ELSEVIER

Advances in Engineering Software 36 (2005) 11–19

ADVANCES IN ENGINEERING SOFTWARE

www.elsevier.com/locate/advengsoft

A combined heuristic optimization technique

H. Schmidt[1,*], G. Thierauf

Department of Civil Engineering, Institute of Structural Mechanics, University of Essen, Essen, Germany

Received 1 November 2002; accepted 2 October 2003
Available online 19 August 2004

Abstract

Realistic problems of structural optimization are characterized by non-linearity, non-convexity and by continuous and/or discrete design variables. There are non-linear dependencies between the optimised parameters. Real-world problems are rarely decomposable or separable. In this contribution a combined heuristic algorithm is described which is well suited for problems, for which the application-requirements of gradient-based algorithms are not fulfilled. The present contribution describes a combination of the Threshold Accepting Algorithm with Differential Evolution with particular emphasis on structural optimization, it can be classified as a Hybrid Evolutionary Algorithm. The Threshold Accepting Algorithm is similar to Simulated Annealing. Differential Evolution is based on Genetic Algorithms.

Keywords: Structural optimization; Heuristic algorithms; Non-linearity; Non-convexity; Threshold accepting algorithm; Differential evolution; Hybrid evolutionary algorithms

1. Introduction

Techniques of stochastic search are widely used for structural optimization. An approach for combinatorial optimization is to embed local search into the framework of evolutionary algorithms or the combinations of various techniques.

Burke and Smith [6b] incorporated a local search operator into a genetic algorithm. The resulting algorithm from this hybrid approach has been termed a Memetic Algorithm. The paper investigates the use of a memetic algorithm in solving a thermal generator maintenance scheduling problem. The main purpose is to discover whether a memetic approach can be advanced.

Burke et al. [6a] presented a hybrid population-based metaheuristic algorithm for the space allocation problem in academic institutions. The proposed approach performs an automatic selection of the parameters according to the problem characteristics, the solution quality is evaluated with a penalty function and includes: local search, heuristics, adaptive cooling schedules and population-based techniques.

Magoulas et al. [12] introduced a new hybrid evolutionary approach for improving the performance of neural network classifiers in slowly varying environments. They investigated a combination of Differential Evolution Strategy and Stochastic Gradient Descent. The use of a Differential Evolution Strategy is based on the concept of evolution of a number of individuals from generation to generation, the on-line gradient descent refers to the concept of adaptation to the environment by learning.

Galinier and Hao presented [8] a Hybrid Evolutionary Algorithm for graph coloring. They embedded local search into the framework of population-based Evolutionary Algorithms, leading to Hybrid Evolutionary Algorithms. The basic idea consists of using the crossover-operator (Greedy Partition Crossover) to create new and potentially interesting configurations, which are then improved by the local search operator (Tabu Search).

Botello et al. [1] combined the search-operators selection, crossover, mutation of genetic algorithms with the acceptance operator of the Simulated Annealing and calls this the General Stochastic Search Algorithm. The unmodified individuals of a population (before variation by recombination and mutation) are compared with the varied

* Corresponding author. Tel.: +49-201-183-2654; fax: +49-201-183-2675.
E-mail address: holger_schmidt@uni-essen.de (H. Schmidt).
[1] URL: http://www.statik.uni-essen.de.

0965-9978/$ - see front matter © 2004 Civil-Comp Ltd and Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2003.12.001

ones. The acceptance operator selects solutions to be carried over to the next generation.

Mahfoud and Goldberg [13] presented the Parallel Recombinative Simulated Annealing. After initialization of a population and choice of a system temperature T, parents are chosen selected. The offsprings are produced by recombination and mutation, followed by a comparison between parents and their offsprings. This is carried out, e.g. by a comparison between solutions i and j, where i is the winner with a chance of $1/(1+\exp(E_i - E_j)/T)$. Subsequently, the parents are replaced by the winners and T is reduced. This process is repeated for the complete population in every iteration.

In this paper, the combination with the Threshold Accepting Algorithm (TA), which computes the functionals in every cycle of the iteration, is essential for the increased performance. The Differential Evolution (DE) helps to avoid local optima. By application of penalty functions even inadmissible solutions are allowed, whereby the approximation of the global optimum is possible either from the admissible direction as well as from the inadmissible direction. Admissible results are stored and treated similar to the elite individual in the Genetic Algorithms.

2. The mixed discrete–continuous optimization

A mixed discrete–continuous problem can be expressed as follows:

$$\begin{aligned}&\text{Find :}\\ &X = \{x_1, x_2, x_3, \ldots, x_n\} = [X^{(\mathrm{d})}, X^{(\mathrm{c})}]^{\mathrm{T}}\\ &\text{to minimize}\\ &f(X)\\ &\text{subject to constraints}\\ &\mathrm{g}_j(X) \geq 0 \quad j = 1, \ldots, m\\ &\text{and subject to boundary constraints}\\ &x_i^{(\mathrm{L})} \leq x_i \leq x_i^{(\mathrm{U})} \quad i = 1, \ldots, n\\ &\text{where}\\ &X^{(\mathrm{d})} \in \Re^{\mathrm{d}}, \quad X^{(\mathrm{c})} \in \Re^{\mathrm{c}},\end{aligned} \tag{1}$$

$X^{(\mathrm{d})}$ and $X^{(\mathrm{c})}$ denote feasible subsets of discrete and continuous variables, respectively.

