A Hybrid Global Optimization Algorithm Involving Simplex and Inductive Search

Chetan Offord[1] and Željko Bajzer[2]

[1] Biomathematics Resource, Mayo Clinic
200 1st St. SW, Rochester, MN 55905, USA
`offord.chetan@mayo.edu`
[2] Biomathematics Resource and Department of Biochemistry & Molecular Biology,
Mayo Clinic, Mayo Medical and Mayo Graduate School
200 1st St. SW, Rochester, MN 55905, USA
`bajzer@mayo.edu`

Abstract. We combine the recently proposed inductive search and the Nelder-Mead simplex method to obtain a hybrid global optimizer, which is not based on random searches. The global search is performed by line minimizations (Brent's method) and plane minimizations (simplex method), while the local multidimensional search employs the standard and a modified simplex method. Results for the test bed of the Second International Competition of Evolutionary Optimization and for another larger test bed show remarkable success. The algorithm was also efficient in minimizing functions related to energy of protein folding.

1 Introduction

Nelder-Mead simplex method [1] for minimization has been used in various fields and especially for optimization in chemistry [2], [3]. It is known as a robust method which does not use derivatives, and has the advantage of easy implementation [4]. It is also known that the simplex search works best for low-dimensional problems and that the position and size of the starting simplex are important for the success of the search. Within the context of global optimization, the simplex method was successfully used as a component of hybrid algorithms that involve adaptive random search [5] or genetic algorithms [6].

Here we propose a hybrid algorithm (SIH) for global minimization based on a combination of inductive search [7] and multiple simplex searches. Two general characteristics of the proposed algorithm are different from standardly accepted paradigms. First, we do not use random numbers in our search as is done in many current global optimizers. This makes our algorithm immune from variations generated by random numbers. Second, besides using line minimizations (frequently employed in classical minimization methods) we also use multiple plane minimizations, i.e. we try to find the minimum in pairs of variables while the rest are kept constant. This feature makes it possible to capture some interdependencies among variables as they approach their values at the minimum, and yet it is less expensive in terms of function calls than an extensive n-dimensional search.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2074, pp. 680–688, 2001.

2 The Algorithm

The search domain for a given objective function $f(\mathbf{x})$ is defined by a n-dimensional box $\{\mathbf{x}|\mathbf{x} = (x_1, \ldots, x_n) \in \mathbf{R}^n,\ x_i \in [b_i^l, b_i^u],\ i = 1, \ldots, n\}$, where b_i^l and b_i^u are lower and upper limits for the variable x_i respectively.

Phase 1. Inductive search follows Bilchev and Parmee [7]. This search requires that the objective function $f(\mathbf{x})$ is also an explicit function of space dimension n. Thus, it is assumed that the following sequence of functions can be defined:

$$\phi(1, x_1),\ \phi(2, x_1, x_2),\ \phi(3, x_1, x_2, x_3),\ \ldots \phi(n, x_1, \ldots, x_n) \equiv f(x_1, \ldots, x_n) \equiv f(\mathbf{x}) \tag{1}$$

For many functions usually used in testing global minimization algorithms, this is naturally true, as they are defined for any dimension n, i.e. they are also explicit functions of n. However, in many applications this is not true; therefore we define the sequence of functions as follows:

$$\phi(i, x_1, \ldots, x_i) = f(x_1, \ldots, x_i, x_{i+1}^s, x_{i+2}^s, \ldots, x_n^s), \quad i = 1, \ldots, n \tag{2}$$

where $(x_1^s, \ldots, x_n^s)$ is the point obtained by an initial n-dimensional search with the simplex algorithm based on the implementation in [4]. The initial simplex for that search is defined by points $\mathbf{p}_j = \mathbf{c} + 0.5[\mathbf{e}_{j-1}(\mathbf{d} \cdot \mathbf{e}_{j-1}) - \mathbf{d}/(n+1)],\ j = 1 \ldots, n+1$, where $\mathbf{e}_0 = \mathbf{0}$, and $\mathbf{e}_i, i = 1, \ldots, n$ are unit vectors of $\mathbf{R}^n$, $\mathbf{d} = (b_1^u - b_1^l, \ldots, b_n^u - b_n^l)$ determines the dimensions of the n-dimensional box, and $\mathbf{c} = \mathbf{d}/2$ is its central point. Points $\mathbf{p}_j$ define a simplex with its center of gravity in the central point of the search domain. The induction algorithm now goes as follows:

