University of Ostrava

Institute for Research and Applications of Fuzzy Modeling

Adaptive population-based search: application to estimation of nonlinear regression parameters

Josef Tvrdík, Ivan Křivý and Ladislav Mišík

Research report No. 98

2006

Submitted/to appear:

Comput. Stat. & Data Analysis

Supported by:

Grant 201/05/0284 of the Czech Grant Agency and the research scheme MSM 6198898701

University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
30. dubna 22, 701 03 Ostrava 1, Czech Republic

tel.: +420-59-6160234 fax: +420-59-6120 478
e-mail: tvrdik@osu.cz

Adaptive population-based search: application to estimation of nonlinear regression parameters

Josef Tvrdík [a,*], Ivan Křivý [a] and Ladislav Mišík [b]

[a] *Department of Computer Science*
University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic

[b] *Department of Mathematics*
University of Ostrava, 30. dubna 22, 701 03 Ostrava, Czech Republic

Abstract

This paper deals with algorithms for the estimation of nonlinear regression parameters. Adaptive population-based search algorithms were proposed and implemented for finding reliable estimates at a reasonable time consumption with default setting of their tuning parameters. The algorithms were tested on the NIST collection of datasets containing 27 nonlinear regression tasks of various level of difficulty. The experimental results proved that both our algorithms with competing heuristics (CRS4HC and CRS4HCe) are significantly more reliable as compared with the algorithm based on Levenberg-Marquardt optimizing procedure.

Key words: Global optimization, evolutionary algorithms, controlled random search, convergence, heuristics, nonlinear regression

1 Introduction

In an additive nonlinear regression model, the elements of random vector $\mathbf{Y}$ are expressed as follows

$$Y_i = g(\mathbf{x}_i, \boldsymbol{\beta}) + \varepsilon_i, \quad i = 1, 2, \ldots, n \tag{1}$$

* Corresponding author, phone +420 596 160 231, fax +420 596 241 082
Email address: tvrdik@osu.cz (Josef Tvrdík).
URL: albert.osu.cz/tvrdik (Josef Tvrdík).

where $\mathbf{x}_i^T = (x_1, x_2, \ldots, x_k)$ is i-th row of regressor matrix $\mathbf{X}$, $\boldsymbol{\beta}$ is vector of parameters, g is a given function nonlinear in parameters, and ε_i's are iid random variables with zero means. The estimation of parameters by the least squares method means to find such estimates of $\boldsymbol{\beta}$ that minimize the residual sum of squares $Q(\boldsymbol{\beta})$ given by the following equation

$$Q(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[Y_i - g(\mathbf{x}_i, \boldsymbol{\beta})\right]^2 \ . \tag{2}$$

Because of the function $Q(\boldsymbol{\beta})$ need not be unimodal, the estimation of $\boldsymbol{\beta}$ is the global optimization problem. Iterative deterministic algorithms (Levenberg-Marquardt or Gauss-Newton) used in standard statistical packages often fail when searching for the true solution of the problem. Several statistical packages were tested (Tvrdík and Křivý, 2004) on higher-level-difficulty tasks collected by NIST (2001). For approximately one half of the tasks, the algorithms either completely failed or resulted in a significant disagreement with the certified parameter values.

This paper presents an attempt to propose an adaptive population-based algorithm giving reliable estimates at reasonable time consumption with default setting of its control parameters. The paper is an extended version of the contribution submitted to Compstat (Tvrdík et al., 2006).

2 Global optimization

Let us consider a continuous global optimization problem with box constraints, i.e. for a given objective function $f : D \to \mathbb{R}$, $D \subset \mathbb{R}^d$, the point $\mathbf{x}^*$ is to be found such that $\mathbf{x}^* = \arg\min_{\mathbf{x}\in D} f(\mathbf{x})$. The point $\mathbf{x}^*$ is called the global minimum, D is the search space defined as $D = \prod_{i=1}^{d}[a_i, b_i]$, $a_i < b_i$, $i = 1, 2, \ldots, d$, and the objective function $f(\mathbf{x})$ is supposed to be computable.

The global optimization problem mentioned above is hard to solve. A heuristic attempt to solving the problem like stochastic search (Spall, 2003) and evolutionary algorithms (e.g. Bäck, 1996) is playing ever-growing role in most practical tasks of the global optimization.

Both the stochastic search and the evolutionary algorithms adapt the current search strategy that was gathered from the search up to the present time. The adaptation in the evolutionary algorithms was mainly focused to a population of individuals (points in D). Attempts that involve self-adaptive mechanisms into the core of algorithms were presented in recent years. These mechanisms are based on the competition and cooperation of heuristic evolutionary operators – see Deb (2005); Winter et al. (2005); Tvrdík et al. (2001, 2002). In

spite of the No Free Lunch theorems indicating the same performance of all the search algorithms over the set of possible optimization problems (Wolpert and Macready, 1997), empirical comparisons imply that the self-adaptive algorithms outperform the others in many optimization problems.

3 Controlled random search with competing heuristics

Controlled random search (CRS) is a simple stochastic algorithm searching for the global minimum in the problem defined above. The algorithm was proposed by Price (1977), where the reflection known from simplex method (Nelder and Mead, 1964) was used for generating a new trial point. There are several modifications of the CRS algorithm which were successfully used in solving the global optimization problems (Ali and Törn, 2004). The CRS written in pseudo-code is described in Algorithm 1.

