývá problémem srovnání různých optimalizačních metod a jejich vhodnosti pro různé typy optimalizace. Také poukazuje na neexistenci jednotného postupu jak testovat a hodnotit tyto metody. V tom práce shledává jeden z největších nedostatků dosud publikovaných článků.

Klíčová slova: Stochastická optimalizace, diferenciální evoluce, evoluční strategie, meta-heuristika, optimalizace konstrukcí, globální optimalizace

This thesis deals with the problem of comparing several optimization methods and thein suitability for different types of optimization. Presented work shows lack of methodology of already known testing algorithms and proposes a better scenario.

Key word: Stochastic optimization, differential evolution, evolution strategy, metaheuristics, structural optimization, global optimization.

3 Poděkování

Na tomto místě bych chtěl vřele poděkovat vedoucímu diplomové práce Ing. Matěji Lepšovi, Ph.D. za velkou trpělivost, podporu, velké množství konzultací a zodpovězení mnoha otázek při vypracování této práce. Dále bych rád poděkoval i ostatním za jejich postřehy a rady, zejména Ing. Anně Kučerové, Ph.D. a Ing. Janu Zemanovi, Ph.D.

4 Úvod

4.1 Současné poznání

V současné době je navrženo mnoho stochastických metod optimalizace. Jsou známé i některé přednosti a nedostatky metod. Co však považujeme za neprozkoumané je využití těchto metod v inženýrské praxi. Nelze totiž tvrdit, že metoda, která se matematikovi jeví jako dobrá se bude stejně jevit i inženýrovi a naopak. Proto se pokusíme ukázat metodiku srovnání optimalizačních metod tak, aby byla použitelná pro inženýrské využití.

4.2 Motivace

Vysoký pokrok výpočetní techniky a současný rozvoj materiálového inženýrství umožňuje vytvářet velmi přesné modely stavebních konstrukcí do nejmenších detailů, včetně všech jejich vlastností. Běžným pojmem, který se pro tyto pokročilé metody dnes používá, je BIM (Building Information Modeling) [21]. Snahou této práce je poukázat na možnosti nejnovějších přístupů k optimalizaci. Optimalizovat se dají zejména následující položky konstrukce (z hlediska statiky). Tvar (shape optimization), topologie (topology optimization), rozměry (size optimization) a skladba (layout optimization). Každá z těchto optimalizací má jiné vlastnosti z hlediska spojitosti a omezujících podmínek. Z toho vyplývá potřeba různých optimalizačních metod pro jednotlivé úlohy. Ve skutečnosti se většinou jedná o kombinaci optimalizačních problémů, ale kombinace optimalizačních metod je velmi obtížná. Proto se snažíme najít takové metody, které „obstojně“ zvládají většinu problémů. V současnosti je napsáno mnoho článků o jednotlivých optimalizačních metodách. Obvykle se snaží autor propagovat jednu metodu na úkor ostatních. Největším nedostatkem je obvykle metodika porovnání metod. Proto se budeme snažit ukázat jednu metodiku, která je podle nás “fair-play“.

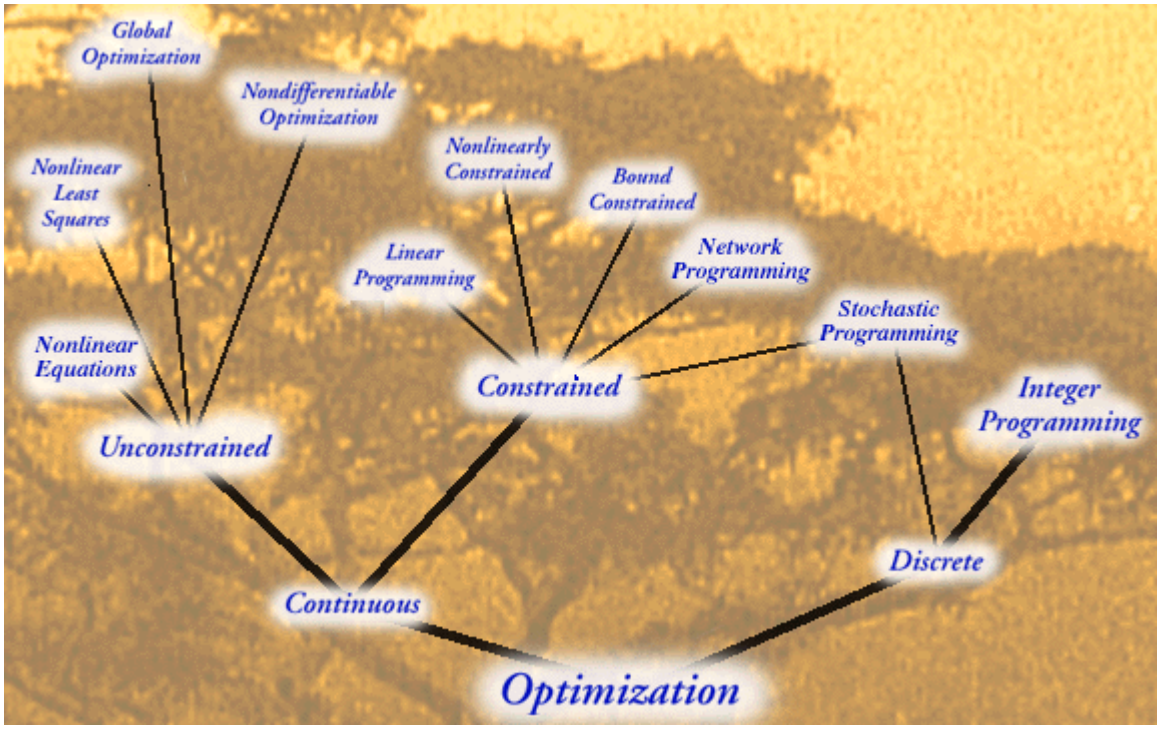
4.3 Cíl práce

Cíl této práce je ukázat srovnání metod a jejich „výkonnost“ na jednotlivých testovaných problémech. Aby to bylo možné udělat, tak musíme upravit některé zažité

postupy a hodnocení testů. V dosud publikovaných pracích je ukázáno mnoho testů a srovnání jednotlivých metod. Srovnání obvykle pracují se znalostí optima, což je u praktických problémů velmi složité.

4.4 Rozdělení úloh

V současné době je obvykle jednou z nejobtížnějších fází optimalizace samotná formulace zadání a zatřídění problému. Když víme jaká úloha před námi stojí, tak obvykle dokážeme najít nástroj, který ji zvládne vyřešit, resp. dokážeme najít patřičnou třídu optimalizačních algoritmů, které nám ji pomůžou vyřešit. Na Obr. 4.4-1 můžeme vidět jeden z možných způsobů, jak se dají klasifikovat optimalizační úlohy [1].



Obr. 4.4-1 – klasifikace optimalizačních problémů

4.5 Obsah práce

Práce se zabývá metodikou srovnání optimalizačních metod. Po obecném úvodu k optimalizacím je podrobně vysvětlen algoritmus diferenciální evoluce [2]. V další části je popsán test na sadě testovacích funkcí „ANDRE“ [27] a jeho vyhodnocení. Poslední velkou kapitolu, kterou se práce zabývá, tvoří srovnání diferenciální evoluce [2] a evoluční strategie [24] na reálných prutových konstrukcích. Práce si neklade za cíl upřednostnit a pozvednou některou z metod, ale snaží se o seriozní srovnání obou. Tomu jsou přizpůsobena i testovací

kriteria tak, aby žádnou metodu nezvýhodňovala. Jestliže v práci není napsáno jinak, tak se budeme vždy snažit o minimalizaci dané objektivní funkce.

5 Proč se používá optimalizace

5.1 Cíl optimalizace

Optimalizace si klade za cíl najít nejlepší řešení. Z inženýrského hlediska je zbytečné tvrdit, že potřebujeme vždy znát nejlepší řešení. Velmi často stačí najít lepší řešení než doposud známé (sub-optimum). Matematik se obvykle snaží najít opravdu nejlepší řešení problému a současně mít jistotu, že je to opravdu nejlepší možné řešení (globální minimum nebo maximum). V některých případech je z inženýrského hlediska podstatně výhodnější zůstat pouze v lokálním minimu / maximu, které je stabilní a „robustní“ [5].

5.2 Optimalizace ve stavebnictví

Dnešní velmi rychlý svět klade značné nároky na všechny, kteří se účastní procesu výstavby. Finanční trhy jsou velmi nestabilní a tak se investoři snaží stlačit dobu výstavby na co nejkratší čas. Je zcela běžné, že ještě není hotová projektová dokumentace k celé stavbě, ale již se například provádí spodní stavba. To klade obrovské nároky na projektanta, který se tím dostává do časové tísně. Přitom většina projektantů je vybavena vysokým výpočetním výkonem. Bohužel samostatný výpočetní výkon nic neznamená. Je třeba mít i vhodný software. V současné době moderní software nabízí parametrické modelování, které je velmi dobrou výchozí půdou pro následnou optimalizaci. Tím že projektant zadá do softwaru konstrukci, vytvoří matematický model, který lze chápat jako funkci mnoha proměnných. Zadáním dalších požadavků (např. minimálním stupněm vyztužení, maximálním průhybem, omezením napětí, konstrukčními zásadami) tvoří okrajové podmínky pro danou funkci. Tím je vytvořena živná půda pro optimalizaci.

V současné době dostupný velký výpočetní výkon, který obvykle většinu času pouze morálně stárne (není aktivně využit) a postupem času se z něj stává pouze kus „železa“ usnadňuje využití optimalizace. Morální stárnutí předbíhá o několik řádů stárnutí fyzické, není proto žádný důvod nechat počítače nevyužité.

Otázkou je proč se ve stavebnictví již optimalizace neuplatňuje. Je zde několik důvodů. Pojďme se podívat na ty nedůležitější. Ve stavebnictví se nejedná o hromadnou, ale o kusovou výrobu. Můžeme tvrdit, že když se budeme snažit postavit dva stejné domy, tak ve výsledku budou jiné. Stačí se podívat například na základové poměry, které se mohou lišit dost výrazně i v rámci jednoho staveniště. Dalším faktorem je rozmanitost výroby. Každý investor chce být pokud možno originální a architekt se mu snaží vyhovět. Nelehký úkol má pak projektant, který na základě zkušeností musí dát budově reálný rozměr. Často se stane, že se projektant snaží vyhnout komplikovanějším konstrukcím a tak se objekt pokouší vystavět na jednoduchých prvcích. Velmi často tomu napomáhá již zmíněná časová tíseň. I když si projektant dá velmi mnoho práce (jestliže ji bude investor ochoten zaplatit), tak se nikdy nemůže dostat na úroveň optimalizace návrhu, který lze dosáhnout softwarem.

Jen pro úplnost bychom měli podotknout, že je vždy potřeba automaticky navrhnuté „optimální“ řešení prozkoumat z hlediska proveditelnosti a zda opravdu splňuje všechny námi požadované vlastnosti. Stačí když chybně zadáme okrajové podmínky problému (třeba krytí výztuže) a algoritmus může naší chyby využít pro svůj prospěch. Bohužel stále za správnost projektu ručí projektant, nikoliv počítač a dokonce ani programátor, který mohl v aplikaci udělat chybu. Ovšem nejčastěji udělá chybu uživatel, který svou neznalostí nebo nedbalostí nazadá patřičně konstrukci do programu.

6 Meta-heuristické metody

Heuristiku můžeme charakterizovat jako metodu jak nalézt rozumné řešení daného problému. Heuristické postupy nezaručují téměř nic. Nezaručují nalezení nejlepšího řešení a ani to, že najdou řešení v krátkém čase. V mnoha případech však tyto metody pracují velmi dobře.

Předponou „meta“ se snažíme vyjádřit vstup okolností do metody. Jde jen o částečně náhodný algoritmus, který se snaží využívat předchozích znalostí. Nutno podotknout, že může dávat i horší výsledky než heuristický algoritmus [22].

6.1 Historie metod

První návrhy okolo meta-heuristik jsou z poloviny minulého století [23], kdy vznikaly první evoluční strategie a jako samostatné odvětví i evoluční programování. Tyto metody se snažily napodobením evoluce optimalizovat problém. Asi o 20 let později vznikají genetické algoritmy a s nimi svázané genetické programování. V roce 1983 a 1985 nezávisle na sobě představují Kirkpatrick a Černý metodu simulovaného žíhání. O rok později je publikována

„Tabu search“ metoda, která jako první zavádí do algoritmu paměť a začíná se hovořit o pojmu meta-heuristiky. Další prudký rozvoj nastal v polovině devadesátých let minulého století, kdy je představena metoda „particle swarm optimization“ a o dva roky později diferenciální evoluce.

V současném století byly představeny například algoritmy „honey bee algorithm“ [25] a „artifical bee algorithm“ [26].

Jak je zřejmé, tak metod je mnoho a stále se vyvíjejí nové, které mají za cíl překonat již známé. Abychom mohli všechny tyto metody porovnávat, tak je nutno nejdříve stanovit provnávací kritéria. Je zde však několik problémů, které je nutno zmínit. První je neexistence standardního benchmarku, druhý problém tvoří metodologie testů a interpretace výsledků. Oba problémy podrobněji rozebereme dále.

6.2 Popis algoritmů

Nejjednodušším algoritmem je metoda „Monte Carlo“. Metoda zcela náhodně volí body, v nich počítá funkční hodnoty. Jestliže je nalezená řešení lepší než doposud známé, pak ho nahradí a pokračuje se dál. Tato metoda se dá považovat za metodu hrubé síly.

Za další meta-heuristickou metodu můžeme považovat „horolezecký algoritmus“. Tento algoritmus prohledá okolí daného bodu a na základě spádnic se rozhodne kam půjde dál. Obvykle skončí v nějakém optimu, ale většinou v lokálním. Opakovanými starty z různých bodů, se snažíme dojít do globálního optima.

Evoluční strategie je jako jeden z řady evolučních algoritmů. Tyto algoritmy jako první zavedli adaptaci sama-sebe. Podrobný popis této metody lze nalézt v [19].

Diferenciální evoluce je relativně mladá metoda. Protože byla zvolena jako hlavní metoda, tak jí věnujeme celou kapitolu.

6.3 Diferenciální evoluce (DE)

Jedná se o poměrně novou metodu, která byla představena až v roce 1995 [2]. Tato metoda má pouze 2 parametry a přesto je poměrně robustní.

Metoda pracuje způsobem, který lze shrnout do několika málo bodů:

a) Stanovení parametrů – řídí cyklus

   NP – počet jedinců v populaci, obvykle se stanovuje na 10 x D, kde D je počet neznámých

   F – konstanta, která určuje velikost kroku mutace

   C (crossover) – určuje práh křížení

b) Tvorba populace první generace

Novou generaci obvykle stanovujeme náhodně v zadaných mezích dle 6.3-1

$$x_d^{i,0} = x_{d,\min} + rand(x_{d,\min}, x_{d,\max}) \qquad (6.3\text{-}1)$$

c) Reprodukční cyklus

Pro každého jedince (rodiče) je vygenerován zkušební jedinec následujícím způsobem. Jsou vybráni tři jedinci aktuální generace. Rozdíl prvních dvou vynásobený konstantou F je přičten ke třetímu jedinci. Tím je vytvořený zmutovaný jedinec. Poté je pro každou složku generováno náhodné číslo <0,1>. Pokud je náhodné číslo menší než práh křížení (C) je zmutovanému jedinci pro aktuální rozměr přiřazen parametr ze zmutovaného jedince. Tím máme vytvořeného zkušebního jedince (trial vector).

d) Testování funkční hodnoty zkušebního jedince

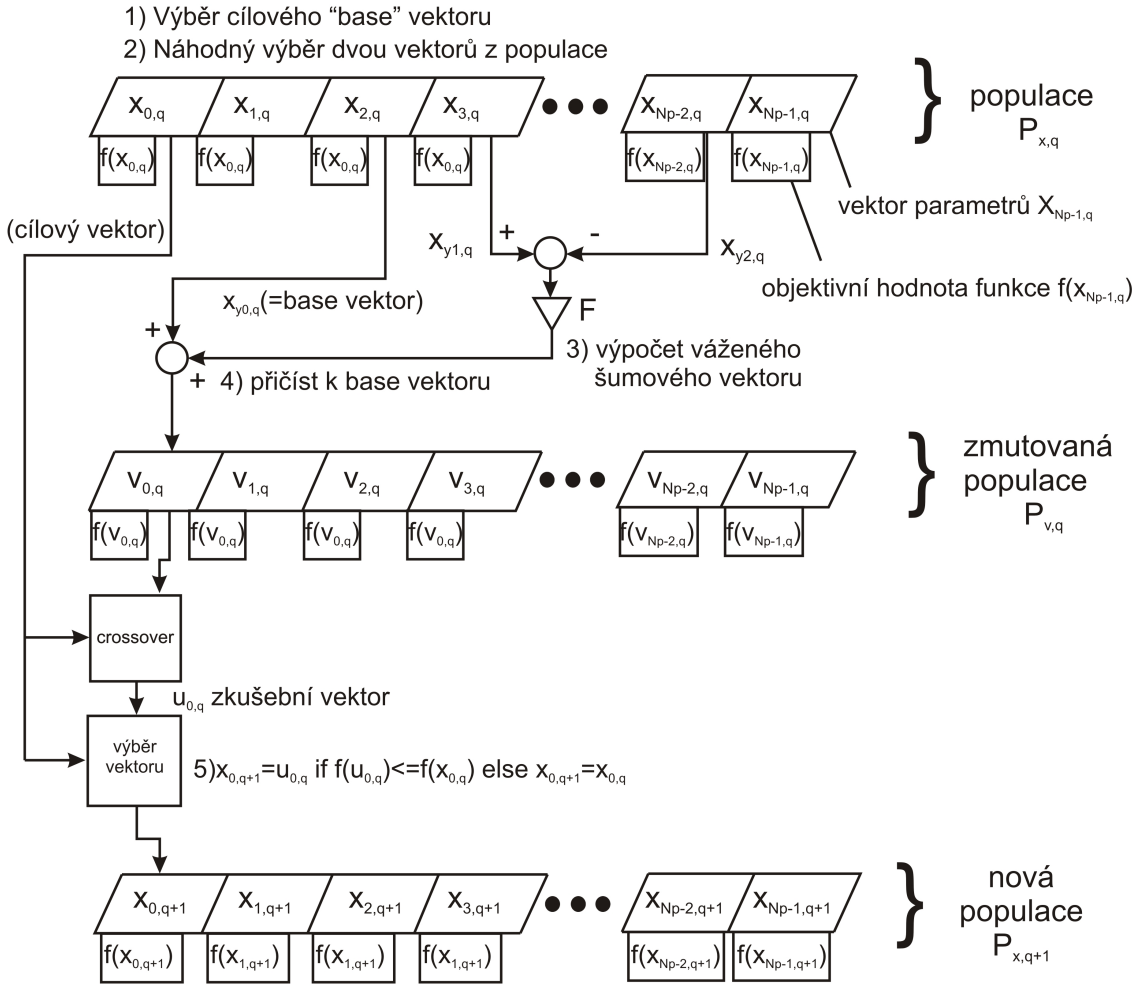
e) Sestavení nové generace

Jestliže funkční hodnota zkušebního jedince je lepší než rodiče, pak postoupí zkušební jedinec. V opačném případě postoupí rodič beze změny. Tím je zajištěno, že se do nové populace nedostane horší jedinec než rodič.

f) Postup c-e opakujeme do dosažení zastavovacích podmínek.