1. The minimum of $\phi(1, x_1)$ at $x_1 = \xi_1$ is found by a line minimization. We used Brent's method as implemented in [4]; see details below.
2. The minimum of $\phi(2, \xi_1, x_2)$ at $x_2 = \xi_2$ is found by a line minimization.
3. The current "minimum" of $\phi(2, x_1, x_2)$ at (ξ_1, ξ_2) is now improved by a two-dimensional simplex minimization (algorithm ESIMP2 described below) to obtain a better or equal value at point (ξ_1^1, ξ_2^1).
4. The minimum of $\phi(3, \xi_1^1, \xi_2^1, x_3)$ at $x_3 = \xi_3$ is found by line minimization.
5. The current best point of $\phi(3, x_1, x_2, x_3)$ at $(\xi_1^1, \xi_2^1, \xi_3)$ is then improved by a three-dimensional simplex minimization (algorithm SIMP(i, r), $i = 3$, described below) to obtain a better or equal value at point $(\xi_1^2, \xi_2^2, \xi_3^2)$.
6. The process described in steps 4 and 5 is now repeated for functions (2), $i = 4, \ldots, n$, i.e. each time a line minimization based on the previous best point is performed and subsequently improved by an i-dimensional minimization (algorithm SIMP(i, r)). At the end, for $i = n$, the full function $f(x_1, \ldots, x_n)$ is minimized by a n-dimensional simplex algorithm with the best point $(\zeta_1, \ldots, \zeta_n)$.

The line minimization by Brent's method initially requires a bracketing triplet [4]. For a given variable x_i we choose the triplet (b_i^l, η_i, b_i^u), where for the function $g(x)$ to be minimized $g(\eta_i)$ is smaller than both $g(b_i^l)$ and $g(b_i^u)$. The point η_i is determined by evaluating the function at n_1 equidistant points within the interval $[b_i^l, b_i^u]$. If by chance $\eta_i = b_i^l$ or $\eta_i = b_i^u$, then the line search is omitted and this point is considered the result of the line minimization. With some experimentation we found that optimal n_1 is around 200 and chose $n_1 = 176$.

The simplex minimizations use the stopping criterion as in [4] with the tolerance $\epsilon = 10^{-7}$. In a simplex search, if a coordinate x_i of a vertex moves outside $[b_i^l, b_i^u]$, it is returned back within that interval to the closest border by a small distance defined as $\delta_i = 0.0001 d_i |\sin(2.2 n_2)|$. Here n_2 is the total number of previous boundary crossings. This is designed to keep the point close to the border, but in various positions, to prevent degeneracy of the simplex.

Phase 2. While Phase 1 can yield the global minimum, sometimes the search must be extended further. We accomplish this by combining plane and n-dimensional simplex minimizations, while always utilizing the position of the current best minimum.

Plane minimizations. These are performed in sequence for each possible pair of variables; the remaining variables are resting on the values of the previous best point (initially the result from Phase 1). The sequence of pairs of variables is given by (x_j, x_{j+i}), $i = 1, \ldots, n-1$, $j = 1, \ldots, n-i$, and the procedure for extensive plane minimizations (algorithm ESIMP2) is described below. When the minimization for a given pair of variables results in a significant relative decrease of the function value $(> \sqrt{\epsilon})$, the plane minimizations are repeated for all pairs previously achieving a significant decrease. The first pair to achieve a significant decrease after Phase 1 is simply remembered and the minimization is not repeated. The described repetition procedure is also employed after the plane minimization for the last pair of variables in the sequence. We consider this last effort by repetition as a natural endpoint of SIH.