Algorithm 1. Generalized controlled random search

```
1  generate P (population of N points in D at random);
2  find x_max (the point in P with the highest function value);
3  repeat
4    generate a new trial point y ∈ D by using a heuristic;
5    if f(y) < f(x_max) then
6      x_max := y;
7      find new x_max;
8    endif
9  until stopping condition;
```

The role of a heuristic mentioned at line 4 can play any non-deterministic rule generating a new trial point $\mathbf{y} \in D$. There are many different heuristics that can be used and, moreover, the heuristics can alternate during the course of search.

Suppose we have h heuristics that can be chosen at each iteration step completely at random with the probability $q_i,\ i = 1, 2, \ldots, h$. The probabilities q_i can change according to the successfulness of heuristics in preceding steps of searching process. The heuristic is successful in the current step, if it generates such a trial point $\mathbf{y}$ that $f(\mathbf{y}) < f_{\max}$, $f_{\max} = f(\mathbf{x}_{\max})$. The probability q_i can be simply evaluated by taking into account the corresponding frequency of successes. Other way consists in the weighting of successes by using the formula

$$w_i = \frac{f_{\max} - \max(f(\mathbf{y}), f_{\min})}{f_{\max} - f_{\min}}\,. \tag{3}$$

Thus $w_i \in (0,\ 1]$ and the corresponding probability q_i is evaluated as

$$q_i = \frac{W_i + w_0}{\sum_{j=1}^{h}(W_j + w_0)} \tag{4}$$

where W_i is the sum of w_i in previous searching steps and $w_0 > 0$ is an input parameter of the algorithm. In order to avoid the degeneration of evolutionary process the current values of q_i are reset to their starting values ($q_i = 1/h$) when any probability q_i decreases below a given limit $\delta > 0$. The CRS algorithm with competing heuristics belongs to the class of evolutionary algorithms described in Tvrdík et al. (2002).

Several sets of heuristics were experimentally checked on test functions (Tvrdík, 2004, 2005) and non-linear regression tasks (Tvrdík and Křivý, 2004). Four competing heuristics selected with respect to these experimental results were used in the implementation of the CRS algorithm for nonlinear regression parameter estimate. Three of them are based on a randomized reflection in the simplex S ($d+1$ points chosen from P) proposed by Křivý and Tvrdík (1995). A new trial point $\mathbf{y}$ is generated from the simplex by the relation

$$\mathbf{y} = \mathbf{g} + U(\mathbf{g} - \mathbf{x}_H) \tag{5}$$

where $\mathbf{x}_H = \arg\max_{\mathbf{x}\in S} f(\mathbf{x})$ and $\mathbf{g}$ is the centroid of remaining d points of the simplex S. The multiplication factor U is a random variable distributed uniformly in $[s, \alpha - s)$, $\alpha > 0$ and s being input parameters, $0 < s < \alpha/2$. All the $d + 1$ points of simplex are chosen at random from P in two heuristics: the heuristic denoted `REFL1` uses $\alpha = 2$, and $s = 0.5$, the heuristic denoted `REFL25` uses $\alpha = 5$, and $s = 1.5$. Regarding the third heuristic denoted `REFLB`, one point of the simplex is the point of P with the minimum objective function value and the remaining d points of the simplex S are chosen at random from remaining points of P. Input parameters of `REFLB` are set to $\alpha = 2$ and $s = 0.5$.

The fourth competing heuristic is based on differential evolution (Storn and Price, 1997). A point $\mathbf{u}$ is generated according to

$$\mathbf{u} = \mathbf{r}_1 + F(\mathbf{r}_2 - \mathbf{r}_3) \tag{6}$$

where $\mathbf{r}_1, \mathbf{r}_2$ and $\mathbf{r}_3$ are three distinct points taken randomly from P and $F > 0$ is an input parameter. The elements y_j, $j = 1, 2. \ldots, d$ of trial point $\mathbf{y}$ are built up by the crossover of randomly taken $\mathbf{x}$ (not coinciding with the current $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$) and $\mathbf{u}$ using the following rule:

$$y_j = \begin{cases} u_j \text{ if } U_j \le C \quad \text{or} \quad j = l \\ x_j \text{ if } U_j > C \text{ and } j \ne l \end{cases} \tag{7}$$

where l is a randomly chosen integer from $\{1, 2, \ldots, d\}$, $U_1, U_2, \ldots, U_d$ are independent random variables uniformly distributed in $[0,\ 1)$, and $C \in [0, 1]$ is an input parameter affecting the number of elements to be exchanged by crossover. Ali and Törn (2004) suggested to adapt the value of the scaling factor F during searching process according to the equation

$$F = \begin{cases} \max(F_{\min}, 1 - |\frac{f_{\max}}{f_{\min}}|) & \text{if } |\frac{f_{\max}}{f_{\min}}| < 1 \\ \max(F_{\min}, 1 - |\frac{f_{\min}}{f_{\max}}|) & \text{otherwise} \end{cases} \tag{8}$$

where $f_{\min}$, $f_{\max}$ are the minimum and maximum objective function values in the population, respectively, and $F_{\min}$ is an input parameter ensuring $F \in [F_{\min}, 1)$. The heuristic which uses (8) for evaluation of F with $F_{\min} = 0.4$ and $C = 0.9$ is denoted `CDEADP` in the following text.