Je nutno zmínit, že toto je jen jedna z možných variant DE. Existují i další, kde například v následující populaci je vícekrát zastoupen nejlepší jedinec.

Nejlépe celý proces popsaný v bodech a-f vystihuje Obr. 6.3-1.



Obr. 6.3-1 – schéma funkce diferenciální evoluce

6.4 Ladění parametrů DE

Diferenciální evoluce má pouze již zmíněné dva parametry. C – crossover, který vyjadřuje pravděpodobnost křížení a parametr F, který udává velikost kroku mutace. Ladění parametrů proběhlo na sadě funkcí „ANDRE“ (viz příloha). Na jednoduchém benchmarku bylo ukázáno, že nastavení parametru C je vhodné volit okolo hodnoty 0,8 ± 0,1. Otázka nastavení parametru F je poněkud složitější. Zde se totiž nabízí otázka jestli musí být parametr F konstantní během všech iterací. V tradiční formulaci DE tomu tak je, ale my jsme se této myšlenky nedrželi a F jsme měnili během optimalizace.

Nastavení parametru F bylo provedeno následujícím způsobem. Jestliže algoritmus stagnuje (nedaří se najít lepší řešení) tak předpokládáme, že mohly nastat 3 situace. V prvním případě jsme v optimu a každý další pokus je zbytečný. Ve druhém jsme se dostali někam do lokálního minima a nedaří se z něj dostat ven. V posledním případě jsme velmi blízko optima, ale většina nových vektorů ho „přestřelí“. Na každý případ se dá velmi vhodně naladit parametr F tak, aby postupně ošetřil jednotlivé případy.

1) Jestliže jsme již v globálním minimu, tak každý další pokus je zbytečný. Proto můžeme nastavit F libovolně a lepší řešení nenajdeme. Jestliže ano, tak jsme jinde než si myslíme.
2) V případě pádu do lokálního minima potřebujeme dostat do systému velmi odlišné možnosti, abychom se dostali mimo toto minimum. To lze velmi snadno udělat tak, že parametr F nastavíme na dostatečně vysokou hodnotu.
3) Jestliže se nacházíme velmi blízko optima, tak naopak F snížíme a tím se přiblížíme k optimu ještě blíže.

Pozorný čtenář si povšimne, že jsme nezmínili jak zjistíme, který případ nastal. Rozhodující je počet iterací po které jsme nenašli takový vektor, který má stejnou nebo lepší funkční hodnotu. V první fázi po *k* iterací tvrdíme, že nastal případ 2. Jestliže během této doby nenalezneme lepší řešení, tak předpokládáme že jsme v případu 3. Když najdeme kdykoliv nový (lepší nebo stejný) vektor tak se celý proces opakuje od začátku. Hodnota *k* je závislá na počtu dimenzí a maximálním počtu iterací vztahem *k=0.03*D*itermax*, kde *D* je počet dimenzí a *itermax* je maximální počet iterací.

S takto nastavovaným parametrem F dostáváme lepší výsledky než když máme F fixní, resp. dosahujeme lepší úspěšnosti v nalezení optima. Celý tento proces je analogií k 1/5 pravidlu v ES, které je ukázáno v následujícím odstavci.

6.5 Evoluční strategie (ES)

Jedná se o metodu, která byla navržena v 60. a 70. letech minulého století. Jako první tato metoda zavádí úpravu sama sebe (self-adaptation). Metodu lze popsat v několika následujících krocích.

a) Zadání vstupních parametrů. Můžeme volit následující parametry.
   Countmax – počet iterací po které se nebude měnit sigma
   Sigma – určuje velikost kroku
   Celkový počet iterací
b) K nejlepšímu známému vektoru přičteme náhodný vektor (v rozsahu prohledávaného prostoru), který vynásobíme parametrem sigma.
c) Ověříme zda vytvořený testovací vektor leží v dovolených hranicích. Jestliže ne, tak ho tam posuneme.

d) Ohodnotíme testovací vektor a přičteme 1 do proměnné „countmax“. Jestliže je úspěšnější než aktuálně známý, tak ho označíme za nejlepší a zapamatujeme si, že jsme měli úspěch (přičteme 1 do proměnné success).
e) Kroky b – d opakujeme až do dosažení maximálního počtu volání (countmax) s danou sigmou.
f) Po dosažení maximálního počtu volání s danou sigmou změníme sigmu podle pravidla 1/5 úspěšnosti. Nejlépe to lze vyjádřit následovně.

C = 0.65 -opravný součinitel [0.6 – 1]

Ps = success / countmax -poměr úspěšných volání

Když ps > 0.2
Sigma = sigma / c
Když ps < 0.2
*Sigma = sigma * c*

Jinak sigma = sigma původní

Po změně parametru sigma opět opakujeme cyklus b-e

g) Celý cyklus b - f opakujeme do dosažení zastavovacích podmínek.

7 Hodnocení testů

Jakékoliv testy má smysl dělat pouze tehdy, když máme definováno i zpracování jejich výsledků. V opačném případě to můžeme považovat za házení kamení do studny.

7.1 Zastavovací podmínky

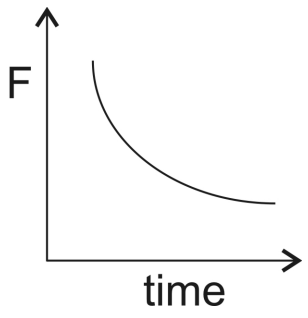
Dnes neexistuje žádný, nám známý konsenzus, který by stanovoval, kdy a jak mají zastavit iterační optimalizační algoritmy. Tradiční přístup, který hodnotí nezbytný počet volání funkce k nalezení optima a procentuální úspěšnost nalezení optima s danou odchylkou, je podle nás naprosto nevyhovující. Zastavovat při nalezení optima lze pouze u benchmarků, kde známe optimum. U praktických problémů je to nepoužitelné. Navíc je zde problém přesnosti optima. Obvykle se posuzuje pouze funkční hodnota s danou tolerancí. Jestliže je tolerance příliš velká, tak může dojít k pádu do lokálního optima, které má funkční hodnotu blízkou globálnímu optimu. Další možností je dosažení určitého počtu volání funkce (iterací). To je proveditelné u reálných úloh, ale je zde otázka, kolik iterací se má spočítat. Na tuto

otázku neexistuje jednotná odpověď. Další možností je stanovení časového úseku a výpočetní síly, kterou jsme ochotni problému obětovat. To se již blíží reálným úlohám.

7.2 Kriteria hodnocení

Při hodnocení je třeba rozdělit úlohy na ty, kde známe optimum (benchmarky) a ostatní. Jestliže známe optimum, pak je velmi výhodné hodnotit algoritmus podle toho, zda jej dosáhl či nikoliv. Zde ale musíme dát velký pozor na toleranci s jakou budeme optimum posuzovat. Jak již bylo zmíněno, jestliže bude tolerance příliš velká, pak může dojít k vyhodnocení lokálního optima jako globálního.

U běžné úlohy, kde neznáme optimum, toto hodnocení nemůžeme udělat. Je však možné sledovat vývoj hodnot postupně po jednotlivých iteracích metody. V našem případě budeme sledovat jaké řešení najde algoritmus po *n* volání funkce. Z takto získaných dat můžeme říci, které algoritmy jsou vhodné pro přesnou a které pro rychlou optimalizaci. Velmi dobře náš problém vystihuje Obr. 7.2-1. Pohybujeme se po pomyslné křivce. Jestliže necháme algoritmus běžet příliš krátce, tak nedostaneme dobrý výsledek. V opačném případě budeme jen plýtvat strojovým časem. Je na nás tedy nelehký úkol. Musíme rozhodnout kolik času dáme metodě. I úspěšnost algoritmu závisí na množství času, který problému obětujeme. Jestli budeme hovořit o množství času nebo počtu iterací, je pro naše pokusné účely záměnné.



Obr. 7.2-1

7.3 Efektivita

Základním hnacím motorem vývoje optimalizačních algoritmů je úspora zdrojů. Můžeme mluvit o prostředcích v libovolné formě. Proto i při samotných optimalizačních úlohách bychom měli brát ohled na vynaložené prostředky, na jejich vývoj a implementaci.

Měli bychom brát na zřetel dva hlavní vstupy a to čas člověka a čas stroje. Každý vstup má danou hodnotu (cenu) a naší snahou je minimalizovat jejich součet. Tím chceme ilustrovat příklad, že někdy může být výhodnější metoda, která je sice trochu pomalejší, ale mnohem jednodušší na implementaci. Nejen proto je hodnocení optimalizačních algoritmů velmi složitý problém.

8 Test DE vs. ES na sadě „ANDRE“

Naší snahou je porovnat optimalizační algoritmy a pokusit se je charakterizovat. Na základě popisu chování těchto algoritmů na širokém poli funkcí se můžeme snažit získat třídu úloh, pro které se hodí. Předem je třeba říci, že se chceme vyvarovat jakéhokoliv generalizování, proto říkáme, že ten či onen algoritmus bude „pravděpodobně“ vhodnější.

8.1 Matematické testovací funkce

Zatím bohužel neexistuje sada „standardních“ testovacích funkcí. My použijeme sadu „ANDRE“ [27]. Sada obsahuje 20 různých funkcí. Funkce mají od jedné do 20 proměnných a testují různé aspekty. Seznam funkcí je uveden v příloze. Tato sada byla již použita jako testovací sada pro mnoho jiných algoritmů, proto předpokládáme, že je dostatečně výstižná pro postihnutí běžných typů funkcí.

8.2 Vstupní předpoklady

Budeme testovat dva algoritmy. Prvním z nich bude diferenciální evoluce a jako soupeře jí položíme evoluční strategii.

Maximální počet volání funkce jsme stanovili na 10 000 x 10 x D, kde D je počet proměnných funkce. Tento počet byl zvolen úmyslně. 10 x D je totiž velikost populace u DE. Obě metody spustíme na každé funkci 100x, aby výsledky měly statisticky vypovídající hodnotu.

8.3 Průběh testu

Na provedení testu byl sestaven program v aplikaci MATLAB. Program má tři úrovně.

V nejnižší jsou samotné funkce, vstupním parametrem je zde sada proměnných a výstupním je hodnota v zadaném bodě. Funkce jsou zcela totožné pro DE i ES.

Druhou úroveň tvoří samotný algoritmus. Zde se provádí změny bodů ve kterých chceme počítat hodnoty. Dá se říci, že je zde uloženo jádro algoritmu.

Třetí úroveň pouze volá samotné metody, zadává jim vstupní parametry a ukládá výsledky jednotlivých testů.

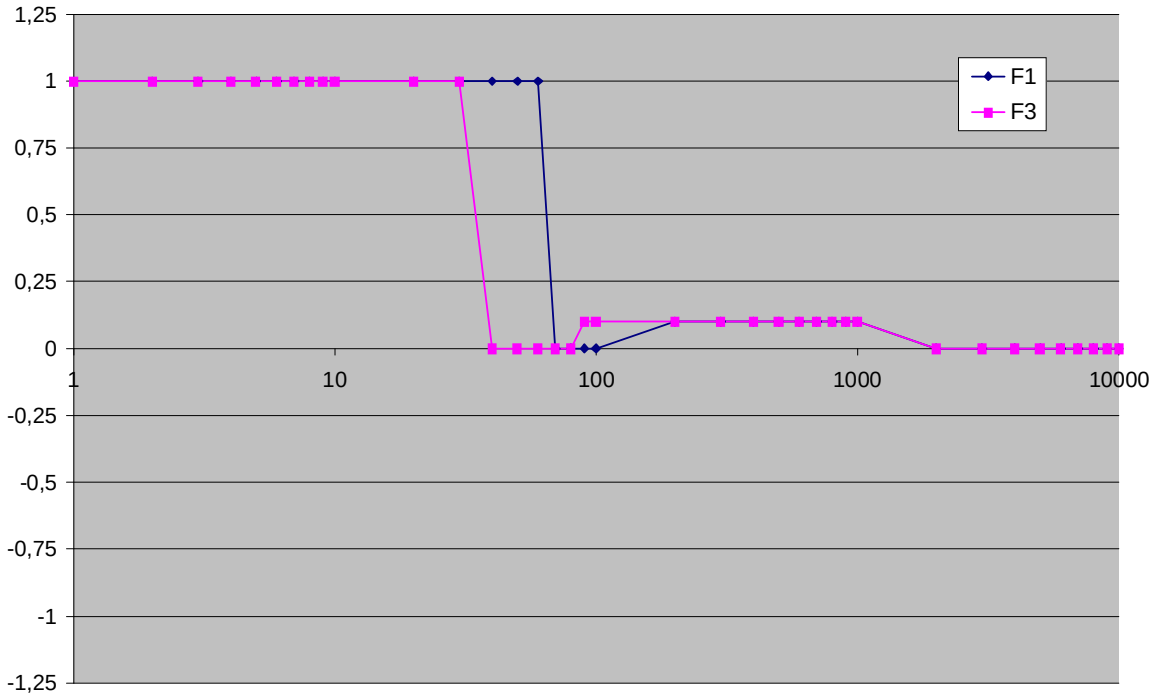
8.4 Vyhodnocení na sadě funkcí

Na vypočtené sadě dat je spuštěn statistický T-test s hladinou spolehlivosti 95%, který vyhodnocuje, zda-li je rozložení dvou porovnávaných výsledků z dvou různých metod (DE a ES) různé. Princip statistické funkce T-test je popsán v příloze. Tím zjišťujeme, zda je možné od sebe obě rozdělení odlišit. Pro lepší přehlednost vždy vykreslíme do jednoho grafu funkce o stejném počtu proměnných. Je nutno podotknout, že obecně neexistuje lineární vztah mezi počtem proměnných funkce a její složitostí. Pro tento účel by bylo vhodné rozdělit funkce do jednotlivých tříd, ale to je mimo rámec této práce.

Ve všech následujících grafech ukazujeme výsledky testu. Na vodorovné ose je počet volání funkce podělený 10 x D (počtem členů populace). Proto i kdekoliv dále budeme mluvit o počtu iterací, máme namysli počet volání funkce podělený 10 x D. Hodnoty na svislé ose mají význam dle následující tabulky:

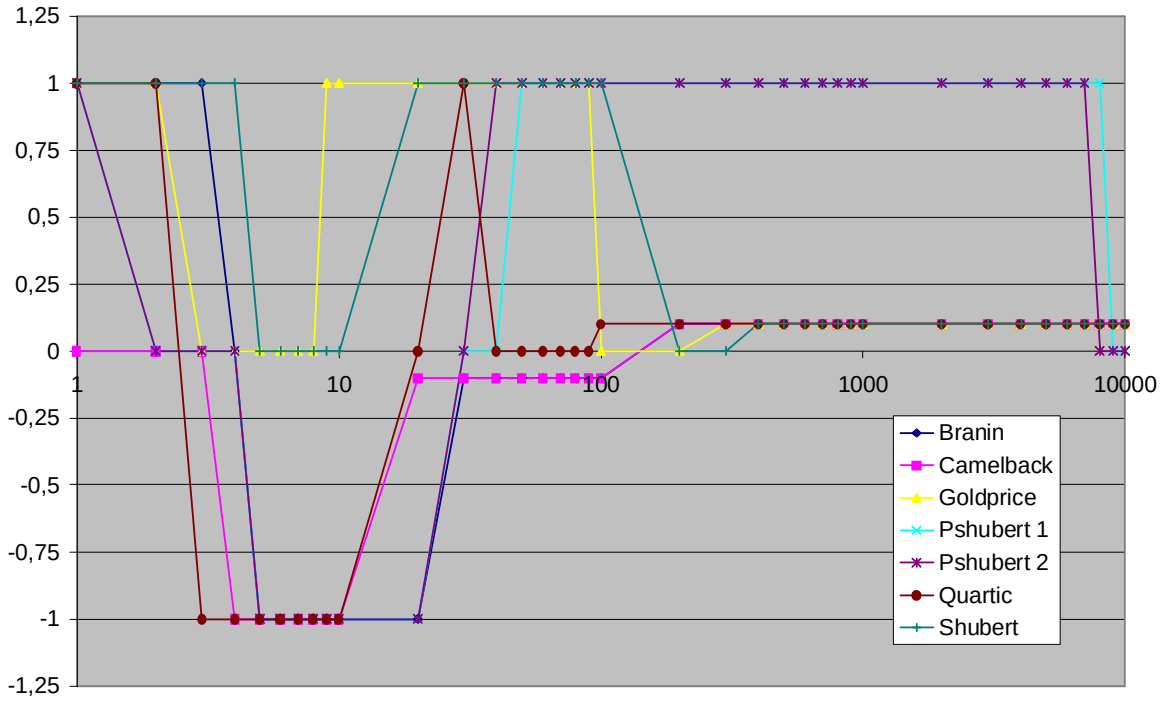
1	DE má nižší průměrnou hodnotu o více jak 10^{-2}
0,1	DE má nižší průměrnou hodnotu o méně jak 10^{-2}
-1	ES má nižší průměrnou hodnotu o více jak 10^{-2}
-0,1	ES má nižší průměrnou hodnotu o méně jak 10^{-2}
0	Jednotlivé průběhy nelze odlišit

Na Gr. 8.4-1 jsou vykresleny funkce 1 proměnné. Konkrétně, funkce F1 a F3.