Multidimensional minimizations. When the plane minimizations for $\mathrm{Int}(n/2)$ consecutive pairs (not counting repeated plane minimizations) are completed, we then interrupt the sequence of plane minimizations and perform two n-dimensional simplex minimizations to take into account all variables simultaneously. Both minimizations are initialized as $\mathrm{SIMP}(n, r)$ with the current best point and with $r = r_m = 0.7 + (m_1 - 1)0.01$, where m counts the number of n-dimensional minimizations and $m_1 = m \bmod 30$ (to avoid $r_m = 1$, which would introduce initial simplex points on the border of the search domain). The two n-dimensional minimizations differ in the basic simplex algorithm. The first is designed to avoid a rapid contraction towards the best point, so that the space is searched more exhaustively. This is achieved by contracting only the worst point towards the best point when in the standard simplex method all points are contracted towards the best point. The second n-dimensional minimization employs the standard method [4].

Sub-algorithm ESIMP2 . The search in a plane consists of five minimizations. The first four simplex minimizations are intended to search the rectangular domain of the plane extensively and independently of the previous best minimum. They are started with the initial triangles as depicted in Fig. 1. The fifth two-dimensional search is initialized by the best point found before the four plane minimizations, and the two points which yielded the lowest function values in those four minimizations. In the fifth plane minimization we use the non-standard simplex method with the slow contraction to the best point in order to find a better minimum in the neighborhood of the current one.

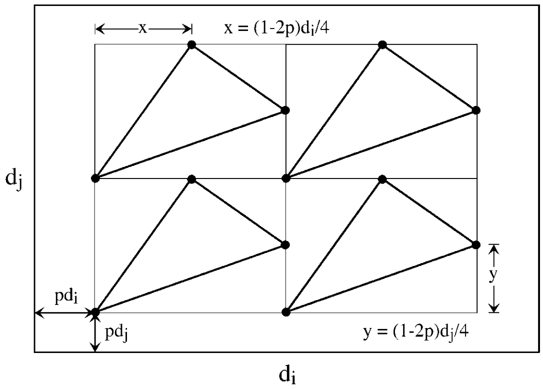

Fig. 1. Initial simplexes (triangles) for plane minimizations. In the ESIMP2 sub-algorithm we used $p = 0.12$

Sub-algorithm SIMP(i, r). Here we use the implementation of simplex method by Press et al., [4]. It is used in Phase 1 for i-dimensional minimizations, $i = 3, \dots, n$ and in Phase 2 for $i = n$. The algorithm starts with the following initial simplex: $\mathbf{s}_j = (s_1^j, \dots, s_n^j)$, $s_i^j = \zeta_i + r\sigma(b_i^u + b_i^l - 2\zeta_i)\max(b_i^u - \zeta_i, \zeta_i - b_i^l)$, $j = 1, \dots, n+1$, where $\sigma(x) = 1$ for $x > 0$ and $\sigma(x) = -1$ for $x \leq 0$. The point $(\zeta_1, \dots, \zeta_n)$ is the current best minimum. For Phase 1, $r = 0.7$ and for Phase 2 as defined above.

3 Testing and Comparisons

To compare the efficiency of SIH with other algorithms, we found most suitable the published results of the Second International Competition on Evolutionary Optimization [8], [6]. In Table 1 we compare the results of SIH on the test bed, consisting of seven unconstrained minimizations with the results obtained by Differential Evolution (DE) [8], and the Simplex Genetic Hybrid (SGH) algorithm [6]. These appear to be the only results of the competition which were published.

Table 1. Comparisons for minimization of test functions: Generalized Rosenbrock (RO, $M = 5000$), Odd Square (OS, $M = 5000$), Modified Langerman (ML, $M = 3750$), Shekel's Foxholes (SF, $M = 16000$), Epistatic Michalewicz (EM, $M = 1250$, and Storn's Chebyshev Polynomials (CP, for $n = 9$, $M = 1500$ and for $n = 17$, $M = 10000$). $\phi(1, x)$ is not defined for RO function; therefore we used sequence (2). The functions and the corresponding search domains were obtained from K. Price; see also [8].