4 Convergence of random combination of heuristics

When analyzing an algorithm for solving the global optimization problem, a natural question arises if the algorithm is convergent, i.e. if it is guaranteed that the sequence of the best points in population (i.e. points with minimum function values) converges to the global minimmum. This is not true in general. Let us briefly reformulate the optimization problem for the study of its convergence. Denote by $\mathcal{L}$ the system of all Lebesgue measurable subsets of D and λ the Lebesgue measure on $\mathcal{L}$. Let f be a real Lebesgue measurable function defined on D. If f is not continuous it can be impossible to find good approximations of $\mathbf{x}^* = \arg\min_{\mathbf{x} \in D} f(\mathbf{x})$. Nevertheless, using stochastic algorithms, it is always possible to find arbitrary good approximations of the value $\mu = inf\{t\,;\ \lambda(f^{-1}(-\infty, t)) > 0\}$, where $f^{-1}(A) = \{x \in D\,;\ f(x) \in A\ \}$, which is said to be the *essential minimum* of f. Therefore, our task is to find an arbitrary good approximation of μ. Notice that if the function f is continuous at its global minimum, this task is equivalent to the task of finding the global minimum.

Speaking on the convergence of an algorithm, we mean the convergence with probability 1, i.e. an algorithm is convergent if

$$P(\lim_{k \to \infty} min\{f(x);\ x \in \mathcal{P}_k\} = \mu) = 1.$$

Starting from the results of Solis and Wets (1981), we can formulate the necessary and sufficient condition for the convergence of a search algorithm as follows (Mišík, 2000).

Theorem 1 *For any measurable subset $S \subset D$ denote by $p_k(S)$ the probability that in the k-th step the algorithm produces a new trial point belonging to the*

set S. Then the algorithm is convergent if and only if for every set $S \subset D$ of positive Lebesgue measure we have $\prod_{k=1}^{\infty}(1 - p_k(S)) = 0$.

Remark 1 *Notice that according to (Šalát, 1974) the last condition is equivalent to the following one: The series $\sum_{k=1}^{\infty} p_k(S)$ is divergent.*

Now we are going to consider the convergence of an algorithm $\mathcal{A}$ (using probability distribution π) obtained by a random combination of h algorithms $\mathcal{A}_i;\ i = 1, 2, \ldots, h$, using probability distributions $\pi_1, \pi_2, \ldots, \pi_h$, respectively. The probability distribution $\pi_j(\mathcal{P}_k)$ means the probability distribution according to which the new trial point is chosen in the $k-$th step assuming that the j-th heuristic is used in this step and the current population is $\mathcal{P}_k$.

Denote by $q_i(k);\ i = 1, 2, \ldots, h;\ k = 1, 2, \ldots$, the probability of using the i-th heuristic in the k-th step of algorithm. Notice that $q_i(k) \in [0,\ 1];\ i = 1, 2, \ldots, h;\ k = 1, 2, \ldots$ and $\sum_{i=1}^{h} q_i(k) = 1;\ k = 1, 2, \ldots$. Then we have the following theorem.

Theorem 2 *Suppose that there exists $\delta > 0$ such that $q_i(k) \geq \delta$ for all $i = 1, 2, \ldots, h;\ k = 1, 2, \ldots$. Then the algorithm $\mathcal{A}$ is convergent if and only if for every $S \subset D$ of positive Lebesgue measure there exists an integer $j \in \{1, 2, \ldots, h\}$ such that the series $\sum_{k=1}^{\infty} \pi_j(\mathcal{P}_k)(S)$ is divergent. Here $\mathcal{P}_k$ denotes the population in the k-th step.*

Proof. According to Remark 1 the algorithm $\mathcal{A}$ is convergent if and only if for every set $S \subset D$ of positive Lebesgue measure we have $\sum\limits_{k=1}^{\infty} \pi(\mathcal{P}_k)(S) = \infty$. Using the definition of π, this is equivalent to

$$\sum_{k=1}^{\infty} \sum_{i=1}^{h} q_i(k) \left(\pi_i(\mathcal{P}_k)(S)\right) = \infty.$$

As all terms in the series on the left side of the last equation are non-negative numbers, the condition is equivalent to the following one:

There is an integer $j \in \{1, 2, \ldots, h\}$ such that $\sum_{k=1}^{\infty} q_j(k) \left(\pi_j(\mathcal{P}_k)(S)\right) = \infty$.

Now the statement of the theorem follows from the inequalities

$$\delta \sum_{k=1}^{\infty} \pi_j(\mathcal{P}_k)(S) \leq \sum_{k=1}^{\infty} q_j(k) \left(\pi_j(\mathcal{P}_k)(S)\right) \leq \sum_{k=1}^{\infty} \pi_j(\mathcal{P}_k)(S).$$

□

Remark 2 *The last inequalities also show that the positive uniform lower bound of the values $q_i(k)$ is necessary only for the "only if" implication in Theorem 2. The "if" implication holds without any additional conditions.*

The following corollary is a straightforward consequence of Theorem 2.

Corollary 1 *(Sufficient condition for the convergence of random combination). Algorithm $\mathcal{A}$ is convergent if there is an integer $j \in \{1, 2, \ldots, h\}$ such that $\mathcal{A}_j$ is convergent.*

The following example shows that the implication opposite to that in the preceding corollary does not hold. Thus, a random combination of algorithms can be convergent even if none algorithm among its components is convergent.