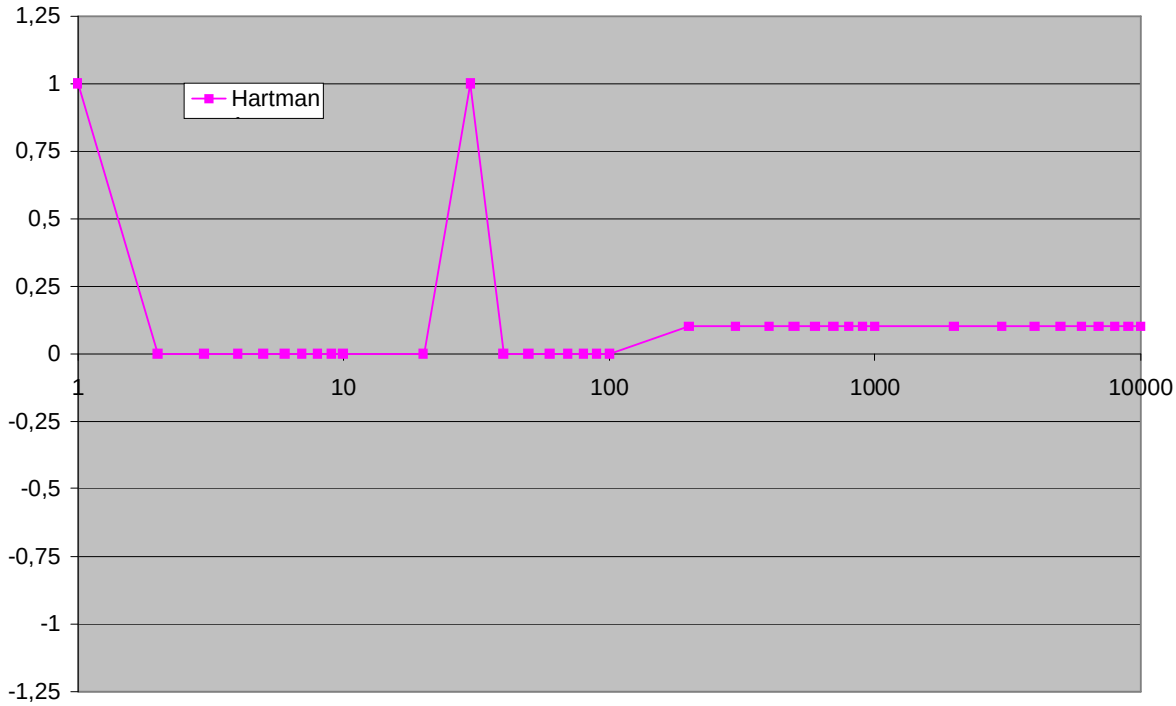
Gr. 8.4-1 - T-test na funkcích 1 proměnné v závislosti na počtu iterací

Na Gr. 8.4-2 jsou vykresleny funkce 2 proměnných. Těchto funkcí je celkem 7. To již lze považovat za reprezentativnější vzorek než předchozí případ. Problém je v tom, že každá funkce má jiný aspekt obtížnosti (je z jiné třídy). Jedna má ostrá lokální minima, jiná má zase velké „údolí“, které je mírně svažité.

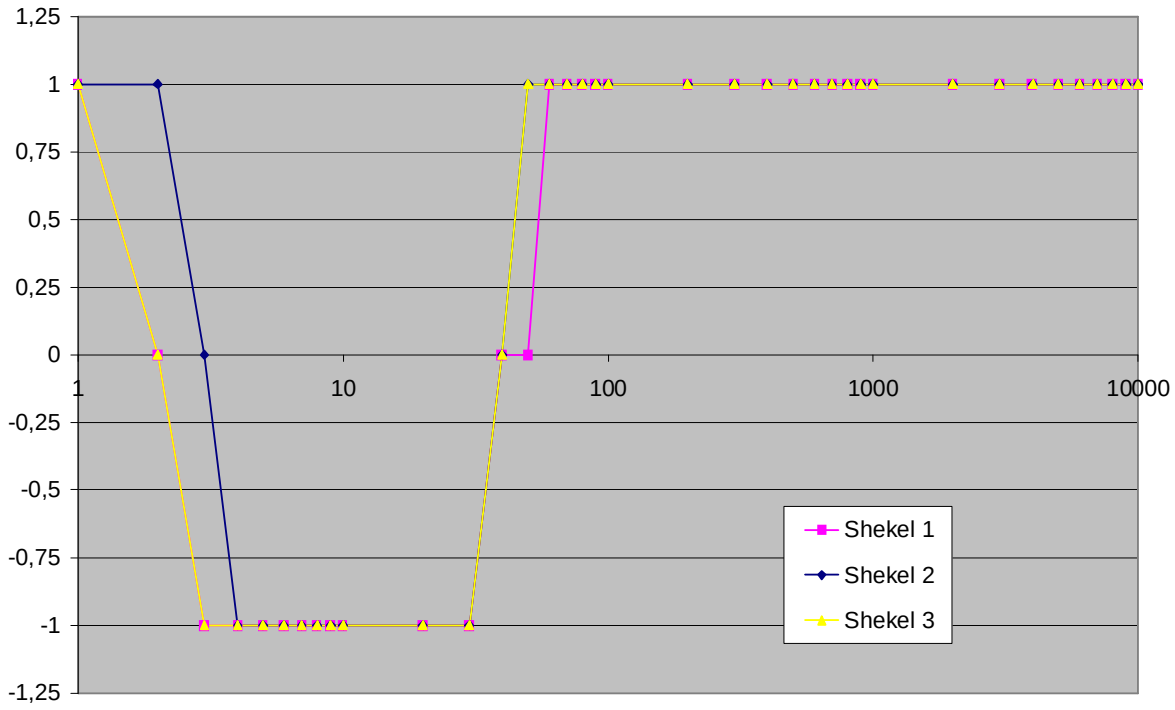


Gr. 8.4-2 - T-test na funkcích 2 proměnných v závislosti na počtu iterací

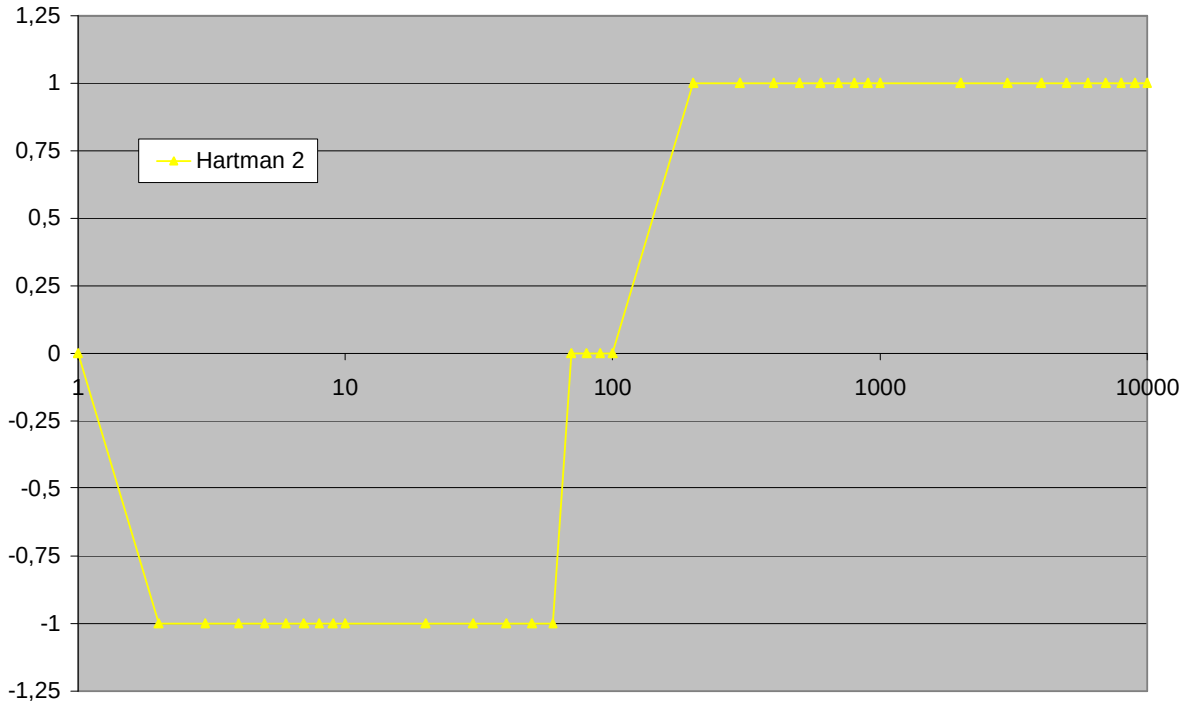
V Gr. 8.4-3 až Gr. 8.4-7 zobrazujeme úspěšnost metod na jednotlivých funkcích.



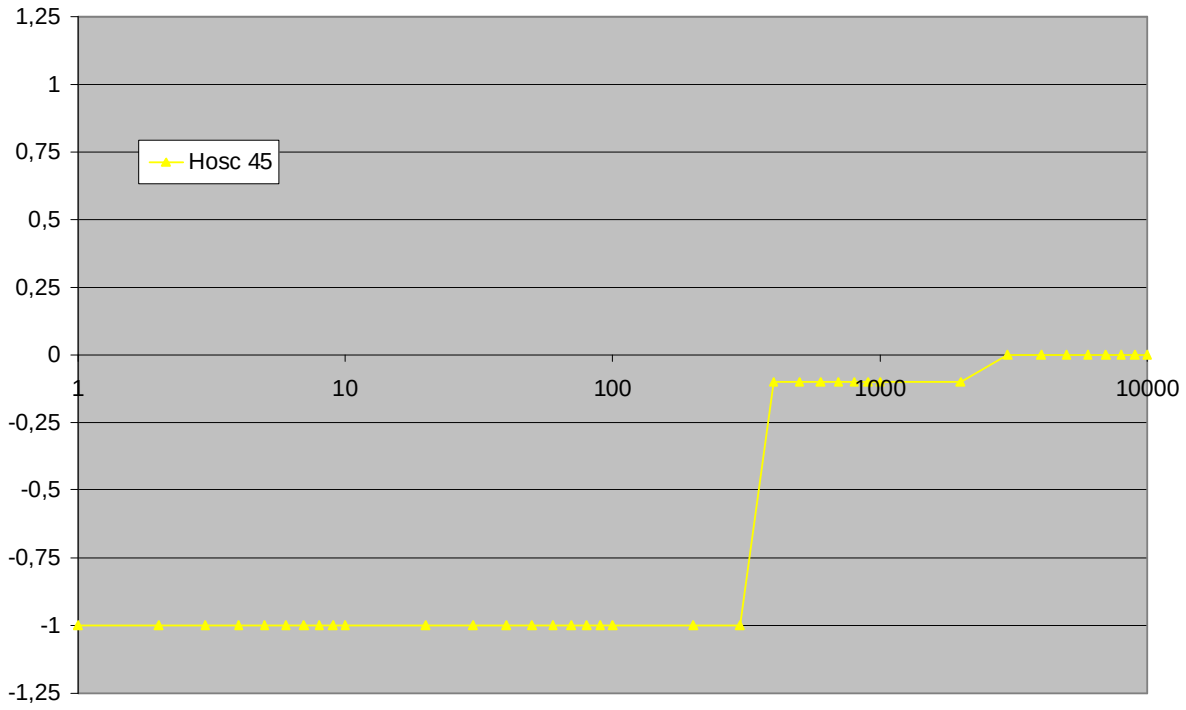
Gr. 8.4-3 - T-test na funkci 3 proměnných v závislosti na počtu iterací



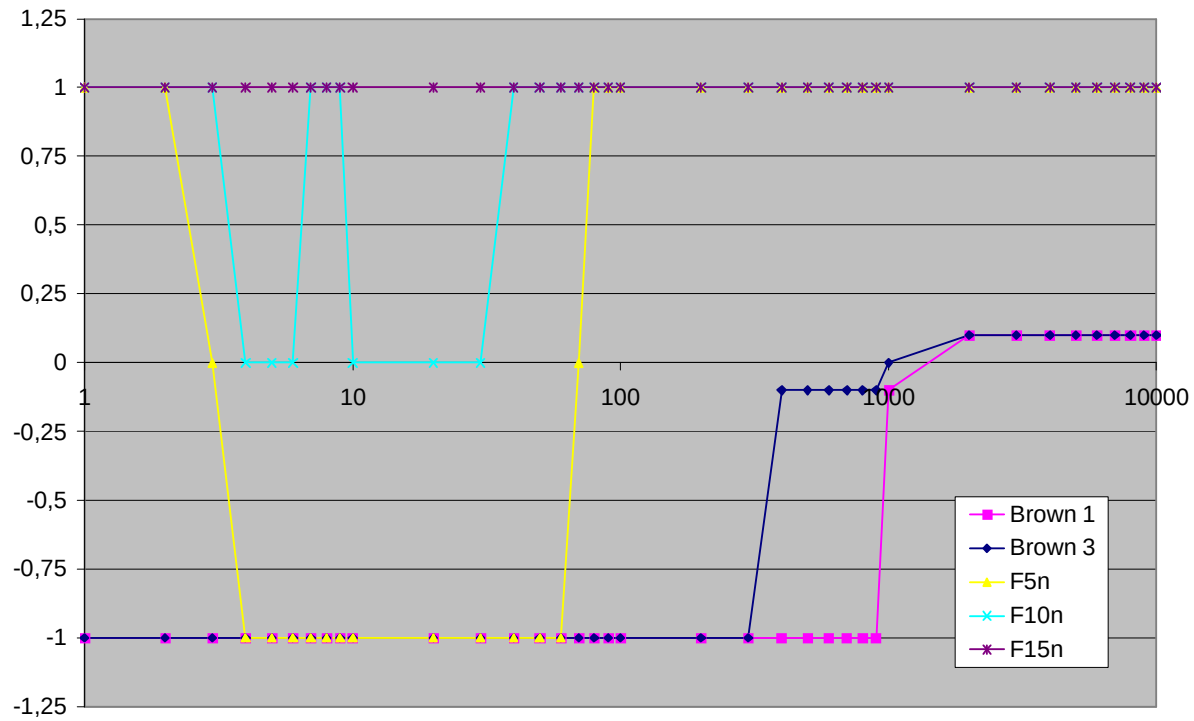
Gr. 8.4-4 - T-test na funkcích 4 proměnných v závislosti na počtu iterací