$F(\mathbf{x})$	Alg.	n	VTR	M	$2M$	$4M$	$8M$	$16M$	BVAT
RO	DE	10	10^{-6}	30.54	4.844	0.059	VTR	VTR	0
	SGH			7.93	3.75	0.122	$3.8 \cdot 10^{-4}$	VTR	$1.8 \cdot 10^{-9}$
	SIH			VTR	VTR	VTR	VTR	•	$8.5 \cdot 10^{-20}$
OS	DE	10	-0.999	-0.415	-0.735	-0.843	-0.859	-0.870	-0.873
	SGH			-0.173	-0.178	-0.178	-0.178	-0.178	-0.635
	SIH			NFD	VTR	VTR	VTR	•	-1.143
ML	DE	10	N/A	-0.348	-0.176	-0.460	-0.687	-0.834	-0.965
	SGH			-0.315	-0.487	-0.510	-0.510	-0.510	-0.655
	SIH			NFD	-0.965	-0.965	•		-0.965
SF	DE	10	N/A	-0.967	-2.67	-7.50	-10.1	-10.2	-10.208
	SGH			-1.477	-1.477	-1.477	-1.477	-1.477	-1.706
	SIH			-10.208	•				-10.208
EM	DE	10	-9.66	-5.77	-6.79	-7.81	-8.90	-9.44	-9.660
	SGH			-6.29	-6.29	-8.62	-8.94	-8.94	-9.556
	SIH			NFD	NFD	NFD	-8.34	VTR	-9.660
CP	DE	9	10^{-6}	$5.88 \cdot 10^{4}$	$4.12 \cdot 10^{3}$	46.4	0.0223	VTR	0
	SGH			9924	4442	3089	2018	835	11.85
	SIH			NFD	NFD	NFD	VTR	VTR	0
CP	DE	17	10^{-6}	$4.9 \cdot 10^{6}$	$3.5 \cdot 10^{4}$	11.8	$3.9 \cdot 10^{-5}$	VTR	0
	SGH			$6.4 \cdot 10^{6}$	$1.4 \cdot 10^{6}$	$3.9 \cdot 10^{5}$	$2.3 \cdot 10^{5}$	$1.2 \cdot 10^{5}$	$3.0 \cdot 10^{4}$
	SIH			NFD	NFD	NFD	2445	573	573

Corresponding to the difficulty in the minimization of a given function, each minimization is characterized by a different number M of function evaluations for the first check point. Subsequent check points are given as multiples of M. For each check point the corresponding value of the function is displayed. Value to reach (VTR) is a value presumed to lie within the basin of attraction which contains the global minimum, and if it is reached, the search is considered successful. The values obtained at different check points for DE and SGH correspond to the average of best values reached in 20 independent searches, differed by realization of random numbers. The best value attained BVAT is the lowest result achieved in any of the 20 searches at the last check point ($16M$). In the case of our nonrandom algorithm, we present the result of a single search in each column and the value (BVAT) as either the value obtained when SIH completed the search (all plane minimizations in Phase 2 performed, signified by •) or when the last check point had been reached, whichever happened first. In some cases the first few check points were reached before the function attained full dimensionality in the Phase 1; this is signified by NFD. The results of Table 1 show that our algorithm performed better than the DE algorithm in all cases but the last one, and much better than the SGH algorithm. We note that K. Price has communicated to us that the current version of DE is more efficient than the version from Table 1.

Table 2. Comparison of achievement of various algorithms after 20000 function calls for functions described in [9]: Odd Square (OS′ [F29]), Ackley (AC [F20]), Shekel's Foxholes (SF [F14]), Griewank (GR [F16]), Katsuuras (KA [F18]), Rastrigin (RA [F19]), Rosenbrock (RO′ [F2]), Epistatic Michalewicz (EM [F28]), Neumaier no. 3 (N3 [F5]), Hyper-Ellipsoid (HE [F4]), Storn's Chebyshev Polynomials (CP [F30]), Neumaier no. 2 (N2 [F26]), Modified Langerman (ML′ [F27]), Shekel-7 (S7 [F7]), Colville (CO [F25]).