Example 1 *Let $d = 1,\ D = [-1,\ 1],\ h = 2$ and for every population $\mathcal{P}$ let $\pi_1(\mathcal{P}), \pi_2(\mathcal{P})$ be uniform probability distributions on sets $[-1,\ 0]$ and $[0,\ 1]$, respectively. Then the simple algorithms using π_1 and π_2 are evidently non-convergent, although their random combination with $q_1 = q_2 = 1/2$ is convergent.*

5 Numerical experiments and their results

One of the criteria for examining the reliability of software is the comparison of the experimental parameter values with the "certified values" from sources that are considered to be reliable. The collection of datasets (NIST, 2001) contains 27 nonlinear regression datasets (tasks) ordered according to their level of difficulty (lower – 8 tasks, average – 11 tasks, and higher – 8 tasks). The certified values are reported to 11 decimal places for each dataset. Numerical tests were carried out for all the NIST nonlinear regression tasks. The accuracy of the experimental results obtained by using our algorithms was estimated according to the number of duplicated digits when compared with the certified results. The number of duplicated digits λ can be calculated via *log relative error* (McCullough and Wilson, 2005) as

$$\lambda = \begin{cases} 0 & \text{if } \frac{|m-c|}{|c|} \geq 1 \\ 11 & \text{if } \frac{|m-c|}{|c|} < 1 \times 10^{-11} \\ -\log_{10}\left(\frac{|m-c|}{|c|}\right) & \text{otherwise} \end{cases} \tag{9}$$

where c denotes the certified value and m denotes the estimated value. According to NIST (2001), excepting the cases where the certified value is essentially zero (e.g. for the three Lanczos problems), a good nonlinear least squares procedure should be able to duplicate the certified results to at least 4 or 5 digits. Two values of the number of duplicated digits are reported in our results: λ_Q for the residual sum of squares, see (2), and λ_β for the mean of estimated parameters $(\beta_1, \beta_2, \ldots, \beta_d)$.

Several algorithms for the estimation of parameters were compared in our numerical experiments. One of them is a modification of Levenberg-Marquardt algorithm implemented in `nlinfit` procedure of Statistical Toolbox (Matlab, 2005). Each NIST dataset contains two starting d-tuples of the parameters values for iterative deterministic algorithms. The results obtained with these starting values are labelled `Start1` and `Start2` in the following text. Especially values in `Start2` are very close to the certified values. Control parameters of `nlinfit` were set to their default values.

Other algorithms are population-based, namely differential evolution (Storn and Price, 1997) and several variants of the CRS. Because of their stochastic

Table 1
Comparison of standard algorithms

	`nlinfit`				`DER`				`DEADP`			
	`Start1`		`Start2`									
Task	λ_Q	λ_β	λ_Q	λ_β	λ_Q	λ_β	*RP*	*rne*	λ_Q	λ_β	*RP*	*rne*
chwirut1	10.8	9.0	10.6	8.9	11.0	8.0	100	574	10.9	7.7	99	66
chwirut2	11.0	9.1	11.0	9.2	11.0	7.9	100	571	10.8	7.7	98	69
danwood	11.0	10.2	11.0	9.9	11.0	8.7	100	300	11.0	8.7	100	37
gauss1	11.0	8.1	11.0	8.5	11.0	8.6	100	530	10.7	8.3	97	125
gauss2	10.6	8.3	10.6	8.5	10.4	8.5	98	1039	6.0	5.4	56	192
lanczos3	10.6	8.1	10.6	8.1	0.0	0.3	0	705	0.0	0.3	0	650
misra1a	10.5	10.2	10.4	10.2	10.4	7.9	100	325	1.3	1.2	12	264
misra1b	11.0	10.2	11.0	10.2	11.0	7.9	100	386	3.1	2.5	28	230
enso	8.4	4.9	8.1	4.7	11.0	5.5	100	722	10.8	5.3	98	84
gauss3	11.0	7.7	11.0	7.7	8.8	4.3	99	1912	4.7	2.4	43	413
hahn1	10.6	7.0	10.6	6.0	0.0	0.1	0	1596	3.4	2.2	32	1037
kirby2	11.0	7.3	10.2	6.6	8.7	5.5	100	2251	10.3	6.5	94	141
lanczos1	1.9	10.7	1.9	10.7	0.0	0.3	0	746	0.0	0.3	0	678
lanczos2	10.0	8.7	9.9	8.5	0.0	0.4	0	750	0.0	0.3	0	680
mgh17			11.0	7.6	3.8	2.9	25	1714	0.1	0.8	0	730
misra1c	11.0	8.2	11.0	8.2	11.0	7.9	100	435	0.2	0.2	2	292
misra1d	11.0	10.3	11.0	10.3	11.0	7.9	100	414	0.1	0.1	1	320
nelson	10.9	8.9	10.9	9.0	10.9	8.3	100	619	0.0	0.5	0	265
roszman1	11.0	6.4	11.0	6.5	11.0	7.5	100	1245	11.0	7.4	100	73
bennett5	2.0	1.3	2.1	1.4	1.7	1.2	5	190	1.3	1.0	1	-35
boxbod			10.1	5.7	10.4	8.8	100	206	10.3	8.7	100	32
eckerle4	0.0	0.0	10.6	7.7	10.7	9.5	100	140	10.7	9.3	100	46
mgh09	3.9	2.1	8.2	4.3	9.9	5.9	100	1427	0.0	0.1	0	574
mgh10	0.0	2.0	0.0	2.5	0.0	0.3	0	478	0.0	0.1	0	57
rat42	11.0	7.0	10.9	7.0	11.0	8.6	100	474	11.0	8.4	100	59
rat43	10.0	5.7	11.0	5.9	11.0	7.9	100	904	10.8	7.6	99	78
thurber	7.8	5.0	8.2	5.2	1.5	2.1	0	1912	8.1	5.8	78	484