3. The threshold accepting algorithm

The TA was developed by Dueck and Scheuer [7a,b]. TA can be classified as a simplified *simulated annealing algorithm*. The formal process of a TA is:

(1) Initialize first solution $x_j^{(0)}$ and compute $f(x_j)^{(0)}$.
(2) Generate new solution $x_j^{(g+1)}$ by local modification of $x_j^{(g)}$.
(3) Compute positive threshold of tolerance T.
(4) Compute new solution $f(x_j)^{(g+1)}$.
(5) If $f(x_j)^{(g+1)} \leq f(x_j)^{(g)} + f(x_j)^{(g)} T$.
  (a) YES: Replace $x_j^{(g)}$ by $x_j^{(g+1)}$,
  (b) NO: Reduce threshold of tolerance T.
(6) Criterion of termination.
  (a) If satisfactory go to (7),
  (b) If not satisfactory go back to (2).
(7) End of optimization.

Handling of discrete and integer variables is to be explained later. The aim of the optimization is to minimize the objective function $f(x)$

$$\min(f(x)), \tag{2}$$

by optimization of its parameters

$$X = (x_1, \ldots, x_{n_{\mathrm{param}}}) \quad x \in \Re, \tag{3}$$

where X denotes a vector composed of n_{param} objective function parameters. The parameters of the objective function are also subject to lower and upper boundary constraints $x^{(\mathrm{L})}$ and $x^{(\mathrm{U})}$, respectively

$$x_j^{(\mathrm{L})} \leq x_j \leq x_j^{(\mathrm{U})} \quad j \in n_{\mathrm{param}}. \tag{4}$$

The first step for the TA is to create an arbitrary initial solution. For continuous variables it reads

$$x_j^{(0)} = r_j(x_j^{(\mathrm{U})} - x_j^{(\mathrm{L})}) + x_j^{(\mathrm{L})} \quad j = 1, \ldots, n_{\mathrm{param}}, \tag{5}$$

where r denotes to a uniformly distributed random value within the range [0.0,1.0]. A new solution is generated by a local modification of the actual solution

$$x_j^{(g+1)} = x_j^{(g)} + \Delta x_j^{(g)} \quad j \in n_{\mathrm{param}}, \quad g = 1, \ldots, g_{\max}, \tag{6}$$

where

$$\Delta x_j^{(g)} = \begin{cases} \sigma\sqrt{-2\ln(r_1)}\cos(2\pi r_2) & \text{or} \\ \sigma\sqrt{-2\ln(r_1)}\sin(2\pi r_2), \end{cases} \tag{7}$$

and r_1 and r_2 are uniformly distributed random values within the range [0.0,1.0]. The Box–Mueller-method [2] calculates two normaly distributed random values with the deviation σ and the average value $x_j^{(g)}$ from two uniformly distributed random values. The objective function $f(x_j)^{(g+1)}$ is computed. By means of a threshold of tolerance T, even realizations $f(x_j)^{(g+1)}$, which result in a deterioration of the objective function $f(x_j^{g+1})$, are accepted. Searching for a minimum of $f(x)$, a new realization x_j^{g+1} is accepted if $f(x_j^{g+1}) \leq f(x_j^g) + f(x_j^g)T$, where $T > 0$ is the tolerance. The initially high tolerance is continuously reduced (Fig. 2), which corresponds to a reduction of the probability, that deteriorations of the objective function are accepted.

4. The differential evolution

Storn and Price [17] first introduced the DE algorithm a few years ago. DE can be classified as an *evolutionary optimization algorithm*. At present, the best known representatives of this class are *genetic algorithms* by Goldberg [10] and *evolution strategies* by Schwefel [16]. The formal process of a DE is shown in Fig. 1. The DE uses the search-operators mutation, recombination and selection. A detailed compilation of publications in this field can be