Gr. 8.4-5 -T-test na funkci 6 proměnných v závislosti na počtu iterací



Gr. 8.4-6 - T-test na funkci 10 proměnných v závislosti na počtu iterací

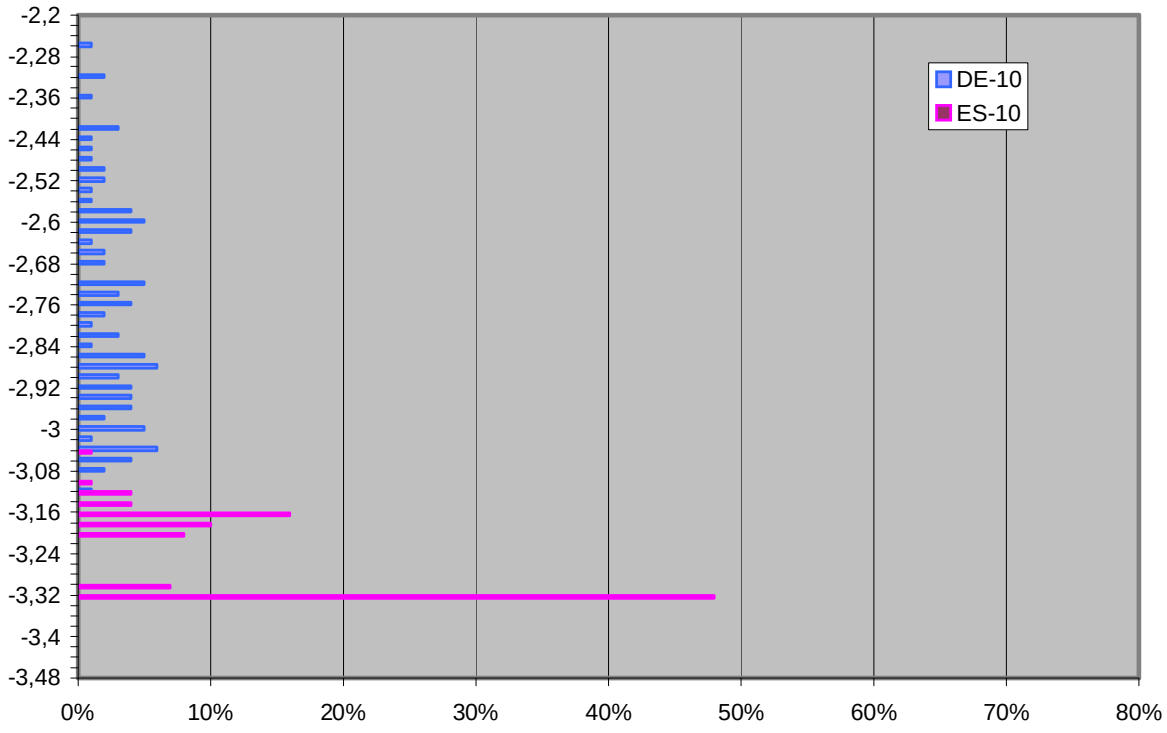


Gr. 8.4-7 - T-test na funkcích 20 proměnných v závislosti na počtu iterací

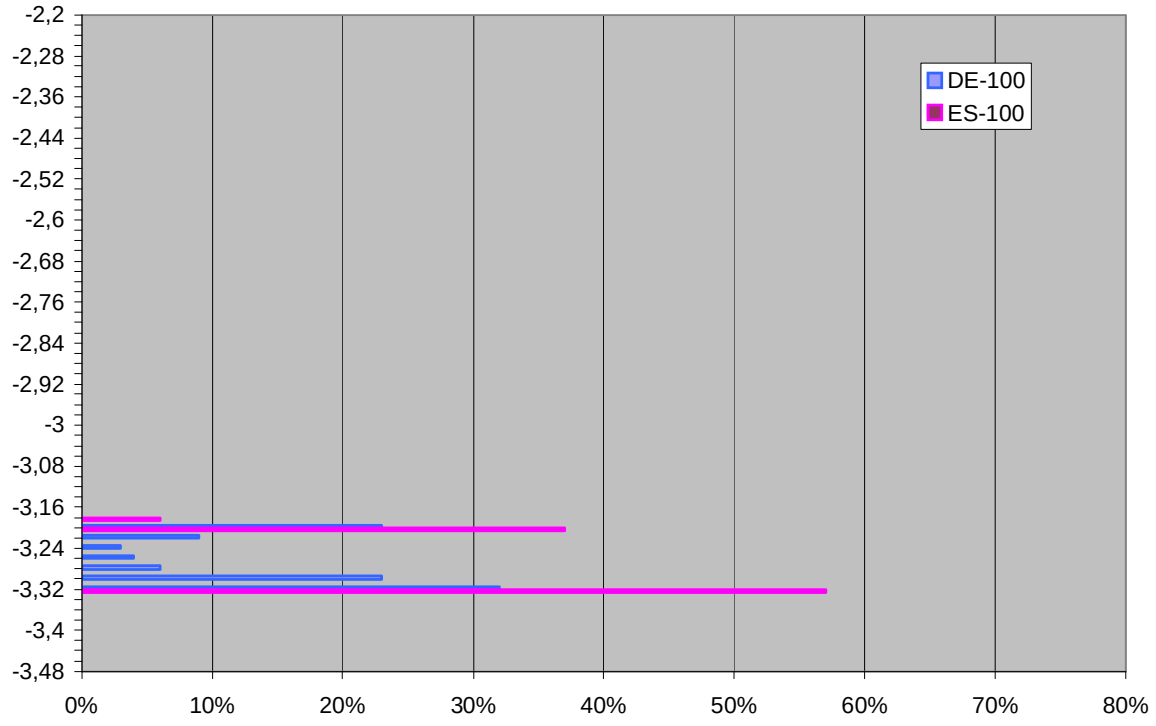
Z většiny uvedených grafů je zřejmé, že nelze jednoznačně říci, že je jedna metoda lepší než druhá. Co však pozorné oko uvidí je, že při nízkém počtu iterací dává většinou lepší výsledky ES a při vyšším DE. Toto povšimnutí blíže rozebereme v dalších kapitolách.

8.5 Vyhodnocení v jednotlivých řezech

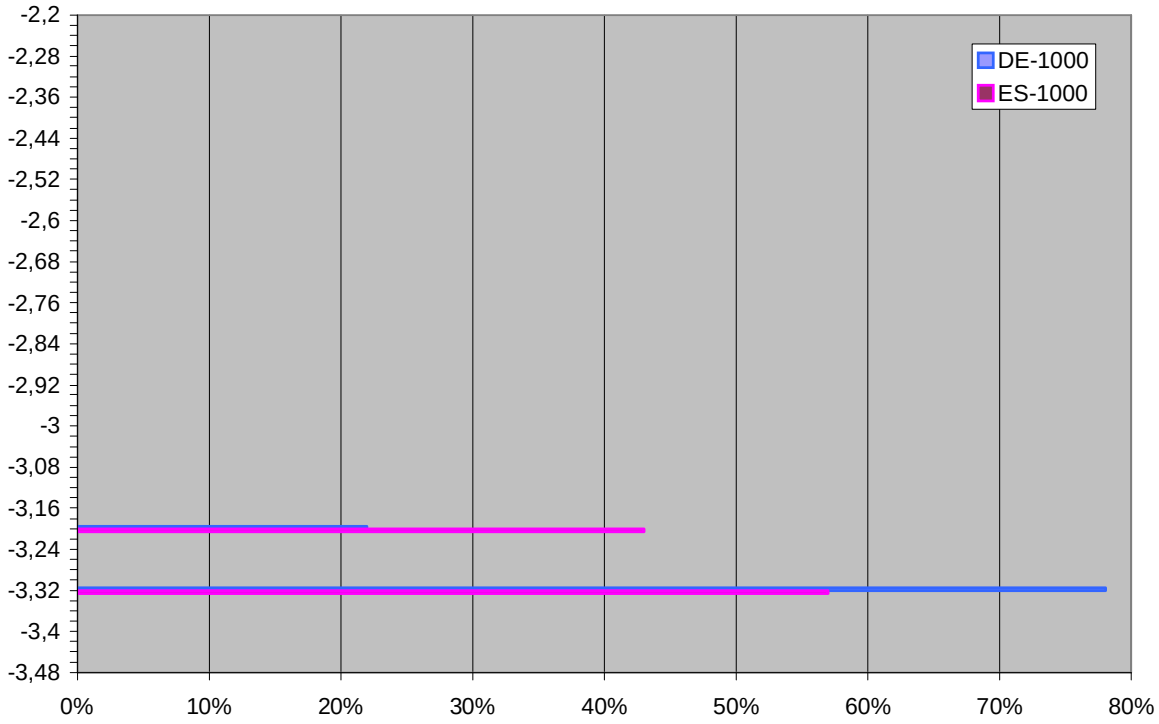
Jestliže vezmeme výsledky z funkce Hartman 2, a uděláme rozložení hodnot při 10, 100 a 1000 iteracích, tak získáme velmi zajímavé rozložení hodnot. Na Gr. 8.5-1 vidíme procentuální rozložení hodnot při 10 iteracích. Na svislé ose je dosažená hodnota. Je zřejmé, že diferenciální evoluce (modrá) se teprve začíná orientovat, zatímco ES má mnohem užší rozložení výsledků. Při sto iteracích (Gr. 8.5-2) již jasně vidíme, že u ES jasně převládají pouze dvě hodnoty. Kde jedna je globální optimum (57%) a druhá je lokální minimum (37%). DE je při tomto počtu iterací „pouze“ z 32% v globálním optimu a z 23% v lokálním minimu. V ostatních případech se pohubuje někde mezi. V této situaci také T-test ukázal, že nelze oddělit tyto dva případy. Velmi důležitý je ale Gr. 8.5-3, který ukazuje situaci při 1000 iteracích. Zde je jednoznačně vidět „robustnost“ DE, která nespadla do lokálního minima tolikrát jako ES. Je zřejmé že DE dosáhla správného globálního optima v 78% a ES jen v 57%. Tento rozdíl je poměrně velký. Stojí ovšem za úvahu zda opakovaným restartem ES nelze dosáhnout lepších výsledků. Například jestliže bychom 10x restartovali ES a vždy ji nechali dojít pouze do 10 iterací a ona nám pokaždé se 48% pravděpodobností našla globální optimum, pak pravděpodobnost, že optimum nenajdeme je rovna $0{,}52^{10} = 0{,}0014$. To znamená mnohem méně než 1 %, a celkem jsme provedli jen 100 iterací. Tento závěr ovšem nelze generalizovat. Je to jen spekulativní úvaha.



Gr. 8.5-1 – procentuální rozložení nalezených hodnot po 10 iteracích na funkci Hartman 2

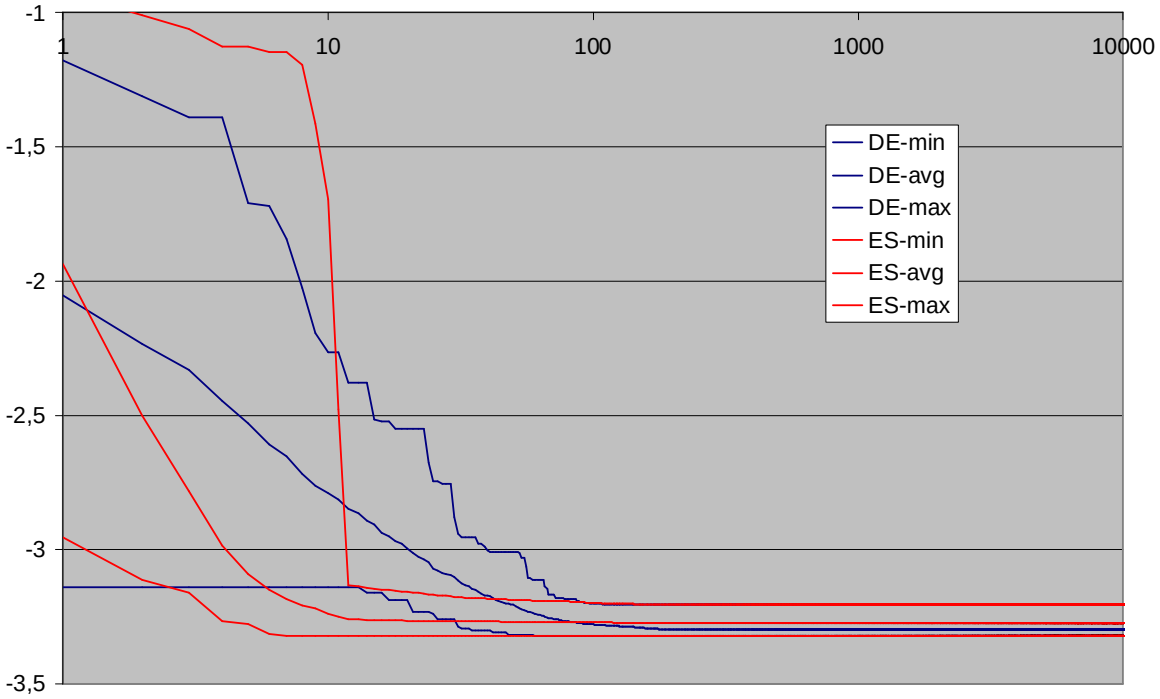


Gr. 8.5-2 – procentuální rozložení nalezených hodnot po 100 iteracích na funkci Hartman 2



Gr. 8.5-3 – procentuální rozložení nalezených hodnot po 1000 iteracích na funkci Hartman 2

Dále si můžeme ukázat vývoj nalezených hodnot funkce s počtem iterací. Na Gr. 8.5-4 můžeme vidět, jak postupně klesá funkční hodnota nalezených bodů. U každé metody máme znázorněny 3 hodnoty. Jedná se o minimum, maximum a průměr v dané iteraci. Jak je vidět u ES již mezi iterací 100 a 10 000 neprobíhají žádné změny. Proto nemá smysl tuto metodu iterovat tak dlouho. Naopak u DE probíhají změny zhruba do 1000 iterace, pak se opět již nic neděje. Proto v předešlém odstavci nebyly zobrazovány řezy v 10 000 iteracích, byly by zcela stejné jako při 1000 iteracích.



Gr. 8.5-4– rozptyl nalezených hodnot v závislosti na počtu iterací na funkci Hartman 2

Pro přehlednost můžeme ještě ukázat Tab. 8.5-1, která názorně ukazuje, která metoda na dané funkci při určitém počtu iterací má nižší průměrnou hodnotu. Jestliže je pole červené, pak je to DE. Zelená barva pole značí, že na daném segmentu je lepší ES. Jestliže pole nemá barvu, pak statistický T-test nedokázal rozhodnou na hladině pravděpodobnosti 95%, která metoda je lepší.

	10	20	30	40	50	60	70	80	90	100	200	300	400	500	600	700	800	900	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
F1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F3	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Branin	-1	-1	-0	-0	-0	-0	-0	-0	-0	-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Camelback	-1	-0	-0	-0	-0	-0	-0	-0	-0	-0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Goldprice	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Pshubert 1	-1	-1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
Pshubert 2	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
Quartic	-1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shubert	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hartman 1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shekel 1	-1	-1	-1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Shekel 2	-1	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Shekel 3	-1	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Hartman 2	-1	-1	-1	-1	-1	-1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Hosc 45	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-0	-0	-0	-0	-0	-0	-0	-0	0	0	0	0	0	0	0	0
Brown 1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-0	0	0	0	0	0	0	0	0	0
Brown 3	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-0	-0	-0	-0	-0	-0	0	0	0	0	0	0	0	0	0	0
F5n	-1	-1	-1	-1	-1	-1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F10n	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F15n	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tab. 8.5-1 – znázornění, která metoda nalezla lepší výsledky na dané funkci v závislosti na počtu iterací (-1 ES, +1 DE, 0 pat).

9 Test DE vs. ES na testovacích konstrukcích

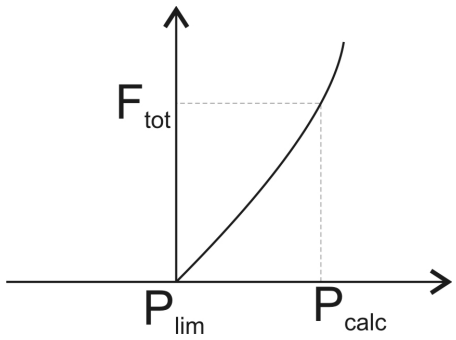
Abychom netestovali algoritmy jen na funkcích, které jsou čistě matematické, tak jsme provedli testy i na několika konstrukcích. Jedná se o jednoduché konstrukce, které již byly v literatuře testovány a jsou u nich známé výsledky z ostatních metod.

9.1 Testovací konstrukce

V následujících odstavcích se pokusíme na několika jednoduchých (až elementárních) konstrukcích otestovat algoritmy. Budeme měnit pouze průřezy prutů, proto se jedná o tzv. rozměrovou optimalizaci[*]. Na těchto konstrukcích se lépe ukáže praktická použitelnost algoritmů.

[*] z ang. size optimization

Zde již řešíme skutečné problémy, včetně všech dopadů. Jedním z prvních faktorů, s kterými se musíme vyrovnat, jsou omezující podmínky. V našem případě se jedná o omezení průhybů a napětí v jednotlivých místech konstrukce. Tento problém budeme řešit pomocí penalizačních funkcí [4]. V našem případě jsme použili jednu z nejprimitivnějších penalizačních funkcí ve tvaru $f_{tot} = f * (P_{calc}/P_{lim})^n$, kterou můžeme vyjádřit Obr. 9.1-1. Hodnota parametru „*n*“ byla v počítaných úlohách položena n = 2, *f* je funkční hodnota, P_{calc} je hodnota parametru, která vyšla z výpočtu a P_{lim} *je* maximální povolená hodnota.



Obr. 9.1-1 – vývoj penalizační funkce

Specifika těchto testů spočívají také v tom, že se jedná i o úlohy diskrétní, které tvoří samostatnou kapitolu na rozdíl od úloh nad spojitým prostorem.

Všechny následující konstrukce byly převzaty z článku [6].

9.2 Testovací konstrukce „the ten-bar truss“

Konstrukce je vyobrazená na Obr. 9.2-1. Vše je v imperiálních jednotkách. Je to tím, že příklady přebíráme ze zahraniční literatury v původním znění, aby naše výsledky bylo možno srovnat s výsledky jiných prací.

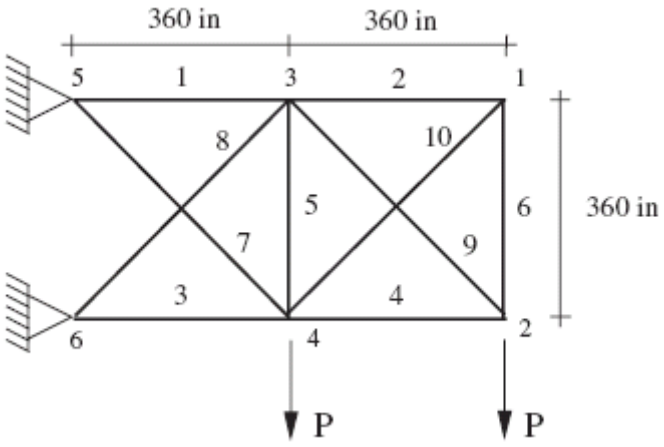
Konstrukce je složena z 10 prutů. Naším cílem je minimalizace hmotnosti konstrukce při splnění omezujících podmínek. Povolené limitní napětí je ± 25 kpsi* a průhyb je limitován na 2 in ve směru *x* i ve směru *y*. Modul pružnosti je uvažován $E = 10^4$ ksi a hustota materiálu je 0,1 lb / in^3. V uzlech 4 a 2 je aplikováno zatížení P o velikosti 100 kips.

Uvažujeme případ spojitý a diskrétní. V prvním případě uvažujeme, že průřez prutů může být libovolné reálné číslo, minimálně však 0,1 in^2†. V případě diskrétního problému

* tlak tělesa o váze 1000 liber na čtvereční palec, 1 kpsi ≈ 6 894 KPa

† čtverečný palec, 1 in^2 = 645,16 mm^2

můžeme vybírat profily pouze ze sady čtyřiceti dvou předem daných hodnot. Hodnoty jsou: 1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.48, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.87, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90. 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50. Tím máme definováno kompletní zadání .



Obr. 9.2-1 – konstrukce „the ten-bar truss"

Pro řešení úlohy nad spojitým prostorem jsou některá známá řešení vypsaná v Tab. 9.2-1. v posledním sloupci je uvedeno i naše nejlepší řešení, které bylo vypočteno za pomoci DE. Naše řešení splňuje okrajové podmínky (posun uzlu je roven 2 in a největší napětí je 24.998 kpsi. Dále již budeme rozebírat pouze řešení diskrétní, které jsou u těchto typů konstrukcí obvyklejší.