nature, they must be tested in repeated runs. One hundred of repetitions were carried out for each task. In addition to λ_Q and λ_β other variables are also reported. The time consumption is expressed as average number (ne) of objective function evaluations needed to reach the stopping condition or maximum allowed number of objective function evaluations (40000 d). The variability of ne is characterized by coefficient of variance vc in percents of ne. The reliability RP of the search is measured as percentage of successful searches in which $\lambda_Q > 4$ (with exception of Lanczos1 where RP is percentage of runs with $\lambda_Q > 2.4$). The stopping condition was defined in all the tasks as $R^2_{\max} - R^2_{\min} < \varepsilon$, $\varepsilon = 1 \times 10^{-15}$ for all the population-based algorithms except `CRS4HCe`. $R^2_{\max}$ and $R^2_{\min}$ are the maximum and the minimum values of the determination index R^2 defined by

$$R^2 = 1 - \frac{Q(\boldsymbol{\beta})}{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}\,. \tag{10}$$

The population size was set to $N = 10\ d$ for all the population-based algorithms. Search spaces D for the individual tasks are given in Appendix.

Algorithm 2. Adaptive mechanism in `CRS4HCe`

```
generate population P;
ε = ε0;
pass = false;
while 1 − R² < γ × ε & pass is false (outer loop)
  while R²max − R²min > ε (inner loop)
    generate a new trial point y ∈ D;
    if f(y) < f(xmax) then
      xmax := y;
      find new xmax;
    endif
    pass = true;
  endwhile; (end of inner loop)
  if pass is false then γ = γ/10 endif;
  if 1 − R² < γ × ε & pass is true then
    ε = ε/10; pass = false;
  endif;
endwhile; (end of outer loop)
```

The comparison of `nlinfit` and differential evolution is presented in Table 1. DER is standard variant of differential evolution with tuning parameters set to their recommended values, $F = 0.8$ and $C = 0.5$ (Storn and Price, 1997). `DEADP` uses adaptive F according to (8) with parameters set as in `CDEADP`. The columns denoted by rne contain the relative change of ne (in percents) when compared with the `CRS4HC`, see Table 2. Then the comparison of time

Table 2
CRS with competing heuristics

	CRS4HC				CRS5HC				CRS4HCe			
Task	λ_Q	*RP*	*ne*	*vc*	λ_Q	*RP*	*rne*	*vc*	λ_Q	*RP*	*rne*	*vc*
chwirut1	11.0	100	3008	5	11.0	100	12	5	8.5	100	-35	6
chwirut2	11.0	100	2987	4	11.0	100	13	4	8.3	100	-35	6
danwood	11.0	100	1620	7	11.0	100	13	8	8.4	100	-28	9
gauss1	11.0	100	14137	2	11.0	100	13	2	7.1	100	-35	3
gauss2	10.4	98	14726	3	10.4	98	13	3	7.0	98	-36	4
lanczos3	6.9	100	29810	19	7.0	100	11	17	7.0	100	2	23
misra1a	10.4	100	2157	8	10.4	100	12	7	7.8	100	-17	7
misra1b	10.9	100	1861	7	10.9	100	15	8	7.7	100	-19	9
enso	9.7	87	19220	7	9.0	80	13	7	8.1	86	-30	11
gauss3	11.0	100	15908	4	10.9	99	12	5	7.0	99	-35	6
hahn1	9.9	93	16509	7	9.2	87	11	7	6.5	93	-26	10
kirby2	11.0	100	8508	2	11.0	100	13	3	7.3	100	-23	3
lanczos1	0.0	0	28361	18	0.0	0	14	19	2.5	100	639	22
lanczos2	4.1	55	28251	17	4.1	40	18	20	7.1	100	8	18
mgh17	11.0	100	11023	4	11.0	100	13	5	7.5	100	-18	5
misra1c	11.0	100	2104	7	11.0	100	13	6	8.3	100	-11	8
misra1d	11.0	100	2043	7	11.0	100	12	7	8.4	100	-12	9
nelson	10.9	100	5904	4	10.9	100	12	4	9.0	100	-17	5
roszman1	11.0	100	5301	4	11.0	100	12	4	7.2	100	-36	5
bennett5	11.0	100	41335	34	11.0	100	14	35	5.1	100	-11	39
boxbod	10.4	100	1308	7	10.4	100	13	8	9.7	100	-37	9
eckerle4	10.7	100	2629	4	10.7	100	12	5	7.6	100	-35	6
mgh09	11.0	100	10422	8	11.0	100	13	9	7.7	100	-15	11
mgh10	9.0	100	20761	10	9.0	100	17	4	8.1	100	1	8
rat42	11.0	100	2942	6	11.0	100	14	5	7.4	100	-35	6
rat43	11.0	100	4807	4	11.0	100	13	4	7.9	100	-39	5
thurber	11.0	100	13915	3	11.0	100	11	2	7.4	100	-30	3

consumption of different algorithms is easier because negative values mean a less time consumption with respect to `CRS4HC`, while positive values indicate a bigger one. The results of CRS algorithms with competing heuristics are given in Table 2. The common tuning parameters were set as follows, w_0 used in (4), $w_0 = 0.5$, δ for the reset of probabilities q_i to initial values, $\delta = 0.05$. `CRS4HC` denotes the algorithm with four competing heuristics described in section 3, in `CRS5HC` the fifth heuristic generating a random point from uniform distribution in D is added to make the algorithm convergent with probability 1. Finally, `CRS4HCe` stands for the algorithm that contains the same four heuristics as `CRS4HC` and, moreover, provides a procedure for adapting the value ε in the stopping condition. The value ε starts from input value ε_0 and can be decreased if $1 - R^2 < \gamma \times \varepsilon$ where $\gamma \gg 1$ is another input parameter. More details are given in Algorithm 2. The statement at line 14 prevents the endless outer *while*