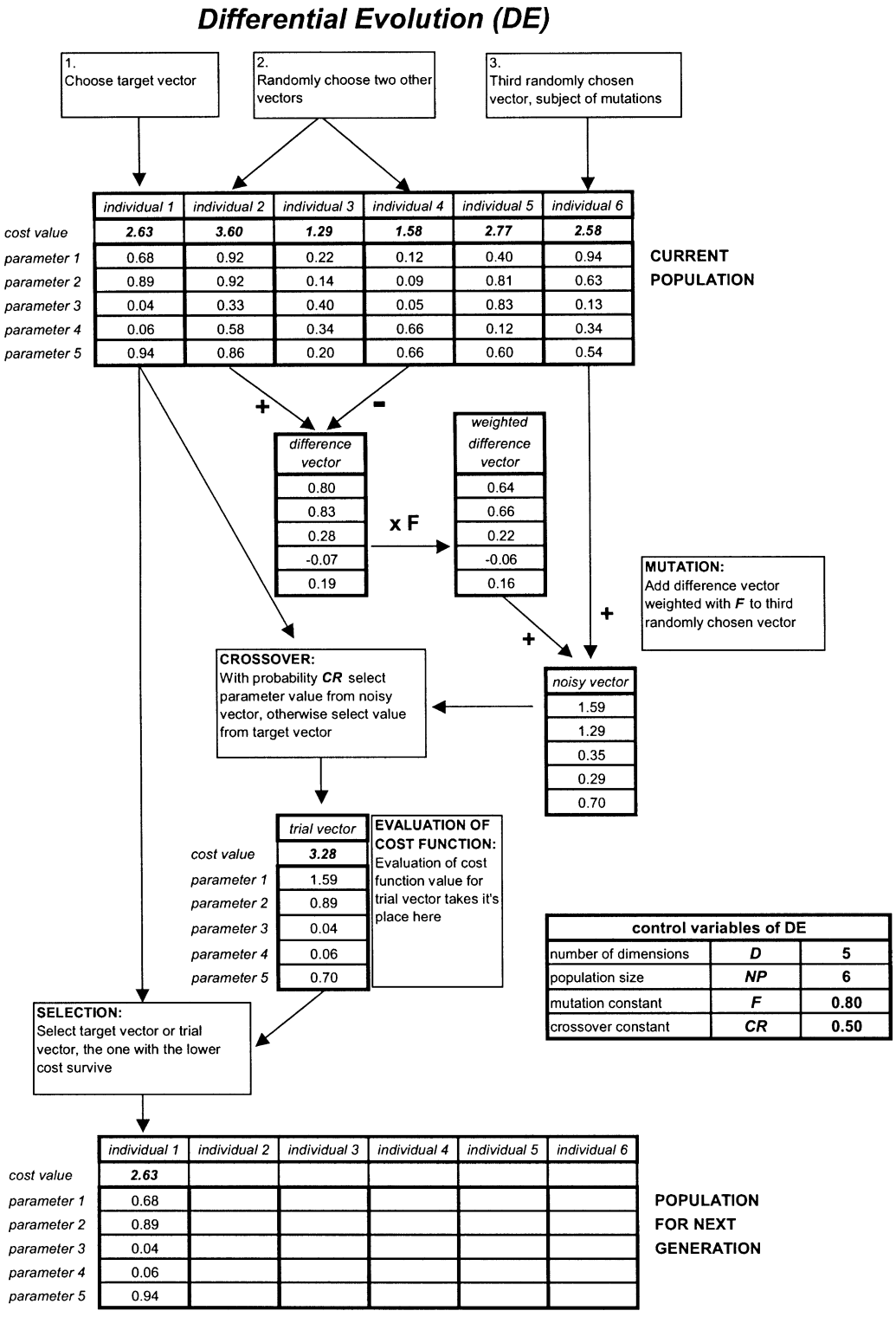

Fig. 1. Differential evolution works directly with the floating-point valued variables of the objective function, not with binary encoding. The functioning of DE is illustrated here for the case of a simple objective function $f(x)=x_1+x_2+x_3+x_4+x_5$ (variables bounded within the range [0.0,1.0]) [11a].

found in Ref. [11b]. The aim of the optimization, the parameters of the objective function and the lower and upper boundary constraints are the same as in Section 3, formulas (2)–(4), respectively. DE operates on a population P of a generation G that contains n_{pop} candidate solutions (individuals). Each vector represents one solution for the optimization problem:

$$P^{(G)} = X_i^{(G)} = x_{i,j}^{(G)} \quad i = 1, \ldots, n_{\text{pop}}, \\ j = 1, \ldots, n_{\text{param}}, \quad G = 1, \ldots, G_{\max}. \tag{8}$$

The first step of the DE is to create an arbitrary initial population $P^{(0)}$

$$P^{(0)} = x_{i,j}^{(0)} = r_{i,j}(x_j^{(U)} - x_j^{(L)}) + x_j^{(\text{L})} \\ i = 1, \ldots, n_{\text{pop}}, \quad j = 1, \ldots, n_{\text{param}}, \tag{9}$$

where r denotes to a uniformly distributed random value within range [0.0,1.0]. From the first generation forward, the population of the following generation $P^{(G+1)}$ ('trial vectors') is created on the basis of the current population $P^{(G)}$. The population of 'trial' vectors $P^{(G+1)}$ is generated as follows (mutation and recombination)

$$x_{i,j}^{(G+1)} = \begin{cases} x_{C_{i,j}}^{(G)} + F(x_{A_{i,j}}^{(G)} - x_{B_{i,j}}^{(G)}) & \text{if } r_{i,j} \le C_r \vee j = D_i \\ x_{i,j}^{(G)} & \text{otherwise,} \end{cases} \tag{10}$$

where

$i = 1,\ldots,n_{\text{pop}}$, $j = 1,\ldots,n_{\text{param}}$,
$D = 1,\ldots,n_{\text{param}}$,
$A = 1,\ldots,n_{\text{pop}}$, $B = 1,\ldots,n_{\text{pop}}$, $C = 1,\ldots,n_{\text{pop}}$,
$A_1 \neq B_i \neq C_i \neq i$,
$C_r \in [0,1]$, $F \in [0,1]$, $r \in [0,1]$.