A_j	Ref. [7]	Ref. [8]	Ref. [9]	Ref. [10]	Ref. [12]	Ref. [13]	Naše výsledky
1	31.350	30.570	25.73	33.432	30.416	30.500	30.4725
2	0.100	0.369	0.109	0.100	0.128	0.100	0.1002
3	20.030	23.970	24.85	24.260	23.408	23.290	23.1728
4	15.600	14.730	16.35	14.260	14.905	15.428	15.2100
5	0.140	0.100	0.106	0.100	0.101	0.100	0.1001
6	0.240	0.364	0.109	0.100	0.101	0.210	0.5614
7	8.350	8.547	8.70	8.388	8.696	7.649	7.4641
8	22.210	21.110	21.41	20.740	21.084	20.980	21.0887
9	22.060	20.770	22.30	19.690	21.077	21.818	21.5265
10	0.100	0.320	0.122	0.100	0.186	0.100	0.100
W	**5112.00**	**5107.30**	**5095.65**	**5089.00**	**5084.90**	**5080.00**	**5060.9**

Tab. 9.2-1 - nalezená řešení nad spojitým prostorem na konstrukci „the ten-bar truss“

V Tab. 9.2-2 jsou uvedeny výsledky ze třech různých zdrojů pro diskrétní hodnoty průřezu prutů. V posledním sloupci jsou uvedeny hodnoty, které jsme spočetli my. Jak je vidět, tak ve všech třech referencích jsou uvedené hodnoty průhybu vyšší než je povoleno. Pro zajištění správnosti jsme dali do našeho algoritmu vektory z [17] a vyšly nám stejné hodnoty jaké jsou uvedeny. Při tom samém testu s hodnotami z [18] nám vyšly zcela odlišné hodnoty (jak posunutí, tak hmotnosti konstrukce). Z tohoto důvodu jsme se rozhodli vypočítat hmotnost konstrukce s pruty z [18] průkazně v Tab. 9.2-3. Stejná hmotnost nám vyšla i v algoritmu. Proto považujeme hodnoty z reference [18] za chybné a dále s nimi nebudeme pracovat. Při přepočtení dat z [9] jsme se dostali ke shodným výsledkům. Proto považujeme algoritmus za správný a vypočtená data za věrohodná. Jak je vidět v

Tab. 9.2-2, tak jsme se dostali s hmotností konstrukce níže než uvedené reference až na výjimku [18] a navíc je splněna podmínka posunutí. Je nutno zmínit, že se nikde nepíše o napětí, i když ve formulaci úlohy byla omezující podmínka. Je to z důvodu velké rezervy v této podmínce. V žádném z uvedených příkladů jsme se k dovoleným 25 ksi ani nepřiblížili (u našich dat je to cca 14 ksi).

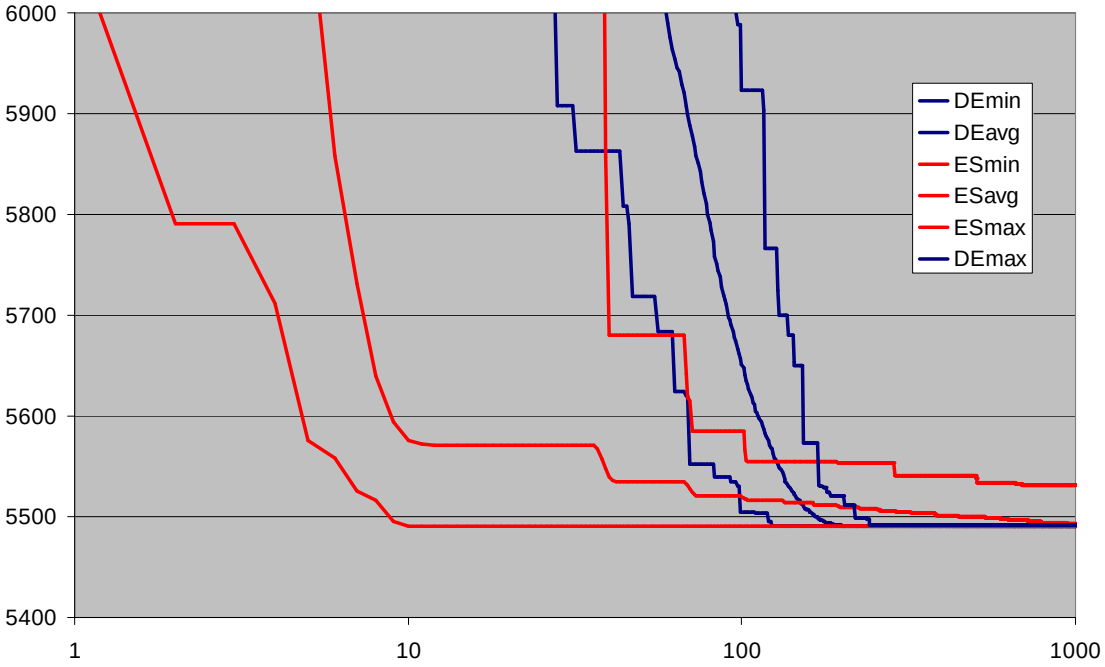
A_j	Reference [17]	Reference [18]	Reference [9]	Naše výsledky
1	33.50	33.50	33.50	33.5
2	1.62	1.62	1.62	1.62
3	22.00	22.00	22.00	22.90
4	15.50	14.20	13.90	14.20
5	1.62	1.62	1.62	1.62
6	1.62	1.62	1.62	1.62
7	14.20	7.97	7.97	7.97
8	19.90	22.90	22.90	22.9
9	19.90	20.00	22.90	22.0
10	2.62	1.62	1.62	1.62
W	**5613.58**	**5458.3**	**5493.36**	**5490.738**
$\|u_{max}\|$	**2.00075**	**2.01227**	**2.00074**	1.9989

Tab. 9.2-2 -nalezená řešení s diskrétními průřezy na konstrukci „the ten-bar truss“

Plocha [in^2]	Délka [in]	Hustota [lb/in^3]	Hmotnost [lb]
33,50	360	0,1	1206
1,62	360	0,1	58,32
22,00	360	0,1	792
14,20	360	0,1	511,2
1,62	360	0,1	58,32
1,62	360	0,1	58,32
7,97	509,1168825	0,1	405,7661553
22,90	509,1168825	0,1	1165,877661
20,00	509,1168825	0,1	1018,233765
1,62	509,1168825	0,1	82,47693496
		Celkem	5356,514516

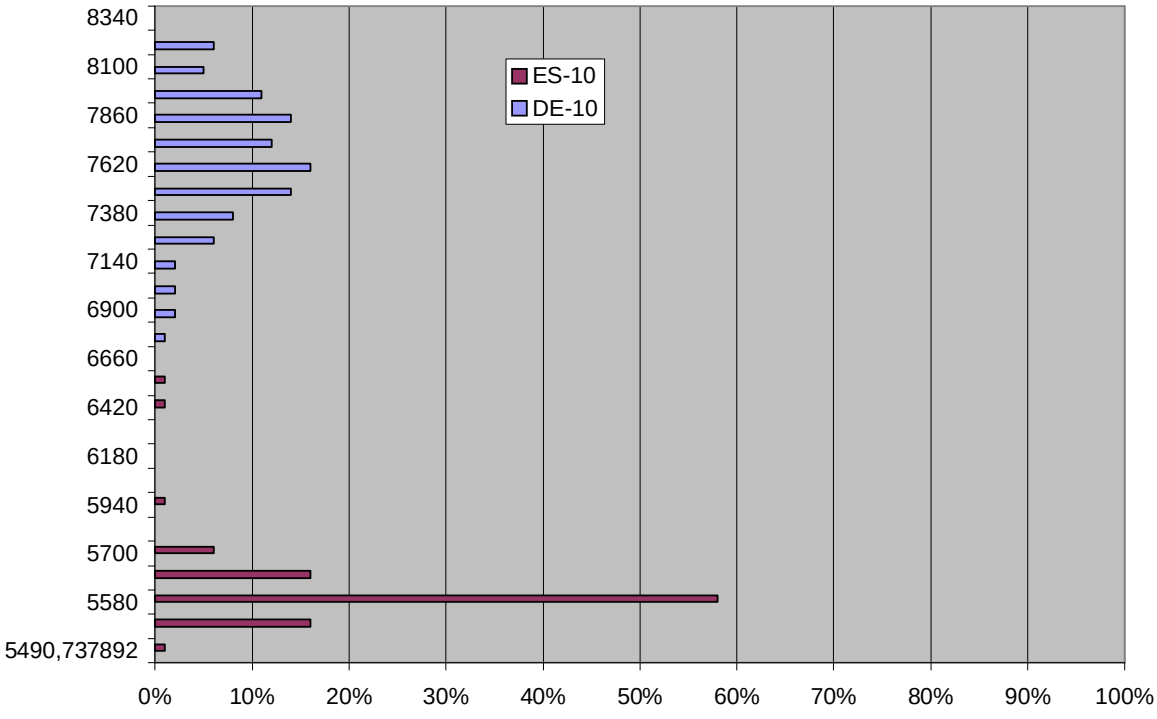
Tab. 9.2-3 – výpočet hmotnosti konstrukce s pruty dle [18]

Na Gr. 9.2-1 je znázorněn vývoj dosažených hodnot v závislosti na počtu iterací. Je zde velmi názorně vidět jak ES již v prvních desítkách iterací získává zajímavé hodnoty, zatímco DE až kolem sta iterací.

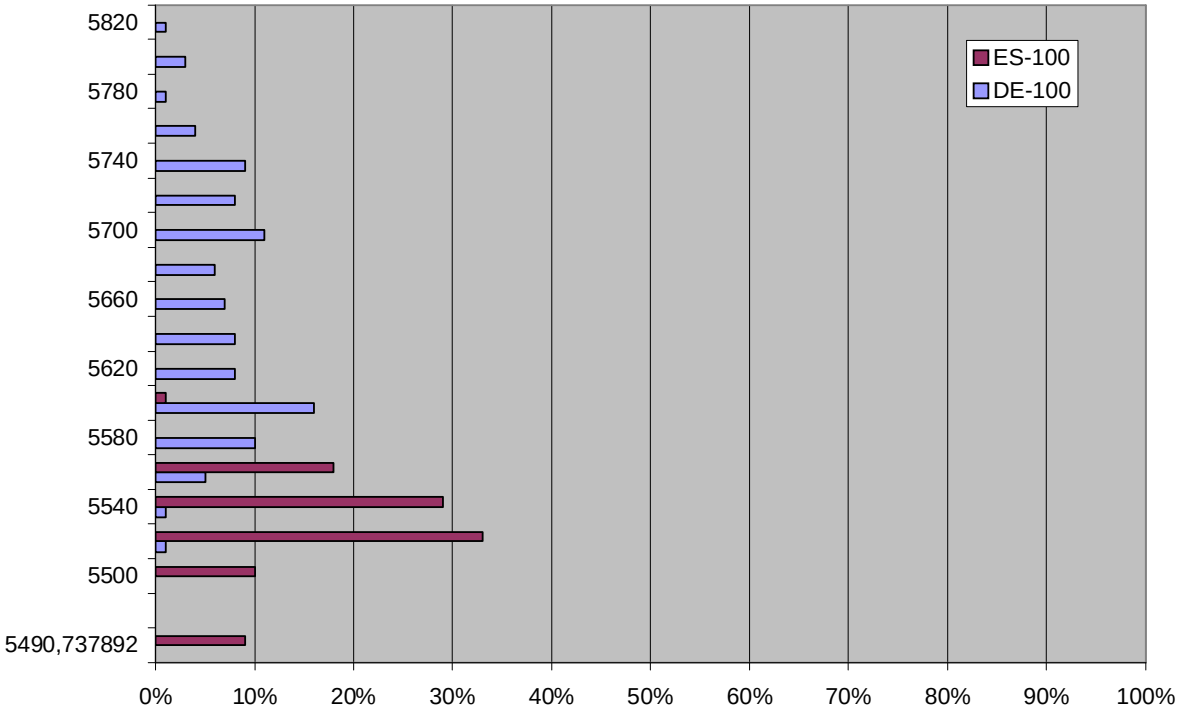


Gr. 9.2-1 – rozložení nalezených hodnot na konstrukci „the ten-bar truss“ v závislosti na počtu iterací

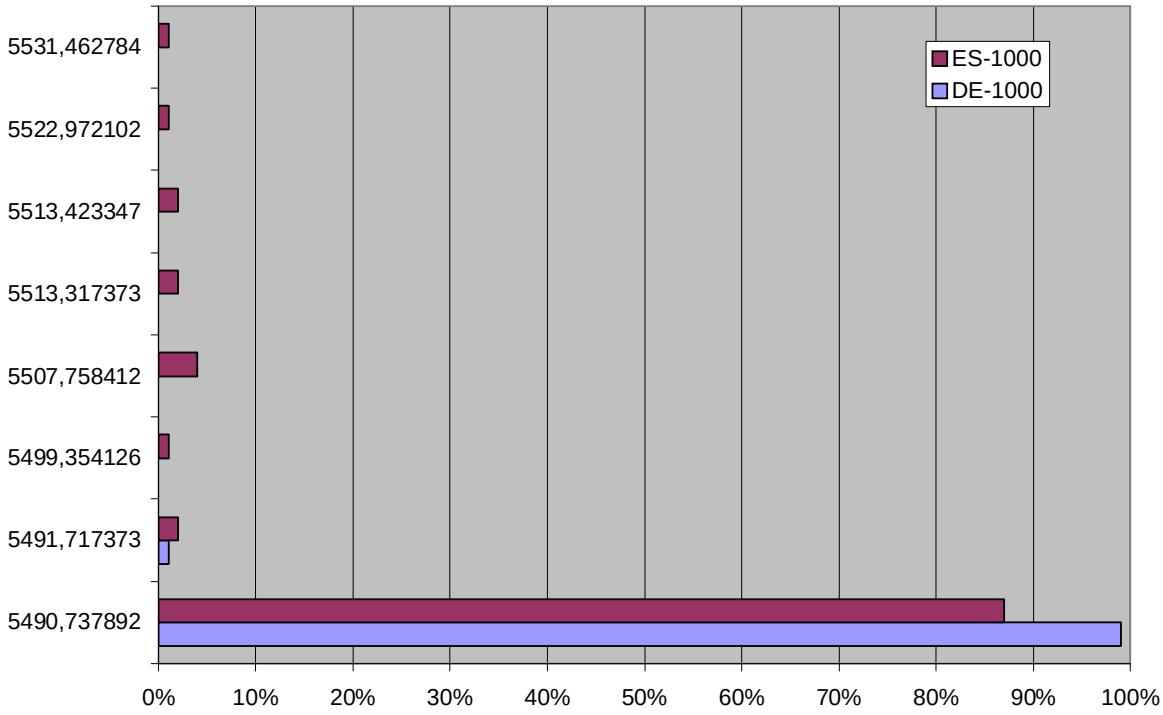
Neméně zajímavé jsou dosažené hodnoty v jednotlivých iteracích. Rozložení těchto hodnot při 10 iteracích je na Gr. 9.2-2. Je zde velmi dobře patrné, jaký náskok má ES před DE. Na Gr. 9.2-3 je vidět stejné rozložení při sto iteracích. A na posledním Gr. 9.2-4 je vidět rozložení při tisíci iteracích. Je ještě nutné upozornit na rozdílné hodnoty na svislé ose.



Gr. 9.2-2 – procentuální rozložení nalezených hodnot na konstrukci „the ten-bar truss“ při 10 iteracích



Gr. 9.2-3 – procentuální rozložení nalezených hodnot na konstrukci „the ten-bar truss“ při 100 iteracích



Gr. 9.2-4 – procentuální rozložení nalezených hodnot na konstrukci „the ten-bar truss“ při 1000 iteracích

9.3 Testovací konstrukce „the 25-bar truss“

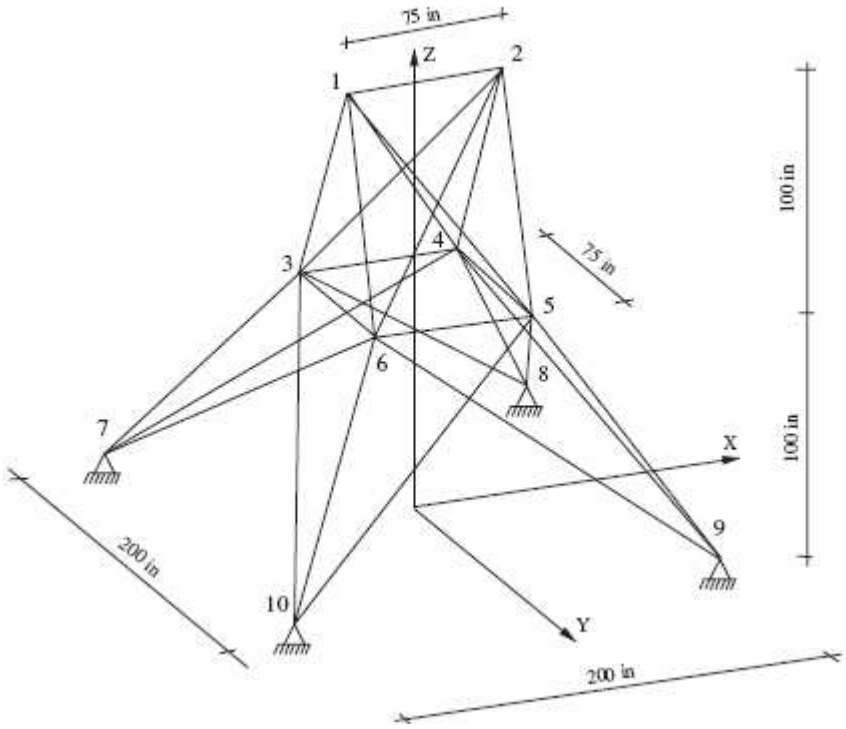
Tato testovací konstrukce je složena z 25 prutů a je ukázána na Obr. 9.3-1. Okrajové podmínky vyžadují, aby napětí v prutech bylo v intervalu [-40, +40] kpsi a aby maximální posun bodu 1 a 2 byl menší nebo roven 0.35 in v každém směru. Zatížení jednotlivých uzlů je dle Tab. 9.3-1. Jedná se o diskrétní úlohu, tudíž budeme průřezy prutů vybírat z definovaných hodnot (0.1, 0.2, 0.3, 0.4 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4). Navíc jsou jednotlivé pruty sloučeny do osmi skupin (dle Tab. 9.3-2). Tím klesá počet proměnných, které můžeme měnit.