Table 3
CRS without competition

	CRS4HA			REFL1			REFL25			REFLB			CDEADP		
Task	*RP*	*rne*	*vc*	*RP*	*rne*	*vc*	*RP*	*rne*	*vc*	*RP*	*rne*	*vc*	*RP*	*rne*	*vc*
chwirut1	100	17	5	84	47	57	100	1922	4	100	-10	4	71	71	65
chwirut2	100	19	5	71	74	60	100	1906	4	100	-11	4	74	62	63
danwood	100	40	7	81	3	46	100	406	6	0	4838	0	88	-3	41
gauss1	100	20	2	24	243	122	0	2164	0	100	-13	3	30	250	134
gauss2	96	20	3	38	280	129	0	2073	0	86	-14	6	36	339	133
lanczos3	100	28	24	0	444	44	0	705	0	100	-35	19	0	510	36
misra1a	100	30	7	2	59	23	100	359	7	0	3609	0	0	61	21
misra1b	100	36	8	9	57	29	100	367	8	0	4199	0	5	64	24
enso	80	22	10	86	56	123	0	1773	0	63	-14	10	78	42	52
gauss3	100	20	4	4	532	86	0	1912	0	91	-17	8	3	475	88
hahn1	91	19	7	0	992	32	0	1596	0	62	-25	13	0	1057	29
kirby2	100	19	3	34	261	115	0	2251	0	100	-15	3	27	188	58
lanczos1	0	36	21	0	523	38	0	746	0	0	-34	21	0	520	38
lanczos2	55	34	24	0	475	41	0	750	0	48	-36	20	0	467	41
mgh17	100	18	3	0	381	83	0	1714	0	100	-18	14	0	451	85
misra1c	100	33	6	0	62	18	100	326	8	0	3702	0	1	63	18
misra1d	100	31	8	0	73	17	100	340	9	0	3816	0	1	72	19
nelson	100	12	4	0	88	26	100	1417	7	100	-15	3	0	88	26
roszman1	100	19	4	91	51	62	100	2918	0	100	-12	3	96	39	55
bennett5	100	3	38	1	-69	24	0	190	0	100	-32	39	1	-71	25
boxbod	100	38	6	90	-1	46	100	510	6	100	6016	0	86	6	54
eckerle4	100	19	4	94	3	59	100	2297	4	100	1	5	95	5	75
mgh09	100	20	8	0	184	40	0	1435	0	100	-13	9	0	178	38
mgh10	100	15	4	0	-50	20	0	478	0	100	-25	10	0	-48	20
rat42	100	19	6	81	42	69	100	2016	4	100	-8	6	83	42	69
rat43	100	19	3	87	83	70	100	3228	0	100	-8	4	82	66	62
thurber	100	20	2	1	824	26	0	1912	0	100	-20	2	1	845	29

loop. Values of these input parameters of `CRS4HCe` were set to $\varepsilon_0 = 1 \times 10^{-9}$ and $\gamma = 1 \times 10^7$.

The benefit of competition is evident when we compare the results of competitive CRS algorithms with the results of non-competitive ones in Table 3. As regards `CRS4HA`, the same four heuristics as in `CRS4HC` do not compete but alternate only, with constant probabilities $q_i = 1/4$, $i = 1, 2, 3, 4$. The remaining columns give the results obtained by CRS with only one heuristic, the algorithms are denoted according to the names of heuristics. The values of $vc = 0$ mean that all the runs of given task were stopped because of exceeding the limit value of $ne = 40000\ d$.

6 Discussion

The results obtained by `nlinfit` procedure can be accepted for all the tasks of lower difficulty level. In other tasks `nlinfit` crashes in two cases and does not perform well in most tasks of higher difficulty. The reliability of differential evolution (DE) is poor in many cases when compared with competitive CRS in spite of the fact that the time consumption is higher. The results of two variants of DE differ from each other in both *RP* and *ne*, which indicates a high sensitivity of DE to tuning parameters setting.

`CRS4HC` is reliable enough for most tasks except Lanczos1 and Lanczos2 (caused by their extremely low values of $1 - R^2$, see Table 4). As regards other tasks, *RP* is less than 100 only in four tasks but it always exceeds 85 percents. The addition of the fifth heuristic to `CRS5HC` does not increase *RP* despite the theoretical expectation but only makes time consumption greater by about one tenth or more. The best results in both *RP* and *ne* were achieved by using `CRS4HCe` where the use of adaptive stopping condition decreases *ne* by one tenth to one third in most cases and increases *RP* in tasks with small value of $1 - R^2$. The results of competitive CRS also show that higher values of *vc* indicate a high time consumption of search, which should be taken into account when proposing new variants of adaptive algorithms for the global optimization.