The population of the next generation $P^{(G+1)}$ is created as follows (selection):

$$X_i^{(G+1)} = \begin{cases} X_i^{(G+1)} & \text{if } f(X_i^{(G+1)}) \le f(X_i^{(G)}) \\ X_i^{(G)} & \text{otherwise.} \end{cases} \tag{11}$$

In this paper, the parameter F is taken according to Zaharie [19]

$$F = \sqrt{1/n_{\text{pop}} - C_r/(2n_{\text{pop}})}, \tag{12}$$

with n_{pop} as the size of population and the parameter C_r which weights the 'differential perturbation' $C_r \in [0,1]$. The values of the parameters which satisfy $2F^2 - 2/n_{\text{pop}} + C_r/n_{\text{pop}} = 0$ can be considered critical [19].

5. Combination of algorithms

The combined Threshold Accepting-Differential Evolution Algorithm (TADE) can be explained as follows:

(1) Initialize original population $P^{(0)}$.
(2) Execute TA for each individual of $P^{(G)}$.
(3) Store the solution after $g_{\max}$ TA-iterations into $P^{(G)}$.
(4) Number of runs of TA = size of population?
  (a) YES: Execute DE and build $P^{(G+1)}$ and store the solution into $P^{(G)}$,
  (b) NO: go back to (2).
(5) Criterion of convergence.
  (a) If satisfactory go to (6),
  (b) If not satisfactory go back to (2).
(6) End of optimization.

TADE starts with an arbitrary initial population $P^{(0)}$; the TA is started sequentially for every individual. The generated solution at the end of TA is stored in population $P^{(G)}$. After completing population $P^{(\text{G})}$, the DE is started and $P^{(G+1)}$ is created. The result is stored in $P^{(G)}$. The worst individual is replaced by the global best admissible result. This procedure is repeated until the termination criteria are fulfilled. This quasi-elite strategy insures the presence of admissible and good results at the end of the optimization. The size of the population and the number of TA-iterations are depending on the specific problem. The threshold T is defined according to Eq. (13)

$$T = a^{-1} \exp\left(-\sin\left(\frac{g+G}{g_{\max}+G_{\max}}\right)\right), \tag{13}$$

where

$a = gG$: for continuous or discrete problems.
$a = 1 + 2g$: for mixed continuous–discrete problems.
g: counter of TA-runs.
$g_{\max}$: maximum number of TA-runs per process.
G: counter of DE-runs.
$G_{\max}$: maximum number of DE-runs per optimization circle.

After each DE cycle the starting amplitude of T, which is operating currently the TA only, is reduced. Fig. 2 shows the reduction of the threshold for $g_{\max} = 40$, $G_{\max} = 40$ and $a = 1 + 2g$ as an example. The convergence velocity is depending on the reduction of T. Similarly the standard deviation σ (decrease of increments) is reduced in formula (7) ($a = 1 + 2g$ for continuous variables, $a = 2$ for discrete variables).

6. Scaling of variables and constraints

The variables and the constraints are scaled for the operators of TADE, e.g. variation of variables, calculation

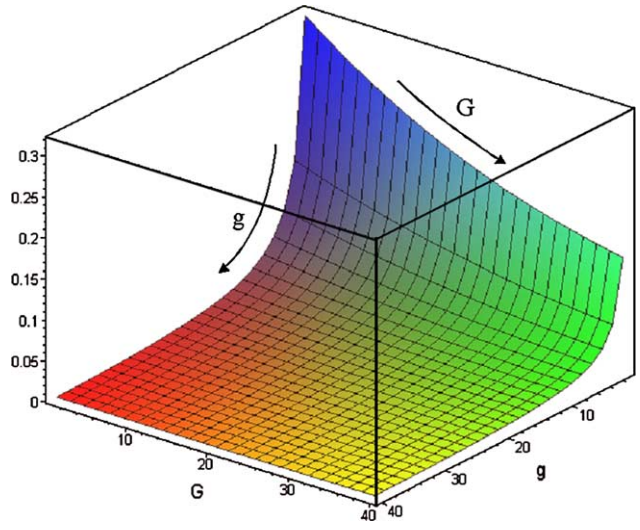

Fig. 2. Threshold of tolerance T.

of penalty-terms. The original floating-point values or discrete values are necessary in order to evaluate the objective function

$$x_{j,s} = \frac{x_j - x_j^{(\mathrm{L})}}{x_j^{(\mathrm{U})} - x_j^{(\mathrm{L})}} \quad j = 1, \ldots, n_{\mathrm{param}}$$
$$g_{j,s}(X) = 1 - \frac{g_j(X) - g_j(X)^{(\mathrm{L})}}{g_j(X)^{(\mathrm{U})} - g_j(X)^{(\mathrm{L})}} \qquad j = 1, \ldots, m, \tag{14}$$

where $x_{j,s}$ and $g_{j,s}(X)$ have the range [0.0,1.0].