$F(\mathbf{x})$	n	SIH	ASA	CS	DE	GE3	GEIII	VFSR	RAND
OS′	10	$\underline{-2.65}$	-0.015	-0.038	-0.137	-0.407	-0.772	-0.027	-0.089
AC	30	0.205^*	2.77	6.47	6.46	2.09	1.40	2.78	5.88
OS′	5	$\underline{-1.81}$	-0.409	-0.500	-0.847	-0.727	-0.898	-0.127	-0.577
SF	5	$\underline{-10.4}$	-2.67	-3.47	-3.18	$\underline{-10.4}$	-2.67	-3.29	-2.14
GR	10	$\underline{0}$	0.176	0.066	1.39	0.178	0.636	0.067	20.6
KA	10	$\underline{1.00}$	10.5	68.5	417	$\underline{1.00}$	316	9.70	199
RA	10	$\underline{0}$	2090	$4 \cdot 10^5$	10^5	803	2490	1910	$7 \cdot 10^5$
RO′	10	$\underline{2 \cdot 10^{-21}}$	2.97	$\underline{4 \cdot 10^{-10}}$	429	8.18	8.85	0.147	5140
EM	5	$\underline{-4.69}$	-3.72	-2.79	-4.12	-4.64	-3.82	-3.84	-3.24
SF	10	$\underline{-10.2}$	-1.47	$\underline{-10.2}$	-0.859	-1.42	-1.41	-1.47	-0.337
EM	10	$\underline{-9.66}$	-7.66	-7.38	-4.11	-7.55	-6.65	-9.03	-5.29
N3	25	$\underline{-2900}^*$	$2 \cdot 10^4$	$\underline{-2900}$	$2 \cdot 10^5$	$3 \cdot 10^4$	$2 \cdot 10^3$	$3 \cdot 10^4$	$9 \cdot 10^5$
N3	30	$\underline{-4930}^*$	10^5	$\underline{-4930}$	$2 \cdot 10^6$	$4 \cdot 10^4$	$2 \cdot 10^4$	10^5	$2 \cdot 10^6$
HE	30	$\underline{4 \cdot 10^{-12*}}$	14.7	$\underline{10^{-21}}$	193	9.71	5.02	15.0	1060
N3	15	$\underline{-665}$	1280	$\underline{-665}$	$2 \cdot 10^4$	240	-178	520	$5 \cdot 10^4$
N3	20	$\underline{-1520}^*$	9600	$\underline{-1520}$	$3 \cdot 10^5$	-609	892	2490	$2 \cdot 10^5$
CP′	9	$\underline{0}$	3400	3.94	$5 \cdot 10^5$	8660	722	2220	$7 \cdot 10^4$
N2	4	$\underline{4 \cdot 10^{-21}}$	0.247	0.217	0.227	0.267	0.239	0.238	0.282
ML′	5	-0.940	$\underline{-0.965}$	-0.482	$\underline{-0.965}$	-0.501	-0.513	-0.908	-0.510
S7	4	$\underline{-10.4}$	-2.75	$\underline{-10.4}$	$\underline{-10.4}$	-5.08	$\underline{-10.4}$	-5.12	-2.83
RO′	5	$\underline{7 \cdot 10^{-20}}$	0.515	$\underline{9 \cdot 10^{-11}}$	$\underline{8 \cdot 10^{-7}}$	2.80	3.59	0.338	10.6
N3	10	$\underline{-210}$	-205	$\underline{-210}$	2140	-39.1	-50.7	-208	2890
ML′	10	-0.806	-0.274	-0.003	-0.040	-0.513	-0.011	-0.513	-0.005
CO	4	$\underline{10^{-20}}$	0.121	$\underline{2 \cdot 10^{-11}}$	$\underline{4 \cdot 10^{-7}}$	0.385	3.29	0.198	25.0

Further testing is based on a very detailed and thorough work of Janka [9] in which he compared the efficiency of eight global minimization algorithms, four additional variants, and a simple random search, on the test bed of thirty functions F1-F30, some in multiple dimensions. The algorithms we included here are: Adaptive Simulated Annealing (ASA) [10], [11], Very Fast Simulated Reannealing (VSFR) [12], [11], clustering with single linkage (CS) [13], [14], Differential Evolution (DE) [8], and Genocop algorithms (GE3, GEIII) [15], [16]. Janka also considered algorithms SIGMA, PGAPack, which however were outperformed by the above mentioned ones. We chose to compare Janka's best variants of ASA, CS and VSFR with SIH. It is noteworthy that the actual implementation of the algorithms Janka used for his study, were based on the software available via the Internet; he did not consult the authors for possible optimal settings [9].