As it is clear from Table 3, `CRS4HA` algorithm (where the cooperation of four heuristics comes through their alternation) outperforms all the CRS algorithms with only one heuristic. When comparing the results of `CRS4HA` and `CRS4HC`, we can state that the addition of competition gives another improvement in both *ne* and *RP*.

Some descriptive characteristics of `CRS4HCe` are given in Table 4. This table contains the average values of the relative successfulness in percents (column *rsuc*), the average number of resets per 1000 function evaluations (column *rst1000*) and average relative frequencies in percents of the use of competing heuristics (the last four columns). The values of $\log_{10}(1 - R^2)$ are computed from the certified values of residual sums of squares, while the values of $\log_{10}(\varepsilon)$ are obtained by using the adaptive mechanism described in Algorithm 2. Most values of the characteristics in Table 4 indicate that Lanczos1 task is very outlying from the others, which is caused mainly by its an extremally small random component of the model ($\log_{10}(1 - R^2) = -25.9$). The task is based on generated data and such small values of $(1 - R^2)$ are not supposed to occur in applications. As we can see from Table 4 (columns REFL1 to DEADP), the frequencies of the use of heuristics differ from one to another task, which shows the ability of `CRS4HCe` to adapt the search strategy to a particular task. Similarly, values of ε in the stopping condition are also capable of adapting

Table 4
Descriptive characteristics of the `CRS4HCe`

Task	*rsuc*	*rst1000*	log $(1-R^2)$	$\log \varepsilon$	REFL1	REFL25	RBEST	DEADP
chwirut1	52.8	20.7	-1.7	-9	27.3	3.4	35.9	33.3
chwirut2	52.7	20.0	-1.9	-9	27.4	3.7	36.6	32.3
danwood	41.0	26.6	-3.2	-11	41.5	6.8	2.5	49.2
gauss1	42.3	21.2	-2.5	-10	30.5	0.6	32.2	36.6
gauss2	40.0	19.9	-2.5	-10	31.8	0.7	33.6	33.9
lanczos3	40.7	20.7	-8.8	-16	38.2	1.9	43.8	16.1
misra1a	42.0	24.6	-4.7	-12	40.7	7.0	3.4	48.9
misra1b	39.4	23.0	-5.0	-12	43.9	8.0	3.3	44.8
enso	33.6	15.9	-0.4	-9	27.8	0.7	28.8	42.7
gauss3	39.7	19.8	-2.5	-10	33.6	0.8	36.6	29.0
hahn1	35.3	16.6	3.7	-11	35.9	1.5	38.3	24.3
kirby2	39.1	18.3	-4.5	-12	34.0	1.7	36.2	28.2
lanczos1	7.7	4.0	-25.9	-31	39.2	1.5	44.3	15.0
lanczos2	40.1	20.5	-11.7	-19	37.9	1.8	43.7	16.5
mgh17	41.2	20.1	-4.3	-12	34.0	1.6	38.4	26.0
misra1c	43.1	22.6	-5.2	-13	44.4	8.5	4.2	42.9
misra1d	43.5	22.5	-5.1	-13	42.4	8.2	4.2	45.2
nelson	46.9	17.4	-1.2	-9	29.4	3.5	31.2	36.0
roszman1	39.3	19.6	-2.8	-10	30.6	1.8	33.1	34.6
bennett5	42.1	16.1	-5.0	-12	31.8	3.7	39.3	25.2
boxbod	47.8	27.4	-0.9	-9	32.6	4.9	5.1	57.4
eckerle4	48.0	24.0	-2.5	-10	24.0	2.5	28.9	44.7
mgh09	43.9	19.9	-2.2	-10	31.5	2.4	35.9	30.2
mgh10	45.7	17.7	-7.2	-15	31.5	3.5	37.8	27.2
rat42	42.3	20.2	-2.8	-10	27.9	2.9	31.9	37.3
rat43	40.9	21.2	-2.1	-10	29.4	1.6	33.0	36.0
thurber	37.2	19.3	-3.3	-11	37.2	0.7	39.1	23.0
min	7.7	4.0	-25.9	-31.0	24.0	0.6	2.5	15.0
max	52.8	27.4	-0.4	-9.0	44.4	8.5	44.3	57.4
average	41.0	20.0		-12.0	33.9	3.2	28.9	33.9

themselves to a task.

As regards the *rsuc* characteristics, its values exceed 40 % in most tasks. When compared with evolutionary algorithms where *rsuc* takes the values about 20 %, it indicates that the search does not vast time by evaluating objective functions in the points that are not useful. The *rst1000* characteristics takes the values about 20 with a small variation.

7 Conclusions

Both adaptive search algorithms with four competing heuristics (`CRS4HC` and `CRS4HCe`) proved to be more reliable than `nlinfit` with acceptable time consumption (one objective function evaluation takes about 1 millisecond on PC) for all the NIST nonlinear datasets and, therefore, they can be used instead of deterministic algorithms in the parameter estimation. `CRS4HCe` performed well with the default setting of its input parameters in all the 27 tasks and no special tuning known from the applications of classic evolutionary algorithms was needed. In contrast to Levenberg-Marquardt or Gauss-Newton deterministic algorithms, these stochastic search procedures are not dependent on the starting values of the parameters to be estimated.

The adaptive search algorithms were implemented in Matlab and their source code (M-files) is available on the author's web site. They are free for download and use in tests and applications. A more user-friendly version of the program is under preparation and it will appear in near future.