7. Constraint handling

7.1. Boundary constraints

In boundary constrained problems it is essential to ensure that parameter values do not exceed their allowed ranges after reproduction. In this work this is guaranteed by replacing parameter values, which violate boundaries with boundary values generated within the feasible range

$$x_{i,j}^{(G+1)} = \begin{cases} x_{i,j}^{(G+1)} = x_j^{(\mathrm{L})} & \text{if } x_{i,j}^{(G+1)} < x_j^{(\mathrm{L})}, \\ x_{i,j}^{(G+1)} = x_j^{(\mathrm{U})} & \text{if } x_{i,j}^{(G+1)} > x_j^{(\mathrm{U})}, \\ x_{i,j}^{(G+1)} & \text{otherwise}, \end{cases} \tag{15}$$

where

$$i = 1, \ldots, n_{\mathrm{pop}}, \quad j = 1, \ldots, n_{\mathrm{param}}.$$

7.2. Constraint functions

In this investigation an adaptive penalty function was applied for handling of the constraints according to Coit and Smith [5a,b]. A Survey of Constraint Handling Techniques is found in Ref. [4]. An individual is evaluated by

$$f(X)^{(gp)} = f(X)^{(g)} + (f(X)_{\mathrm{feas}} - f(X)_{\mathrm{all}}) \sum_{j=1}^{m} \left(\frac{g_j(X)}{\mathrm{NFT}(g)} \right)^k. \tag{16}$$

The adaptive term $(f(X)_{\mathrm{feas}} - f(X)_{\mathrm{all}})$ in Eq. (16) calculates the difference between the non-penalized solution value of the best solution yet found (which will probably be infeasible) and the value of the best feasible solution yet found. $f(X)^{(gp)}$ is the penalized objective function value of solution g, $f(X)^{(g)}$ is the unpenalized objective function value for the solution g, $f(X)_{\mathrm{all}}$ denotes the unpenalized value of the best solution yet found and $f(X)_{\mathrm{feas}}$ denotes the value of the best feasible solution yet found. The exponent k is a constant, which adjusts the 'severity' of the penalty (a value of $k=2$ has been previously suggested in Ref. [5a] and was taken in this paper). The NTF(g) is the so-called *Near Feasibility Threshold*, which is defined as the threshold distance from the feasible region at which the user would consider that the search is 'reasonably' close to the feasible region. The NTF(g) is defined as follows

$$\mathrm{NFT}(g) = \frac{\mathrm{NFT}_0}{\lambda G} \quad G = 1, \ldots, G_{\max}, \tag{17}$$

where NFT_0 is the upper boundary for NFT. In this paper $\lambda G = 1 + (G/n_{\mathrm{pop}})^2$ and $\mathrm{NFT}_0 = 1$. The factor $f(X)_{\mathrm{feas}} - f(X)_{\mathrm{all}}$ has some critical potential: if $f(X)_{\mathrm{feas}}$ is much higher than $f(X)_{\mathrm{all}}$, the penalty would be quite large for all individuals of the population. In this work, $f(X)_{\mathrm{feas}}$ is calculated by the initial vector and $f(X)_{\mathrm{all}} = f(X)_{\mathrm{feas}}/1.5$ after each circle of DE. Of course, one needs a feasible starting point of optimization. On the other hand, if those are identical, the penalty would be zero and all infeasible solutions would pass unpenalized into the next generation.

8. Termination criterion

After each circle of DE the average standard deviation of the individuals/population is calculated. The standard deviation for one variable is

$$S_{\mathrm{d}}(x_{i,j})^{(G+n)} = \sqrt{1/(n_{\mathrm{pop}} - 1) \left(\sum_{i=1}^{n_{\mathrm{pop}}} (x_j - \bar{x})^2 \right)}, \tag{18}$$

where $i = 1, \ldots, n_{\mathrm{pop}}$, $j = 1, \ldots, n_{\mathrm{param}}$, $n = 1, \ldots, G_{\max} - 1$.

The average standard deviation for the population is:

$$\langle S_{\mathrm{d}}(x_{i,j})^{(G+n)} \rangle = \sum_{j=1}^{n_{\mathrm{param}}} (S_{\mathrm{d}}(x_{i,j})^{(G+n)}/n_{\mathrm{param}}). \tag{19}$$

Based on the average standard deviation the following termination criterion is used:

$$\langle S_{\mathrm{d}}(x_{i,j})^{(G+n)} \rangle \leq \varepsilon. \tag{20}$$

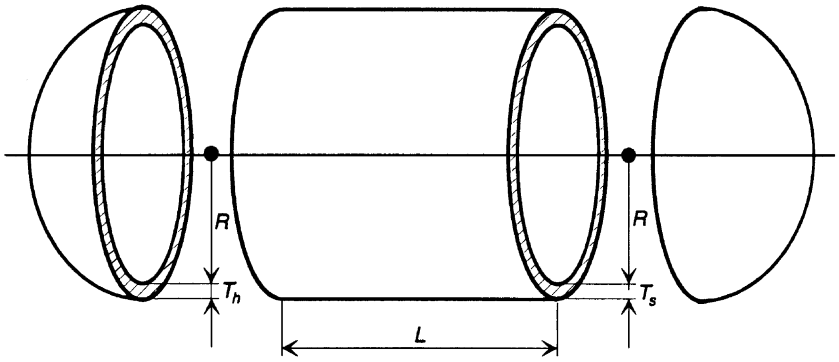

Fig. 3. Example 10.1—pressure vessel [11a].

9. Handling of integer and discrete variables

The TA and DE are only capable of handling continuous variables in the presented form. To extend it for optimization of discrete variables, a couple of modifications are required. Both DE and TA may still work internally with continuous floating-point values. For the evaluation of the cost-function the discrete values will be used. Thus a function for converting a real value to an integer (discrete) value is used. That means, e.g. a continuous value 1.56 is rounded to the integer value 2. Discrete values can be handled in a straightforward manner. Suppose that the subset of discrete variables $X^{(d)}$ contains l elements that can be assigned to variable x:

$$x^{(d)} = x_i^{(d)} \quad i = 1, \ldots, l \quad \text{where} \quad x_i^{(d)} < x_{i+1}^{(d)}. \tag{21}$$

Instead of the discrete value x_i itself, we may assign its index i to x. The discrete variable can be handled as an integer variable, which is boundary constrained to the range 1,…,l. To evaluate the objective function, the discrete value x_i is used instead of its index i.