Uzel	Fx	Fy	Fz
1	1	-10.0	-10.0
2	0	-10.0	-10.0
3	0.5	0	0
6	0.6	0	0

Tab. 9.3-1 – zatížení v uzlech [kips]

Skupina	Propojení uzlů
A_1	1-2
A_2	1-4 ,2-3, 1-5, 2-6
A_3	2-5, 2-4, 1-3, 1-6
A_4	3-6, 4-5
A_5	3-4, 5-6
A_6	3-10, 6-7, 4-9, 5-8
A_7	3-8, 4-7, 6-9, 5-10
A_8	3-7, 4-8, 5-9, 6-10

Tab. 9.3-2 – skupiny prutů



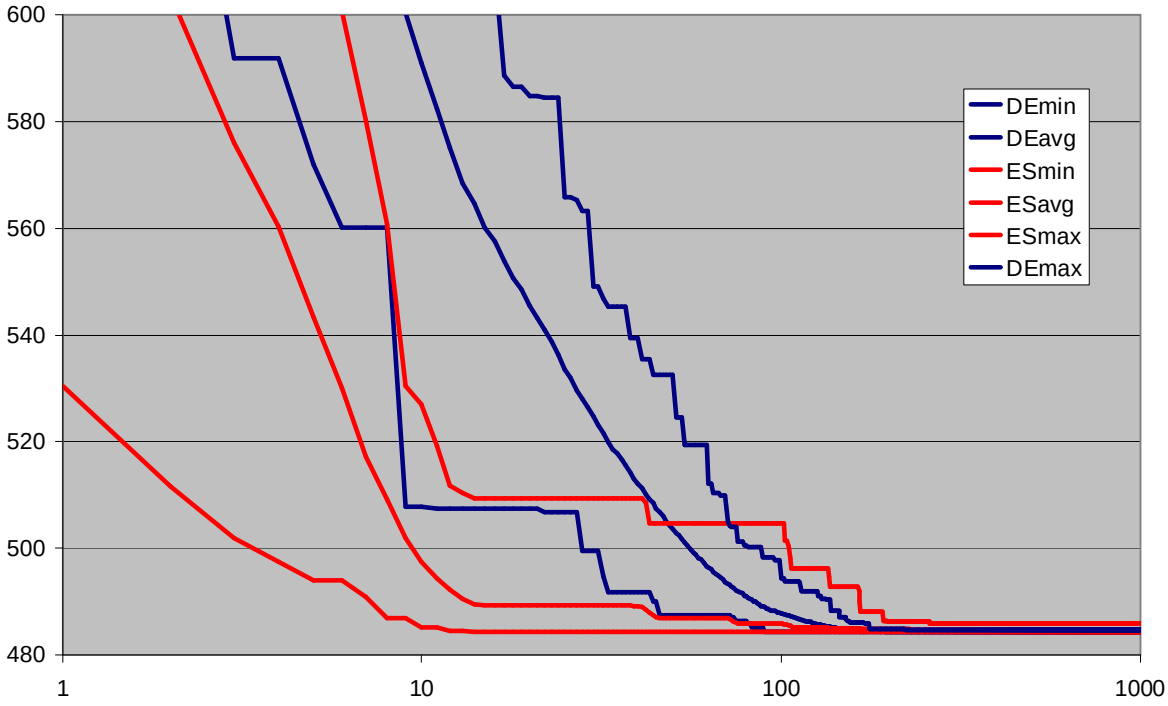
Obr. 9.3-1 – konstrukce „the 25-bar truss“

Výsledky, které jsme dostali, jsme srovnali s výsledky, které se objevily v dostupné literatuře. Souhrn můžeme vidět v Tab. 9.3-3. Jak je vidět, dostali jsme ještě lepší hodnoty než uvádí literatura. I zde jsme zkoušeli pustit algoritmus s jednotlivými plochami z citované literatury, abychom ověřili správnost výpočtu. V případě zdrojů [11], [14], [15] jsme se dostali ke stejným výsledkům. Tím se dá říci, že algoritmus počítá správně. V případě reference [16] jsme obdrželi mírně jiné hodnoty a proto si myslíme že tato reference je chybně vypočtená.

Proměnná	[11]	[14]	[15]	[16]	Náš výsledek
A_1	0.1	0.1	0.1	0.2	0.1
A_2	1.8	1.9	1.2	0.5	0.4
A_3	2.3	2.6	3.2	3.4	3.4
A_4	0.2	0.1	0.1	0.1	0.1
A_5	0.1	0.1	1.1	1.5	2.2
A_6	0.8	0.8	0.9	0.9	1
A_7	1.8	2.1	0.4	0.6	0.4
A_8	3.0	2.6	3.4	3.4	3.4
W	**546.01**	**562.93**	**493.80**	**486.29**	**484.33**
u_{y1}	-0.3481	-0.3486	-0.3499	-0.3495	-0.34795
u_{y2}	-0.3477	-0.3482	-0.3479	-0.3479	-0.34996

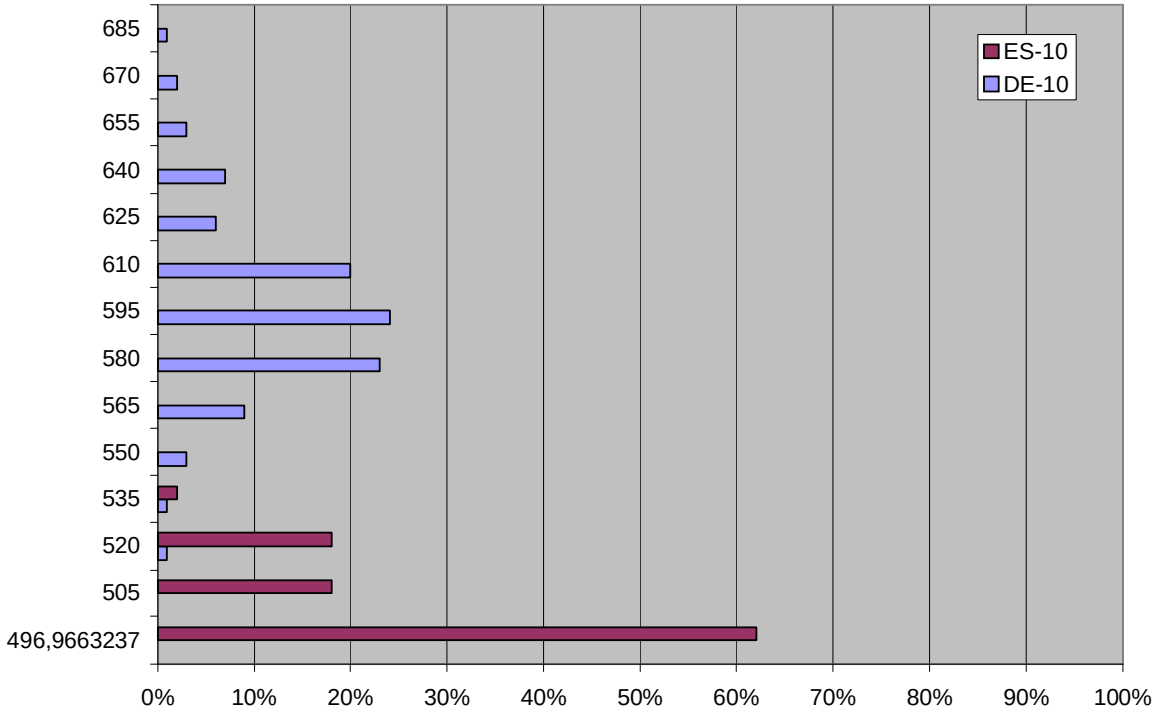
Tab. 9.3-3 – nalezená řešení na konstrukci „the 25-bar truss“

Na Gr. 9.3-1 můžeme vidět vývoj hodnot při jednotlivých iteracích. Stejně jako v předchozím případě je velmi pěkně vidět o kolik rychleji najde ES „dobrou“ hodnotu.

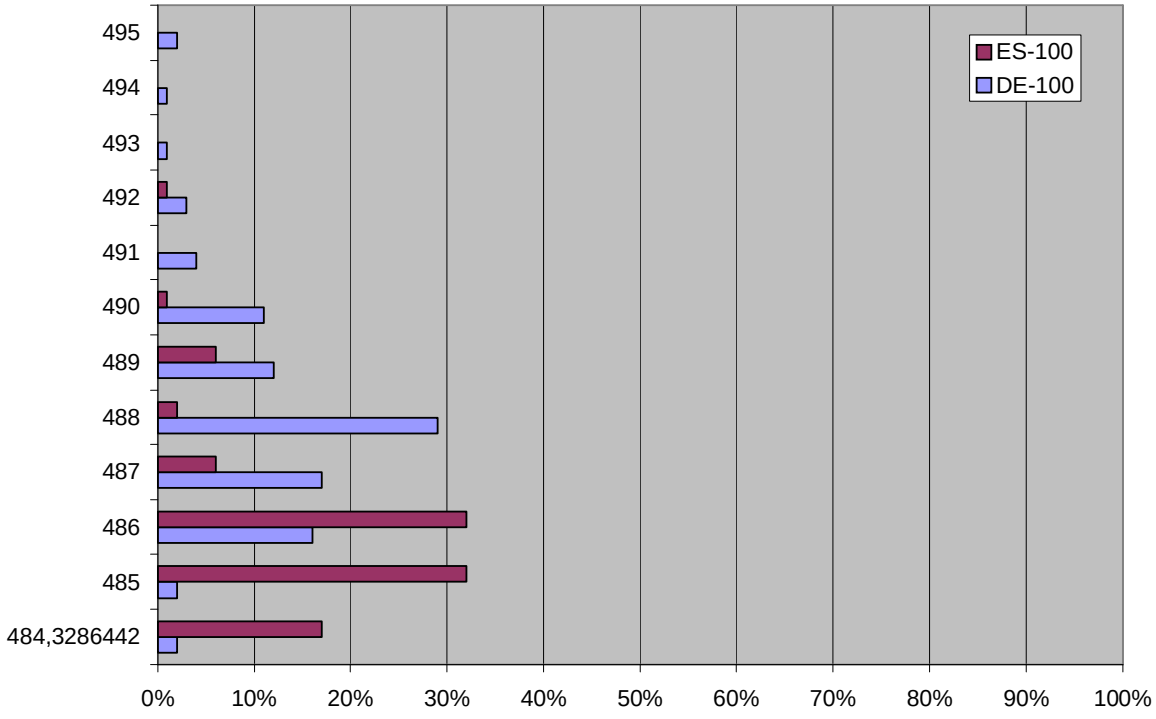


Gr. 9.3-1 – procentuální rozložení nalezených hodnot na konstrukci „the 25-bar truss“

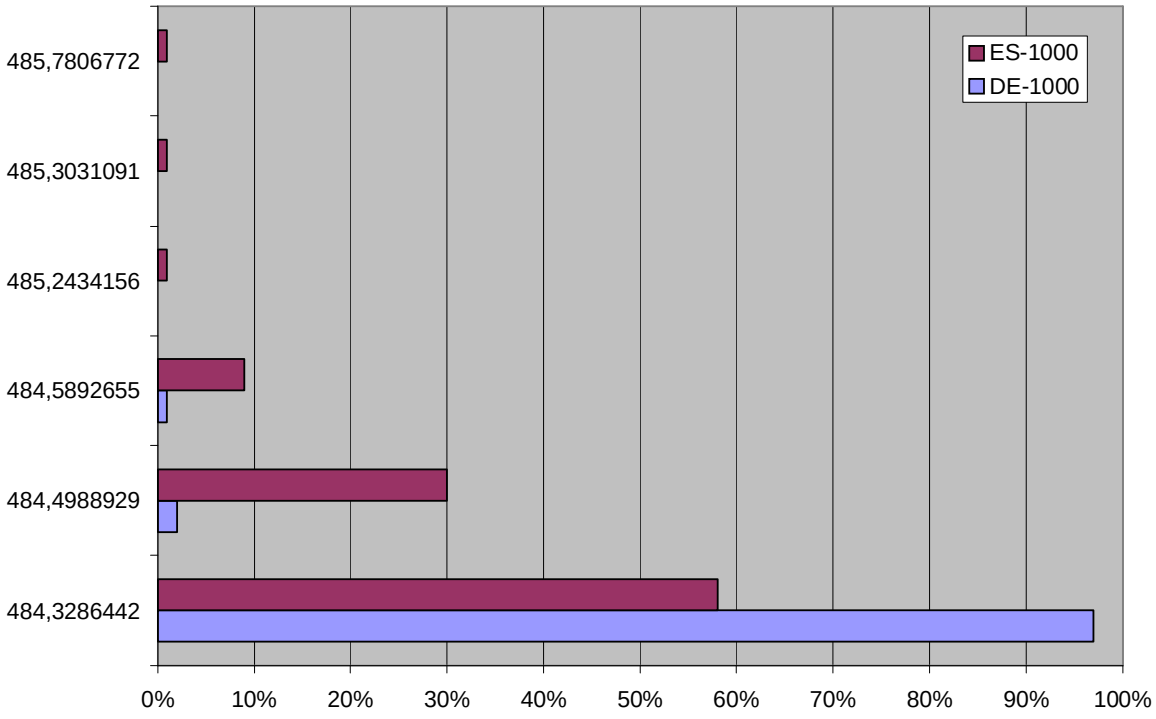
Můžeme si i ukázat rozložení nalezených hodnot v jednotlivých iteracích. Stav při 10 iteracích ukazuje Gr. 9.3-2. Dále na Gr. 9.3-3 vidíme stav při sto iteracích. I při tomto počtu se dá říci, že ES je na tom lépe. V posledním ze série grafů (Gr. 9.3-4) je znázorněno rozložení nalezených hodnot při tisíci iteracích. Zde již zcela jasně převládá DE.



Gr. 9.3-2 - procentuální rozložení nalezených hodnot na konstrukci „the 25-bar truss“ při 10 iteracích



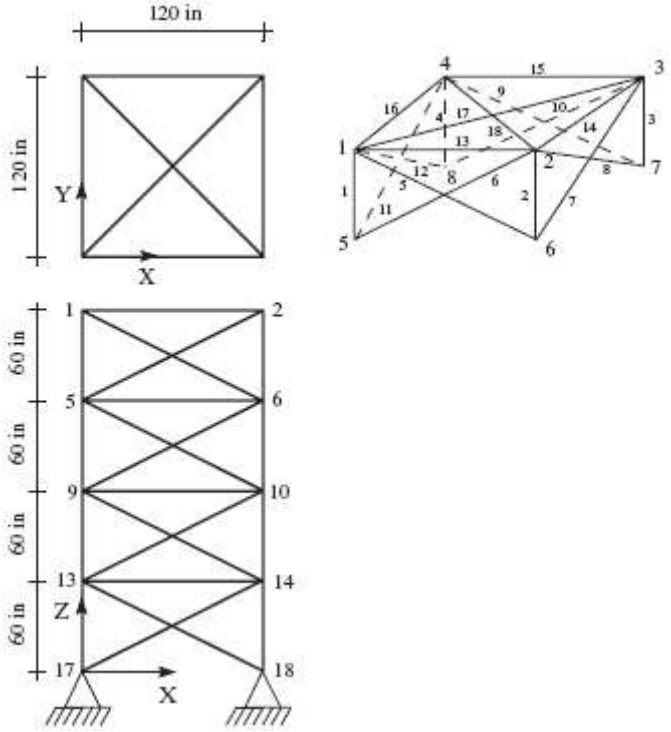
Gr. 9.3-3 - procentuální rozložení nalezených hodnot na konstrukci „the 25-bar truss“ při 100 iteracích



Gr. 9.3-4 - procentuální rozložení nalezených hodnot na konstrukci „the 25-bar truss“ při 1000 iteracích

9.4 Testovací konstrukce „the 72-bar truss“

Další testovaná konstrukce je prostorová příhrada ukázána na Obr. 9.4-1. Na této konstrukci budeme optimalizovat hmotnost celé konstrukce. Jako proměnné jsou průřezy prutů. Pruty jsou uspořádány do 16 skupin (Tab. 9.4-1), čímž se snižuje počet proměnných, ale zvyšuje obtížnost. Minimální průřez prutu je stanoven na 0.1 in^2, maximální není nastaven a mezi těmito mezemi se mění průřez spojitě. Posun každého bodu je limitován na 0,25 in v každém směru. Maximální napětí v prutu musí být v rozsahu [-25, 25] ksi. Hustota materiálu je 0,1 lb/in^3 a modul pružnosti 10^4ksi. V této konstrukci jsou definovány dva zatěžovací stavy (Tab. 9.4-2).



Obr. 9.4-1 – konstrukce „the 72 bar truss“

Skupina	Pruty
A_1	1,2,3,4
A_2	5, 6, 7, 8, 9, 10, 11, 12
A_3	13, 14, 15, 16
A_4	17, 18
A_5	19, 20, 21, 22
A_6	23, 24, 25, 26, 27, 28, 29, 20
A_7	31, 32, 33, 34
A_8	35, 36
A_9	37, 38, 39, 40
A_{10}	41, 42, 43, 44, 45, 46, 47, 48
A_{11}	49, 50, 51, 52
A_{12}	53, 54
A_{13}	55, 56, 57, 58
A_{14}	59, 60, 61, 62, 63, 64, 65, 66
A_{15}	67, 68, 69, 70
A_{16}	71, 72

Tab. 9.4-1 – seřazení prutů do skupin

Zatěžovací stav	Uzel	F_x	F_y	F_z
1	1	5	5	-5
2	1	0	0	-5
	2	0	0	-5
	3	0	0	-5
	4	0	0	-5

Tab. 9.4-2 – zatěžovací stavy (kips)

Zajímavostí na této konstrukci bude, že srovnáme nejen výkonnost algoritmů DE a ES, ale přidáme ještě jeden navíc. Nebude se jednat o žádný z běžných algoritmů, ale o modifikaci DE. Modifikace bude spočívat v následujícím. Nebudeme vytvářet celou zmutovanou populaci, ale pouze jednoho jedince, na něm pak provedeme crossover a modifukujeme populaci. Tím se tato metoda zařadí mezi „steady-state“ algoritmy. U verze DE2 je zachována velikost populace 10*D, kdežto u verze DE3 je velikost populace snížena na 2*D.