A Appendix. Search spaces of NIST tasks

Table A.1
Search spaces – Part 1

i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i
Task	enso		gauss1,2,3		hahn1		thurber		lanczos1,2,3	
1	0	20	1	500	0	10	1000	1500	0	10
2	0	10	0	0.1	-0.5	0.5	0	2000	0	8
3	0	10	1	300	-0.05	0.05	0	2000	0	10
4	1	100	50	150	-0.001	0.001	0	1000	0	8
5	-2	2	1	50	-0.1	0.1	-2	5	0	10
6	-2	2	1	300	-0.01	0.01	0	10	0	8
7	1	100	100	200	-5E-4	5E-4	0	1		
8	-2	2	1	50						
9	-2	2								
Task	kirby2		mgh09		mgh10		mgh17		rozsman1	
1	0	10	0	100	0	100	0	10	0	1
2	-1	0	0	100	0	$1E6$	0	10	-5E-4	5E-4
3	0	1	0	100	0	$1E5$	-20	-0.001	0	6000
4	-1	0	0.01	100			0	1	-500	0
5	0	1					0	1		
Task	chwirut1		chwirut2		nelson		danwood		boxbod	
1	0	10	0	10	1	10	0	100	1	1000
2	0	1000	0	1000	0	1	1	10	0.1	2
3	0	100	0	100	-1	0				

Table A.2
Search spaces – Part 2

i	a_i	b_i	a_i	b_i	a_i	b_i	a_i	b_i
Task	misra1a		misra1b		misra1c		misra1d	
1	-10000	10000	0	2000	0	10000	0	10000
2	-0.2	0.2	0	0.1	0	1	0	1
Task	bennet5		eckerle4		rat42		rat43	
1	-5000	-1000	0	10	0	1000	0	1000
2	0	500	1	10	0	10	0	100
3	0.1	10	400	500	0	1	0	1
4							0.1	10

Acknowledgements

This work was supported by the grant 201/05/0284 of the Czech Grant Agency and by the research scheme MSM 6198898701 of the Institute for Research and Applications of Fuzzy Modeling.

References

Ali, M.M., Törn, A., 2004. Population set based global optimization algorithms: Some modifications and numerical studies, Computers and Operations Research 31, 1703 – 1725.

Bäck, T., 1996. Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York.

Deb, K., 2005. A population-based algorithm-generator for real parameter optimization. Soft Computing 9, 236 – 253.

Křivý, I., Tvrdík, J., 1995. The controlled random search algorithm in optimizing regression models. Comput. Statist. and Data Anal. 20, 229 – 234.

MATLAB, version 7.1.0. The MathWorks, Inc., 2005.

McCullough, B.D.,Wilson, B., 2005. On the accuracy of statistical procedures in Microsoft Excel 2003. Comput. Statist. and Data Anal. 49, 1244 – 1252.

Mišík, L., 2000. On convergence of a class of evolutionary algorithms. In: Proceedings of MENDEL 2000, 6th International Conference on Soft Computing, Technical University Press, Brno, 97 – 100.

Nelder, J.A., Mead, R., 1964. A simplex method for function minimization. Computer J. 7, 308 – 313.

Price, W.L., 1977. A controlled random search procedure for global optimization. Computer J. 20, 367 – 370.

Statistical Reference Datasets. Nonlinear regression. NIST Information Technology Laboratory. http://www.itl.nist.gov/div898/strd/. 2001

Šalát, T., 1974. Infinite Series (in Slovak). Academia, Prague.

Solis, F. J. and Wets, R. J-B., 1981. Minimization by random search techniques. Mathematics of Operations Research, 6, 19 – 30.

Spall, J. C., 2003. Introduction to Stochastic Search and Optimization: Estimation, Simulation and Control. Wiley-Interscience

Storn, R., Price, K., 1997. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11, 341 – 359.

Tvrdík, J., Křivý, I., Mišík, L., 2001. Evolutionary algorithm with competing heuristics. In: Ošmera, P. (ed) MENDEL 2001, 7th International Conference on Soft Computing. Technical University, Brno, 58 – 64.

Tvrdík, J., Mišík, L., Křivý, I., 2002. Competing heuristics in evolutionary

algorithms. In: Sinčák, P. et al. (eds) Intelligent Technologies – Theory and Applications. IOS Press, Amsterdam,159 – 165.

Tvrdík, J., 2004. Generalized controlled random search and competing heuristics. MENDEL 2004, 10th International Conference on Soft Computing, (Matoušek R. and Ošmera P. eds). University of Technology, Brno, 228 – 233.

Tvrdík, J., Křivý, I., 2004. Comparison of algorithms for nonlinear regression estimates. In: Antoch, J. (ed) COMPSTAT 2004. Physica-Verlag, Heidelberg New York, 1917 – 1924.

Tvrdík, J., 2005. Competition and cooperation in evolutionary algorithms: A comparative study. In: Proceedings of MENDEL 2005, 11-th Int. Conference on Soft Computing (Matoušek R. and Ošmera P. eds), Technical University Press, Brno, 108 – 113.

Tvrdík, J., Křivý, I., Mišík, L., 2006. Adaptive population-based algorithm for global optimization. COMPSTAT 2006 (submitted)

Winter, G., Galvan, B., Alonso, S., Gonzales, B., Jimenez, J.I., Greimer, D., 2005. A flexible evolutionary agent: cooperation and competition among real-coded evolutionary operators. Soft Computing 9, 299 – 323.

Wolpert, D. H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1, 67 – 82.