10. Test examples

10.1. Designing a pressure vessel

The first example is the design of a compressed air storage tank, Fig. 3. Variables L and R are both continuous while T_s and T_h are both discrete. The thickness of the shell T_s and the head T_h, are both to be taken from a set of standard sizes. The control parameters are: $n_{pop}=5$, F (see formula (12)), $C_r=0.5$, $g=40$, $G=40$, $\varepsilon=10^{-4}$, function calls = 10,000 or termination criterion (see Section 8). For this example, the steel plate was available in different thicknesses, which were multiples of 0.0625 in. The problem can be formulated as follows:

Find

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

to minimize

$$f(X) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1611x_1^2x_4 + 19.84x_1^2x_3$$

subject to

$$g_1(X) = -0.0193x_3 + x_1 \geq 0,$$
$$g_2(X) = -0.00954x_3 + x_2 \geq 0,$$
$$g_3(X) = -750.0 \times 1728.0 + \pi x_3^2 x_4 + \tfrac{4}{3}\pi x_3^3 \geq 0,$$
$$g_4(X) = -x_4 + 240.0 \geq 0,$$
$$g_5(X) : 1.0 \leq x_1 \leq 12.5,$$
$$g_6(X) : 0.6 \leq x_2 \leq 12.5,$$
$$g_7(X) : 0.0 \leq x_3 \leq 240.0 : \text{non} - \text{negative value of } x_3,$$
$$g_8(X) : 0.0 \leq x_4 \leq 240.0 : \text{non} - \text{negative value of } x_4. \tag{22}$$

The results were compared with Refs. [11a,18] in Table 1. The average results exceeding 100 runs are: $f(X)^{av} = 7011.66$. The optimum was first encountered at $c_{1th}^{av} = 7580.07$ and the total function calls until the termination criterion stopped the search at $c_{tot}^{av} = 9908.42$. These results

Table 1
Comparison of result-test example 10.1

	Solution in Ref. [11a]	Solution in Ref. [18]	Solution by TADE
$f(X)$	7006.358	7006.9	7006.51
x_1	1.000	1.000	1.000
x_2	0.625	0.625	0.625
x_3	51.81347	51.812	51.8131
x_4	84.57853	84.591	84.5851
g_1	0.000	0.000	0.000
g_2	0.131	0.131	0.131
g_3	−0.064	15.000	18.581
g_4	155.421	155.409	155.406
Fct. calls	10,000	4800	10,000/stop

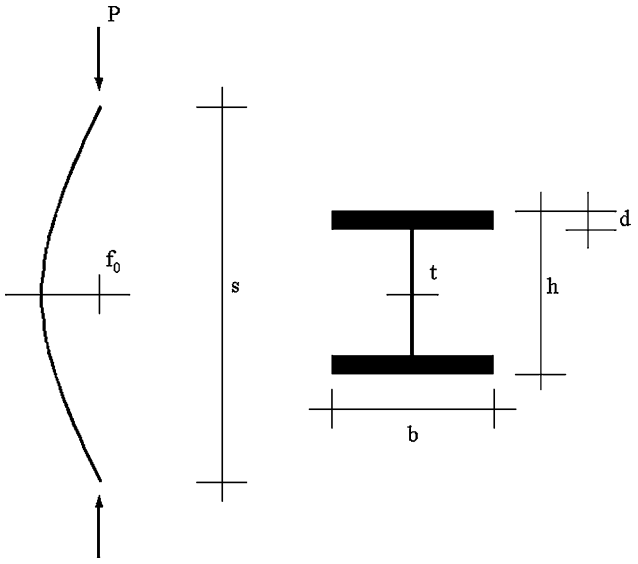

Fig. 4. Example 10.2—steel column [14].

are in good accordance with Refs. [11a,18], where for the latter the function $g_3(X)$ is slightly violated.

10.2. Steel column

As the second example the safety coefficient of a slender steel column with initial excentricity under compression force taken from Ref. [14] was analysed, Fig. 4. The same control parameters as in example 10.1 were used. The problem has 10 uncertain variables with different distribution functions, which are assumed to be stochastically independent. The convergence of the safety coefficients β can be considered as an optimization problem. The objective function and the constraints are defined as

$$f(X) = \sqrt{\sum_{i=1}^{n_{\text{param}}=10} (v_i^2)} \quad i = 1, \ldots, n_{\text{param}}, \tag{23}$$

and

$$g_1(X) : -36.9 \le 36.9,$$

$$g_2(X) = -f_s + P\left(\frac{1}{A} + \frac{F_0}{W}\frac{e}{e-P}\right) \ge 0,$$

where

$$P = P_1 + P_2 + P_3 \quad \text{total load},$$

$$A = 2\text{bd} + \text{th} \quad \text{area},$$

$$W = \text{bdh} + \frac{th^2}{6} \quad \text{section modulus},$$

$$I = \frac{1}{2}\text{bd}h^2 + \frac{th^3}{12} \quad \text{moment of inertia},$$

$$e = \frac{\pi^2 \text{EI}}{s^2} \quad \text{Eulerian buckling load}. \tag{24}$$