V Tab. 9.4-3 jsou ukázány naše dosažené výsledky ve srovnání s výsledky z některých referencí. Toto srovnání jsme dělali hlavně pro ověření správnosti výpočtu.

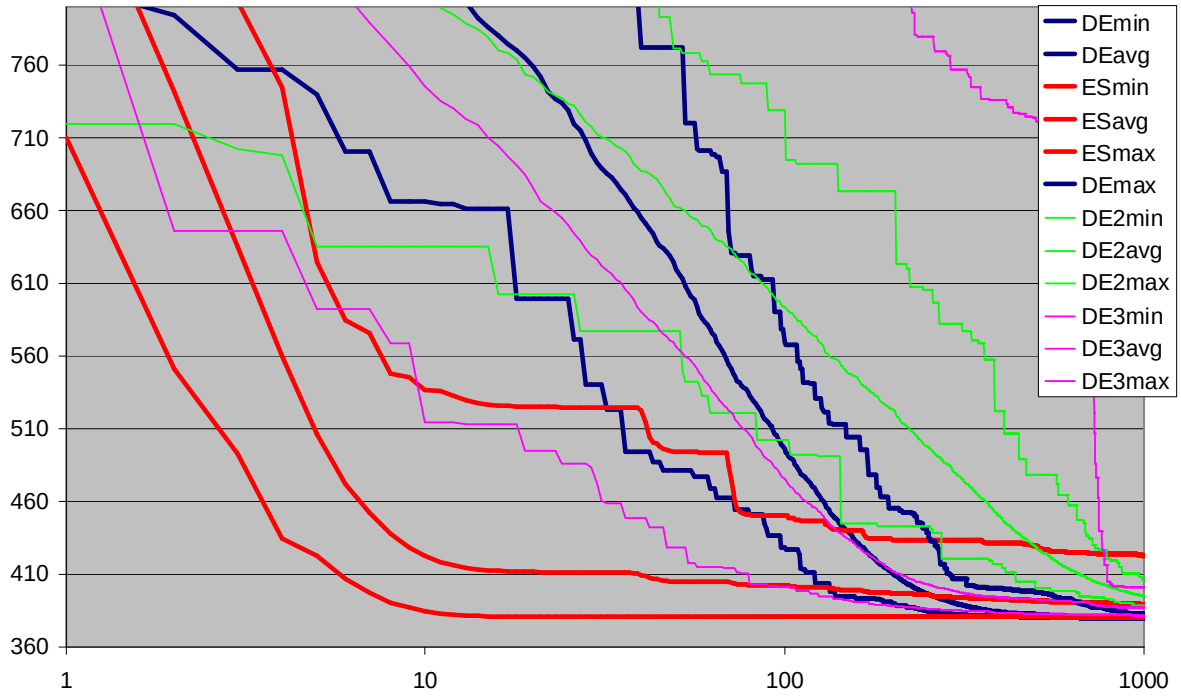
Srovnávali jsme hmotnost celé konstrukce. U každé reference jsme vypočetli hmotnost, posun uzlu i maximální napětí. S referencemi jsme srovnávali pouze hmotnost celé konstrukce. Jak je vidět, jsou zde pouze drobné odchylky způsobené zaokrouhlováním.

Proměnná	[20]	[7]	[10]	[15]	Náš výsledek
A_1	0.161	0.1492	0.1582	0.155	0.1565
A_2	0.557	0.7733	0.5936	0.532	0.5497
A_3	0.373	0.4534	0.3414	0.480	0.4092
A_4	0.506	0.3417	0.6076	0.530	0.5763
A_5	0.611	0.5521	0.2643	0.460	0.5181
A_6	0.532	0.6084	0.5480	0.530	0.5193
A_7	0.100	0.1000	0.1000	0.120	0.1000
A_8	0.100	0.1000	0.1509	0.165	0.1001
A_9	1.246	1.0235	1.1067	1.155	1.2878
A_{10}	0.524	0.5421	0.5793	0.585	0.5109
A_{11}	0.100	0.1000	0.1000	0.100	0.1000
A_{12}	0.100	0.1000	0.1000	0.100	0.1000
A_{13}	1.818	1.4636	2.0784	1.755	1.8842
A_{14}	0.524	0.5207	0.5034	0.505	0.5161
A_{15}	0.100	0.1000	0.1000	0.105	0.1002
A_{16}	0.100	0.1000	0.1000	0.155	0.1000
$W_{reference}$	381.2	395.97	388.63	385.76	
$W_{vypočteno}$	381.09	396.84	388.64	386.12	379.68
Napětí	25.008	25.522	25.005	24.743	24.998
U_{max}	0.24348	0.24974	0.29918	0.25599	0.25

Tab. 9.4-3 – srovnání výsledků na konstrukci „the 72-bar truss“

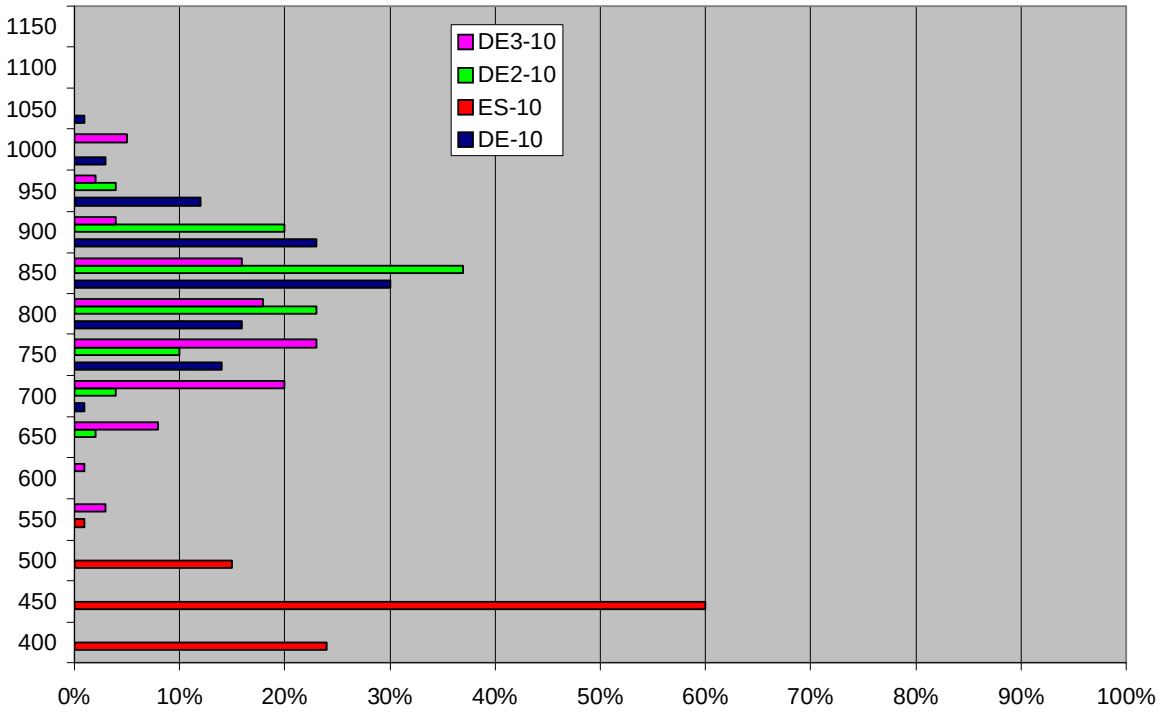
Na Gr. 9.4-1 můžeme vidět průběh nalezených hodnot po jednotlivých iteracích. Jak je vidět i na této konstrukci se chovají algoritmy jako na všech předešlých. Navíc zde máme ukázán modifikovaný algoritmus DE (na grafech je značen jako DE2 a DE3). Jak je vidět

modifikace DE přináší úspěch v začátcích, kde konverguje lépe, ale při 1000 iteracích je horší než původní DE a dokonce i než ES.

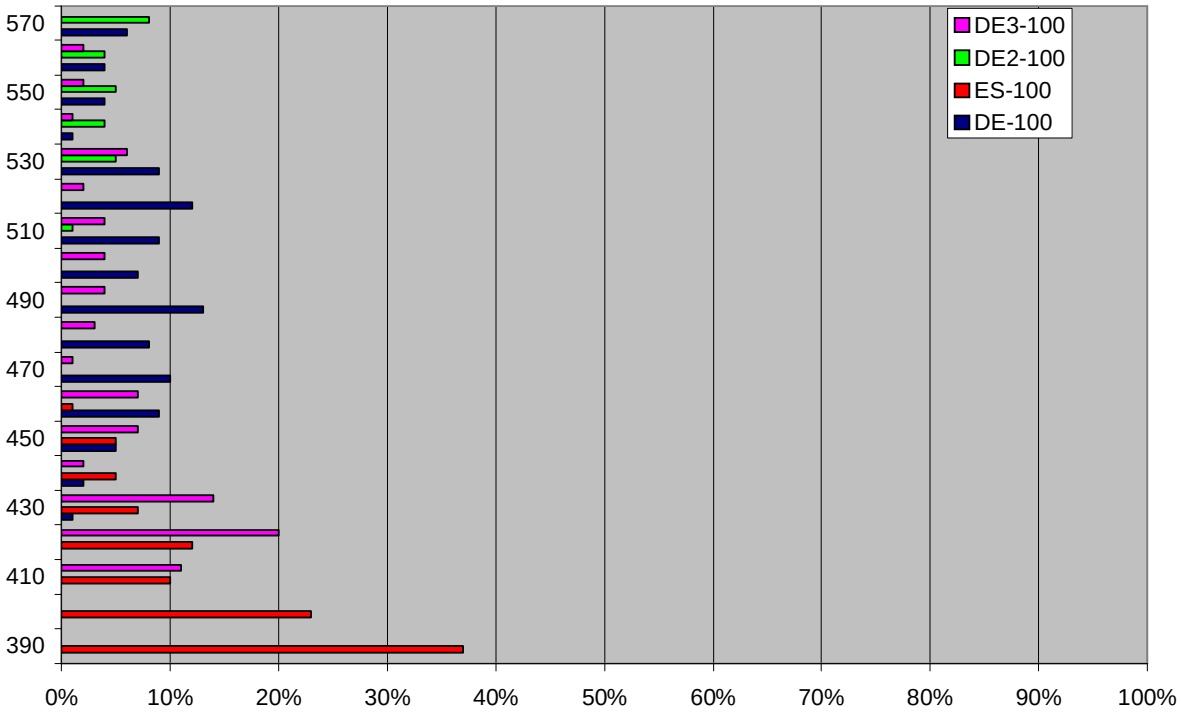


Gr. 9.4-1 – vývoj nalezených hodnot na konstrukci „the 72 bar truss“

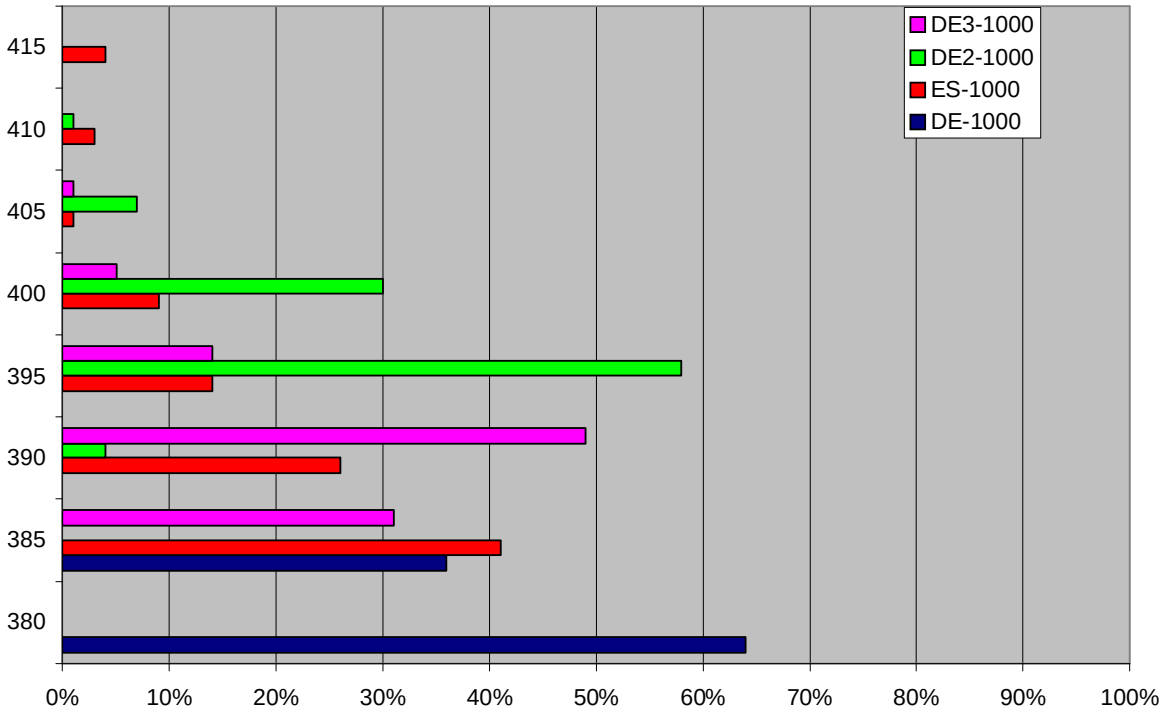
Na Gr. 9.4-2 až Gr. 9.4-4 je vidět rozložení nalezených hodnot při 10, 100 a 1000 iteracích. Z rozložení jasně vyplývá, že pro rychlý „nástřel“ je lepší ES, ale pro doiterování do nejlepšího optima je vhodnější DE. Dále je vidět nevýhodnost algoritmu DE2 (modifikovaná diferenciální evoluce).



Gr. 9.4-2 – procentuální rozložení nalezených hodnot na konstrukci „the 72-bar truss“ při 10 iteracích



Gr. 9.4-3 - procentuální rozložení nalezených hodnot na konstrukci „the 72-bar truss“ při 100 iteracích



Gr. 9.4-4 - procentuální rozložení nalezených hodnot na konstrukci „the 72-bar truss“ při 1000 iteracích

10 Závěr

10.1 Dosavadní testy

Tato práce se snaží ukázat vhodnost jednotlivých algoritmů pro konkrétní použití. Proto se i v závěru omezíme na konstatování, která jsou zřejmá z vypočtených hodnot. Nebudeme tvrdit, že některý algoritmus je lepší než jiný. Na toto téma bylo již napsáno mnoho prací, ale podle nás naprosto neuspokojivých. Jak bylo například zmíněno v kapitole 9.2, někdy jsou i uvedené hodnoty chybné. Proto jsme se snažili o co největší transparentnost v průběhu celé práce i publikovaných výsledků. To je doprovázeno zveřejněním všech použitých zdrojových kódů a vypočtených hodnot. Všechny tyto kroky vedou také k opakovatelnosti všech testů.

10.2 Testy na sadě ANDRE

Na této teoretické sadě byly získány charakteristické rysy konvergence dvou vybraných metod. Jak je vidět z grafů v kap. 8, tak ES většinou velmi rychle najde lokální optimum, v kterém bohužel velmi často zůstane. Naopak DE je velmi robustní metoda, má podstatně vyšší úspěšnost nalezení globálního optima, ale tyto vlastnosti jsou vykoupeny její pomalou konvergencí.

10.3 Testy na reálných konstrukcích

Na všech testovaných prutových konstrukcích vykazovala ES mnohem lepší výsledky při malém počtu iterací. Naopak při dostatečném počtu iterací zdánlivě vychází lépe DE. Jednoznačný závěr ovšem udělat nelze. Když budeme pouštět ES pouze na malém počtu iterací a budeme ji restartovat, tak v mnohých případech dopadne ES lépe než DE při vysokém počtu iterací. Jediné co lze říci, že ES je vhodná pro první nástřely konstrukcí, protože velmi rychle dává uspokojivé výsledky. Jak již bylo zmíněno, většinou pro inženýra není nutné znát optimum (minimum) funkce (konstrukce), ale často stačí lepší řešení než doposud známé.

Je nutné zmínit, že jsme zde prováděli pouze rozměrové optimalizace, ale máme mnoho dalších druhů optimalizací. Na nich nemusí platit zde uvedené závěry.

10.4 Interpretace výsledků

Snahou této práce bylo vytvořit začátek jakési kuchařky pro „běžné“ inženýry, kteří potřebují zoptimalizovat své konstrukce. Výsledky této práce se dají použit při dalším vývoji SW pro optimalizaci stavebních konstrukcí. V optimalizačním nástroji si inženýr vybere kolik času je ochoten obětovat optimalizaci. Aplikace na základě tohoto výběru, typu konstrukce, počtu prvků a dalších parametrů se pokusí vybrat nejvhodnější metodu.

10.5 Doporučení

Je třeba se vyvarovat jakéhokoliv generalizování uvedených výsledků. Všechny výsledky jsou podmíněny uvedenými skutečnostmi. V dalších pracích je třeba ověřit uvedené

skutečnosti na širokém spektru konstrukcí. Vybraná sada funkcí (ANDRE) ani uvedené konstrukce rozhodně nepostihují mnohé aspekty.

10.6 Další kroky

V dalších pracích by bylo zajímavé ověřit, zda by nebyla vhodná kombinace ES a DE tak, že ES by v prvních iteracích napočetla zajímavé sady vstupních parametrů, které by následně tvořili část populace v DE. Bylo by třeba se vypořádat s elitismem (abychom naopak nezhoršili vlastnosti DE), ale současně by to mohlo velmi urychlit konvergenci algoritmu. Je to však jen domněnka, kterou je třeba ověřit.

.

11 Reference

[1] http://www-fp.mcsmanl.gov/otc/Guide/OptWeb

[2] Storn, R and Price, K (1995) "Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces": Technical Report, International Computer Science Institute, Berkley.