The average results exceeding 100 runs are: $f(X)^{\text{av}} = 3.652$. The optimum was first encountered at $c_{1\text{th}}^{\text{av}} = 6233.07$ and the total function calls until the termination criterion stopped the search $c_{1\text{th}}^{\text{av}} = 6731.66$. These results are in good accordance with Ref. [14] ($f(X) = 3.626$), where different optimization algorithms were tested: Adaptive Sampling: 5400 functions calls, Simulated Annealing: 8600 functions calls and Genetic-Evolutionary Search Techniques: 15,000 functions calls. The described algorithm needed approximately the same number of function calls than the fastest heuristic search algorithm tested and it was considerably faster than the Simulated Annealing algorithm tested therein (Table 2).

Table 2
Variables-test example 10.2

Var.	Eval.
Yield stress	$f_s = 25.0v_1 + 500.0$
Load 1	$P_1 = 1.5\times10^5 v_2 + 8.0\times10^5$
Load 2	$P_2 = 1.5\times10^5 v_3 + 8.0\times10^5$
Load 3	$p_3 = \exp\left(\sqrt{\log\left(1+\frac{1}{16}\right)}v_4 + \log(8.0\times10^5) - \log\left(1+\frac{1}{16}\right)/2\right)$
Web thickness	$t = 0.5v_5 + 10.0$
Flange width	$b = \exp(\sqrt{\log(1+10^{-4})}v_6 + \log(300.0) - \log(1+10^{-4})/2)$
Flange thickness	$d = \exp\left(\sqrt{\log\left(1+\frac{1}{400}\right)}v_7 + \log(20) - \log\left(1+\frac{1}{400}\right)/2\right)$
Profile height	$h = \exp\left(\sqrt{\log\left(1+\frac{1}{3600}\right)}v_8 + \log(300.0) - \log\left(1+\frac{1}{3600}\right)/2\right)$
Initial eccentricity	$F_0 = 21.09v_9 + 25.98$
Young's modulus	$E = 4200.0v_{10} + 2.1\times10^5$
Length	$s = 9500$

10.3. Ten-bar truss

The third example is the ten-bar truss. The same control parameters as in example 10.1 were used. For comparability imperial units were used. The objective function is:

$$\min f(X) = \sum_{i=1}^{n_{\text{param}}=10} (l_i A_i \rho). \tag{25}$$

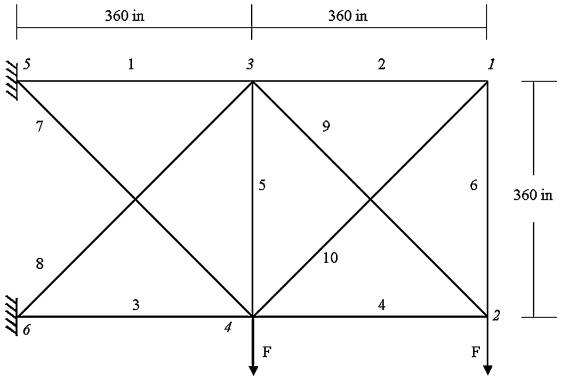

Fig. 5. Example 10.3—ten-bar Truss [9].

Table 3
Comparison of result-test example 10.3

	Solution in Ref. [15]	Solution in Ref. [9]	Solution in Ref. [3]	Solution by TADE
$f(X)$	5613.8	5458.3	5490.75	5490.75
$A1$	33.5	33.5	33.5	33.5
$A2$	1.62	1.62	1.62	1.62
$A3$	22.0	22.0	22.9	22.9
$A4$	15.5	14.2	14.2	14.2
$A5$	1.62	1.62	1.62	1.62
$A6$	1.62	1.62	1.62	1.62
$A7$	14.2	7.97	7.97	7.97
$A8$	19.9	22.9	22.9	22.9
$A9$	19.9	22.0	22.0	22.0
$A10$	2.62	1.62	1.62	1.62
Const.	2.0007	2.0123	1.9989	1.9989

In Fig. 5, the truss topology and the two vertically downward loads of $F = 100.0$ Kips at joints 2 and 4 are applied. A maximum displacement of 2.00 in. is allowed in these joints. The assumed data are

$$\begin{aligned} \rho &= 0.1 \text{ lb/in}^3 && \text{specific weight,} \\ E &= 10^4 \text{ Ksi} && \text{modulus of elasticity,} \\ \sigma_{\max} &= \pm 25 \text{ Ksi} && \text{maximum stress.} \end{aligned} \tag{26}$$

The list of discrete values is taken from the American Institute of Steel Construction (1989), double angle profiles; $s = (1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5)$ (in.^{2}).

The results were compared with Refs. [3,9,15] in Table 3. The average results exceeding 100 runs are: $f(X)^{\text{av}} = 5510.65$ lb. The optimum was first encountered at $c_{1\text{th}}^{\text{av}} = 4985.43$ and the total function calls until the termination criterion stopped the search at $c_{\text{tot}}^{\text{av}} = 5989.6$. These results are in good accordance with Refs. [3,9], where for the latter the displacement boundary is lightly violated.

11. Conclusion

Besides the examples presented in this contribution the algorithm was tested on several other convex and non-convex problems known from optimization literature and problems arising from structural optimization. As far as the problem was constrained the algorithm turned out to be quite effective. It was robust in case the strategy parameters were varied, but the effectiveness and the probability of finding the changed optimum were only moderate.

The combined TADE is a heuristic optimization technique, which is well suited for optimization problems, which are characterized by non-linearity, non-convexity and by continuous and/or discrete design variables. The proposed combination and modification of two known heuristic algorithms has shown good convergence velocity and reliability, and seems to be particularly well suited for realistic problems in structural engineering. These conclusions are based on a limited number of test problems.

The analysis of realistic structures is usually based on finite element computations. Therefore, the fairly simple structure of the algorithms is an advantage for their implementation into existing finite element codes.

References

[1] Botello S, Marraquin JL, Oñate E, Van Horebeek J. Solving structural optimization problems with genetic algorithms and simulated annealing. Int J Numer Methods Eng 1999;45:1069–84.

[2] Box GEP, Mueller ME. A note on the generation of random normal deviates. Ann Math Stat 1958;29:610–1.

[3] Cai J. Diskrete Optimierung dynamisch belasteter Tragwerke mit sequentiellen und parallelen Evolutionsstrategien, Dissertation an der Universitt-Gesamthochschule-Essen, Essen; 1995.

[4] Coello CAC. A survey of constraint handling techniques used with evolutionary algorithms 1999. Technical Report Lania-RI-99-04, Laboratorio Nacional de Informtica Avanzada, 1999, a.v.I.: http://www.cs.cinvestav.mx/constraint.

[5] Coit DW, Smith AE. Penalty guided genetic search for reliability design optimization. Comput Ind Eng 1996;30(4):895–904. Special Issue on Genetic Algorithms. a.v.I.: http://www.cs.cinvestav.mx/constraint. Coit DW, Smith AE, Tate DM. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. IN-FORMS J Comput 1996;8(2):173–82.

[6] Burke EK, Cowling P, Landa Silva EK. Hybrid population-based metaheuristic approaches for the space allocation problem. Proceedings of the 2001 Congress on Evolutionary Computation CEC2001 a.v.I 2001; http//citeseer.nj.nec.com/burke01hybrid.html. Burke EK, Smith AJ. Hybrid evolutionary techniques for the maintenance scheduling problem. IEEE Trans Power Syst 2000;15(1):122–8. ISSN 0885-8950. a.v.I.: http://www.asap.cs.nott.ac.uk/publications/pdf/ajs99_ieee.pdf.

[7] Dueck G, Scheuer T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. J Comput Phys 1990;90:161–75. Dueck G, Scheuer T, Wallmeier H-M. Toleranzschwelle und Sintflut: neue Ideen zur Optimierung. Spektrum der Wissenschaft 1993;3:42–51.

[8] Galinier P, Hao JK. Hybrid evolutionary algorithms for graph coloring. J Comb Optim 1999;3(4):379–97 a.v.I.: http://www.info.univ-angers.fr/pub/hao/papers/JOCO99.pdf.

[9] Galante M. Genetic algorithms as an approach to optimize real-world trusses. Int J Numer Methods Eng 1996;39:361–82.

[10] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley; 1989.

[11] Lampinen J, Zelinka I. Mechanical engineering design optimization by differential evolution. In: David C, Marco D, Fred G, editors. New ideas in optimization. London: McGraw-Hill; 1999, p. 127–46. ISBN 007-709506-5. Lampinen J. A bibliography of differential evolution algorithm. Technical Report. Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing a.v.I.: http://www.lut.fi/jlampine/debiblio.htm.

[12] Magoulas GD, Plagianakos VP, Vrahatis MN. Hybrid methods using evolutionary algorithms for on-line training. In: Proceedings of the INNS-IEEE International Joint Conference on Neural Networks,

Washington DC; 14–19 July 2001, USA. a.v.I.: http://www.info.univ-angers.fr/pub/hao/papers/JOCO99.pdf.

[13] Mahfoud WS, Goldberg DE. Parallel recombinative simulated annealing: a genetic algorithm IlliGAL Report No. 93006, Department of Computer Science, University of Illinois 1994.

[14] Rackwitz R. Comparison of gradient-free optimizers in structural reliability analysis. In: Marti K, editor. Stochastic optimization techniques: numerical methods and technical applications. Berlin: Springer; 2002, p. 309–19.

[15] Rajeev S, Krishnamoorthy CS. Discrete optimization of structures using genetic algorithm. J Struct Eng 1992;118:1233–50.

[16] Schwefel H-P. Evolution and optimum seeking. New York: Wiley; 1995.

[17] Storn R, Price K. Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces 1995. Technical Report TR-95-012, ICSI, March 1995, a.v.I.: http://www.icsi.berkeley.edu/techreports/1995.abstracts/tr-95-012.html.

[18] Thierauf G, Cai J. Parallel evolution strategy for solving structural optimization. Eng Struct 1997;19(4):318–24.

[19] Zaharie D. Critical values for the control parameters of differential evolution algorithms Proceedings of MENDEL 2002, 8th International Conference on Soft Computing 2002.