[3] Hebák P., Bílková D., Svobodová.: Praktikum k výuce matematické statistiky II: Testování hypotéz, ISBN 80–245–0082-5

[4] Coello C.A. (2001) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Computer methods in Applied Mechanics and Engineering, Volume 191, Number 11, 1245-1287(43)

[5] Hans-Georg Beyer, Bernard Sendhoff, Robust optimization – A comprehensive survey, Comput. Methods Appl. Mech.Engrg. 196(2007) 3190-3218

[6] A.C.C.Lemonge, H.J.C. Barbosa (2004) - An adaptive penalty schneme for genetic algorithm in structural optimization, Int.J. Numer. Metho. Engng. (2004), 59:703-736

[7] Gellatly R.A., Berke L., Optional structural design, Technical Report AFFDL-TR-70-165, Air Force Flight Dynamice Laboratorz (1971)

[8] Schmit A.L., Miura H., A new structural analysis/synthesis capability: access 1. AIAA Journal 1976; 14:661-671

[9] Ghasemi M.R., Hinton E., Wood R.D. Optimization of trusses using genetic algorithms for discrete and continous variables, Engineering Computations 1997; 16:272-301

[10] Schimit L.A., Farsi B., Some approximation concepts in structural synthesis AIAA Journal 1974; 12:692-699

[11] Krishnamoorty CS, Rajeev S., Discrete optimization of structures using genetic algorithms, Journal of Structural Engineering 1992; 118(5):1233-1250

[12] Venkayya VB, Khot NS, Reddy VS, Energy distributoin in an optimal structural desing, Technical Report AFFDL-TR-68-156, Flight Dynamics Laboratory, Wright-Patterson ABF, Ohio, 1969

[13] Dobbs MV, Nelson RB, Aplication of optimality criteria in automated structural design, AIAA Journal 1976; 14:1436-1443

[14] Zhu DM, An improved Templeman's algorith for optimum design of trusses with discrete member sizes, Engineering Optimization 1986; 9:303-312
[15] Erbatur F, Hasancebi O, Tütüncü I, Kilc H, Optimal design of planar and space structures with genetic algorithms, Computers and Structures 2000; 75:209-224.
[16] Wu SJ, Chow PT, Steady-state genetic algorithms for discrete optimization of trusses, Computers and Structures 1995; 56(6):979-991
[17] Krishnamoorty CS, Rajeev S. Discrete optimization of structures using genetic algorithms, Journal of Structural Engineering 1992, 118(5):1233-1250
[18] Galante M, Genetic algorithms as an approach to optimize real-world trusses, International Journal for Numerical Methods in Engineering, 1996; 39:361-382
[19] A.E. Eiben, J. E. Smith (2003). Introduction to Evolutionary Computing. Springer
[20] Venkayya VB. Desing of optimum structures, Journal of Computers and Structures 1971; 265-309
[21] http://en.wikipedia.org/wiki/Building_Information_Modeling
[22] http://en.wikipedia.org/wiki/Meta_heuristic
[23] Robbins, H. and Monro, S., *A Stochastic Approximation Method*, Annals of Mathematical Statistics, vol. 22, pp. 400-407, 1951
[24] Rechenberg, I., Cybernetic Solution Path of an Experimental Problem, Royal Aircraft Establishment Library Translation, 1965
[25] Nakrani S. and Tovey S., On honey bees and dynamic server allocation in Internet hosting centers, Adaptive Behaviour, vol. 12, 223 (2004)
[26] Karaboga D. and Basturk B., On the performance of artificial bee colony algorithm, Applied Soft Computing, vol. 8, 687 (2008)
[27] Hrstka O.,Kučerová A. Improvements of Real Coded Genetic Algorithms Based on Differential Operators Preventing the Premature Konvergence, Advances in Engineering Software, Volume 35, Issues 3-4, Pages 237-246, (2004)
[28] http://cs.wikipedia.org/wiki/T_test
[29] http://fast10.vsb.cz/konecny/files/sskII/cv_3.ppt#287,12,Lokalizace – zkrácený tvar

12 Přílohy

12.1 T-test [28]

Zde si ukážeme jak funguje statistický T-test pro porovnání dvou rozdělení abychom lépe porozuměni předchozím pasážím.

Jestliže označíme jednotlivé hodnoty prvního výběru jako x_{11}, x_{12}, x_{13},…. x_{1n} a hodnoty druhého jako x_{21}, x_{22}, x_{23},…. x_{2n}, pak můžeme psát, že výběrový průměr prvního výběru je roven

$$\overline{X}_1 = \frac{1}{n}\sum_{i=1}^{n} x_{1i}$$

A výběrový rozptyl prvního výběru spočteme jako

$$S_1^2 = \frac{1}{n-1}\sum_{i=1}^{n} (x_{1i} - \overline{X})^2$$

Obdobně spočteme charakteristiky i pro druhý výběr. Dále budeme předpokládat, že

$$\delta = \mu_1 - \mu_2$$

Kde μ_1, μ_2 jsou střední hodnoty výběru. Pak můžeme psát

$$T = \frac{X - Y - \delta}{\sqrt{(n-1)S_1^2 + (n-1)S_2^2}}\sqrt{n(n-1)}$$

V případě, že T přestoupí hladinu pravděpodobnosti, tak říkáme, že výběry na dané hladině nelze rozlišit.

12.2 Sada funkcí „ANDRE“

F1:

$$f(x) = 2(x - 0{,}75)^2 + \sin(5\pi x - 0{,}4\pi) - 0{,}125$$

where $0 \leq x \leq 1$

F3:

$$f(x) = -\sum_{j=1}^{5} [j \sin[(j+1)x + j]]$$

where $-10 \leq x \leq 1$

Branin:

$$f(x, y) = a(y - bx^2 + cx - d)^2 + h(1 - f)\cos x + h$$

where a = 1, b = 5,1/4π^2, c = 5/π, d = 6, h = 10, f = 1/8π,

$-5 \leq x \leq 10, 0 \leq y \leq 15$

Camelback:

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

where $-3 \leq x \leq 3, -2 \leq y \leq 2$

Goldprice

$$f(x,y) = [1 + (x + y + 1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)][30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)]$$

where $-2 \leq x \leq 2, -2 \leq y \leq 2$

PShubert1 and 2:

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x + i]\right\}\left\{\sum_{i=1}^{5} i\cos[(i+1)y + i]\right\} + \beta[(x + 1{,}42513)^2 + (y + 0{,}80032)^2]$$

where $-10 \leq x \leq 10, -10 \leq y \leq 10$

for PShubert1: β = 0,5

for PShubert2: β = 1

Quartic:

$$f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2}$$

where $-10 \leq x \leq 10, -10 \leq y \leq 10$

Shubert:

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x + i]\right\}\left\{\sum_{i=1}^{5} i\cos[(i+1)y + i]\right\}$$

where $-10 \leq x \leq 10, -10 \leq y \leq 10$

Hartman:

$$f(x_1,x_2,x_3) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_i - p_{ij}\right)\right]^2$$

where $0 \leq x_i \leq 1, i = 1,\ldots,3\ x = (x_1,\ldots,x_3)$

$p_i = (p_{i1},\ldots,p_{i3}), a_i = (a_{i1},\ldots,a_{i2})$

i				c_i	p_{ij}		
1	3	10	30	1	0,3689	0,1170	0,2673
2	0,1	10	35	1,2	0,4699	0,4387	0,7470
3	3	10	30	3	0,1091	0,8732	0,5547
4	0,1	10	35	3,2	0,03815	0,5743	0,8828

Shekel1,2 and 3:

$$f(x) = -\sum_{i=1}^{m} \frac{1}{(x - a_i)^T (x - a_i) + c_i}$$

where $0 \leq x_j \leq 10$, for Shekel1: m =5, for Shekel2: m = 7, for Shekel3: m = 10.

$x = (x_1,x_2,x_3,x_4)^T, a_i = (a_{i1},a_{i2},a_{i3},a_{i4})^T$.

i	a_{ij}				c_{ij}
1	4,0	4,0	4,0	4,0	0,1
2	1,0	1,0	1,0	1,0	0,2
3	8,0	8,0	8,0	8,0	0,2
4	6,0	6,0	6,0	6,0	0,4
5	3,0	7,0	3,0	7,0	0,4
6	2,0	9,0	2,0	9,0	0,6
7	5,0	5,0	3,0	3,0	0,6
8	8,0	1,0	8,0	1,0	0,7
9	6,0	2,0	6,0	2,0	0,5
10	7,0	3,6	7,0	3,6	0,5

Hartman2:

$$f(x_1,\dots,x_6) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_i - p_{ij}\right)\right)^2$$

where $0 \le x_j \le 1, j = 1,\dots,6.$

$x = (x_1,\dots,x_6), \quad p_i = (p_{i1},\dots,p_{i6}), \quad a_i = (a_{i1},\dots,a_{i6}).$

i	a_{ij}						c_i
1	10,00	3,00	17,00	3,50	1,70	8,00	1,0
2	0,05	10,00	17,00	0,10	8,00	14,00	1,2
3	3,00	3,50	1,70	10,00	17,00	8,00	3,0
4	17,00	8,00	0,05	10,00	0,01	14,00	3,2

i	p_{ij}					
1	0,1312	0,1696	0,5569	0,0124	0,8283	0,5886
2	0,2329	0,4135	0,8307	0,3736	0,1004	0,9991
3	0,2348	0,1451	0,3522	0,2883	0,3047	0,6650
4	0,4047	0,8828	0,8732	0,5743	0,1091	0,0381

Hosc45:

$$f(x) = 2 - \frac{1}{n!}\prod_{i=1}^{n} x_i$$

where $x = (x_1,\dots,x_n), \quad 0 \le x_i \le i, \quad n = 10.$

Brown1:

$$f(x) = \left[\sum_{i \in J}(x_i - 3)\right]^2 + \sum_{i \in J}\left[10^{-3}(x_i - 3)^2 - (x_i - x_{i+1}) + e^{20(x_i - x_{i+1})}\right]$$

where $J = \{1,3,\dots 19\}, \quad -1 \le x_i \le 4, \quad 1 \le i \le 20, \quad x = (x_1,\dots,x_{20})^T.$

Brown3:

$$f(x) = \sum_{i=1}^{19} \left[(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)} \right]$$

$$x = (x_1, \ldots, x_{20})^T, \quad -1 \leq x_i \leq 4, \quad 1 \leq i \leq 20.$$

F5n:

$$f(x) = (\pi/20)\left\{10sin^2(\pi y_1) + \sum_{i=1}^{19}\left[(y_i - 1)^2\left(1 + 10sin^2(\pi y_i + 1)\right)\right] + (y_{20} - 1)^2\right\}$$

where $x = (x_1, \ldots, x_{20})^T$, $-10 \leq x_i \leq 10$, $y_i = 1 + 0{,}25(x_i - 1)$.

F10n:

$$f(x) = (\pi/20)\left\{10sin^2(\pi x_1) + \sum_{i=1}^{19}\left[(x_i - 1)^2\left(1 + 10sin^2(\pi x_i + 1)\right)\right] + (x_{20} - 1)^2\right\}$$

where $x = (x_1, \ldots, x_{20})^T$, $-10 \leq x_i \leq 10$.

F15n:

$$f(x) = (1/10)\left\{sin^2(3\pi x_1) + \sum_{i=1}^{19}\left[(x_i - 1)^2\left(1 + sin^2(3\pi x_{i+1})\right)\right] + (1/10)(x_{20} - 1)^2\left[1 + sin^2(2\pi x_{20})\right]\right\}$$

where $x = (x_1, \ldots, x_{20})^T$, $-10 \leq x_i \leq 10$.

12.3 Deformační metoda

Pro výpočet konstrukcí byla použita deformační metoda. Proto zde ukážeme alespoň základní myšlenky.

Obr. 12.3-1

Budeme předpokládat prismatický prut. Jeden konec je pevně držen a druhý posuneme ve směru jeho osy o *u,* pak síle k tomu potřebná je rovna

$$F = k * u$$

Kde *k* je tuhost prutu, která je rovna

$$k = \frac{E * A}{l}$$

$$\begin{Bmatrix} F_{1x}^{l} \\ F_{2x}^{l} \end{Bmatrix} = \begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{Bmatrix} u_{1x}^{l} \\ u_{2x}^{l} \end{Bmatrix}$$

Za předpokladu $F'_{1z} = F'_{2z} = 0$ můžeme předcházející rovnici rozšířit

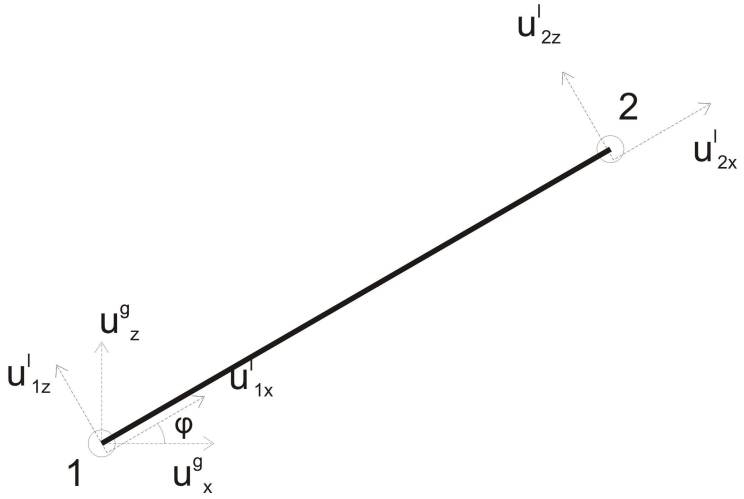
$$\begin{Bmatrix} F_{1x}^{l} \\ F_{1z}^{l} \\ F_{2x}^{l} \\ F_{2z}^{l} \end{Bmatrix} = \begin{bmatrix} k & 0 & -k & 0 \\ 0 & 0 & 0 & 0 \\ -k & 0 & k & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u_{1x}^{l} \\ u_{1z}^{l} \\ u_{2x}^{l} \\ u_{2z}^{l} \end{Bmatrix}$$

Nyní můžeme udělat transformaci vektorů posunutí mezi lokálními a globálními soustavami souřadnic

$$u_{1x}^{l} = u_{1x}^{g}\cos(\varphi) + u_{1y}^{g}\sin(\varphi)$$
$$u_{1z}^{l} = -u_{1z}^{g}\sin(\varphi) + u_{1z}^{g}\cos(\varphi)$$

Pro vektory koncových posunů můžeme psát

$$r^{l} = T * r^{g}$$
$$F^{l} = T * F^{g}$$
$$r^{g} = \left\{u_{1x}^{g}, u_{1z}^{g}, u_{2x}^{g}, u_{2z}^{g}\right\}^{T}$$
$$T = \begin{bmatrix} \cos(\varphi) & \sin(\varphi) & 0 & 0 \\ -\sin(\varphi) & \cos(\varphi) & 0 & 0 \\ 0 & 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix}$$
$$F^{l} = TF^{g}$$
$$T'F^{l} = F^{g} \quad /F^{l} = K^{l}r^{l}$$
$$T'K^{l}r^{l} = F^{g} \quad /r^{l} = Tr^{g}$$
$$T'K^{l}Tr^{g} = F^{g} \quad /T'K^{l}T = K^{g}$$
$$K^{g}r^{g} = F^{g}$$
$$K^{g}r^{g} = F^{g}$$
$$K^{g} = \frac{EA}{l}\begin{bmatrix} \cos(\varphi)^2 & \cos(\varphi)\sin(\varphi) & -\cos(\varphi)^2 & -\cos(\varphi)\sin(\varphi) \\ \cos(\varphi)\sin(\varphi) & \sin(\varphi)^2 & -\cos(\varphi)\sin(\varphi) & -\sin(\varphi)^2 \\ -\cos(\varphi)^2 & -\cos(\varphi)\sin(\varphi) & \cos(\varphi)^2 & \cos(\varphi)\sin(\varphi) \\ -\cos(\varphi)\sin(\varphi) & -\sin(\varphi)^2 & \cos(\varphi)\sin(\varphi) & \sin(\varphi)^2 \end{bmatrix}$$



Obr. 12.3-2

Stejně můžeme odvodit vztah mezi globálními koncovými silami a posunutími ve třech dimenzích. Matice bude pochopitelně mít rozměr 6x6.

Toto jsme provedli v aplikaci MatLab. Ukážeme tedy jen zjednodušeně.

$$T = \begin{bmatrix} xx & xy & xz & 0 & 0 & 0 \\ yx & yy & yz & 0 & 0 & 0 \\ zy & zy & zz & 0 & 0 & 0 \\ 0 & 0 & 0 & xx & xy & xz \\ 0 & 0 & 0 & yx & yy & yz \\ 0 & 0 & 0 & zy & zy & zz \end{bmatrix}$$

Za předpokladu působení pouze normálových sil v prutech (pruty jsou uloženy kloubově) můžeme psát

$$K^l = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \frac{EA}{l}$$

$$K^g = T' K^l T$$

$$K^g = \begin{bmatrix} xx\text{^}2 & xx*xy & xx*xz & -xx\text{^}2 & -xx*xy & -xx*xz \\ xx*xy & xy\text{^}2 & xy*xz & -xx*xy & -xy\text{^}2 & -xy*xz \\ xx*xz & xy*xz & xz\text{^}2 & -xx*xz & -xy*xz & -xz\text{^}2 \\ -xx\text{^}2 & -xx*xy & -xx*xz & xx\text{^}2 & xx*xy & xx*xz \\ -xx*xy & -xy\text{^}2 & -xy*xz & xx*xy & xy\text{^}2 & xy*xz \\ -xx*xz & -xy*xz & -xz\text{^}2 & xx*xz & xy*xz & xz\text{^}2 \end{bmatrix} * \frac{EA}{l}$$

Dále provedeme lokalizaci pomocí kódových čísel. Toto je již standardní proces, který je všeobecně znám a je dobře popsán v literatuře [29].

13 Čestné prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím odborné literatury a pramenů, uvedených v referencích v kapitole 11 a za pomoci konzultací nejen s vedoucím diplomové práce.

14 CD s daty a algoritmy