In Table 2 we compare the results of Janka for problems which he classified as the most difficult minimizations [9] with the results of SIH. Some of the problems are the same as in Table 1 and they are designated with the same abbreviation. Some of the problems are essentially the same as in Table 1, except for small differences in the definition of the function and/or search domain; these are denoted with a prime. The rounded values obtained after 20000 function calls are shown. Those underlined represent the function values obtained within the basin of attraction containing the global minimum. With an asterisk, we denoted the situation when the corresponding function is defined in such a way that sequence (1) can be used, but we had to use sequence (2) because Phase 1 of our algorithm had not reached the full dimensionality before 20000 function calls. This happened for problems with $n \geq 20$. Table 2 clearly reveals that our algorithm significantly outperformed all others. The global optimum was found in all but three cases. In two of those problems (AC, ML$'$ for $n = 10$), our values were the lowest anyway and only in one case (ML$'$ for $n = 5$) our value was the second lowest.

Our final comparison is based on a very recent work by Mongeau et al. [17], who compared the efficiency of public domain software for global minimization, i.e. Adaptive Simulated Annealing (ASA), clustering algorithm GLOBAL, genetic algorithm GAS, multilevel random search GOT, integral global optimization INTGLOB, and interactive system for universal functional optimization (UFO). We compare the results of the two highest dimensional functions: protein folding energy minimizations in 15 (pf5) and 18 dimensions (pf6). Table 3 indicates that SIH is comparable to UFO (variant with the best results) and outperformed all others.

Table 3. Comparisons for minimization of protein folding functions pf5 and pf6 as defined in [17]. Displayed is the number of function evaluations required to attain the minima -9.10 for pf5 and -12.71 for pf6. Algorithms which did not reach the minima in 17000 function evaluations for pf5 and 25000 for pf6, are marked by N.R.

$F(\mathbf{x})$	n	SIH	UFO	INTGLOB	ASA	GLOBAL	GOT	GAS
pf5	15	1336	1300	12200	N.R.	13700	N.R.	N.R.
pf6	18	6256	7700	N.R.	N.R.	22300	N.R.	N.R.

4 Concluding Remarks

The developed algorithm takes advantage of the well-known and liked Nelder-Mead simplex minimization and of inductive search, the novel idea of searching from the simple (e.g., minimizing one dimensional correlate of the function) to the complex (e.g., minimizing full n-dimensional function). Inductive search requires that the object function can be considered not only as a function of n variables, but also as a function of n. We have proposed a simple method of how in the context of minimization this can be achieved, when the objective function does not have such a property.

Another feature of inductive search is the combination of global searches in one dimension and subsequent local multidimensional searches. We carried this idea in Phase 2 of our algorithm by employing global 2-dimensional searches and local n-dimensional searches. Phase 2 proved being necessary in some difficult cases.

While the current tests have shown that our algorithm is quite efficient, obviously more extensive testing, especially in scientific applications, should be performed. For difficult problems we anticipate that Phase 2 should be repeated several times, each time with different starting simplexes.

Acknowledgements. We thank Dr. K. Price for providing us with the test bed of the Second International Competition of Evolutionary Optimization and for helpful suggestions. We are also grateful to Mr. E. Janka for making available his M.Sc. thesis and for additional explanations. This work is supported by Mayo Foundation and in part by GM34847.

References

1. Nelder, J.A., Mead, R.: A Simplex Method for Function Minimization. Comput. J. **7** (1965) 308–313
2. Jurs, P.C.: Computer Software Applications in Chemistry. 2nd edn. John Wiley, New York (1996)
3. Walters, F.H., Parker, L.R.Jr., Morgan, S.L., Deming, S.N.: Sequential Simplex Optimization. CRC Press, Boca Raton (1991)
4. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes. 2nd. edn. Cambridge University Press, New York (1992)
5. Huzak, M., Bajzer, Ž.: A New Algorithm for Global Minimization Based on the Combination of a Adaptive Random Search and Simplex Algorithm of Nelder and Mead. Croat. Chem. Acta **69** (1996) 775–791
6. Yen, J., Lee, B.: A Simplex Genetic Algorithm Hybrid. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation. IEEE Inc., Piscataway, NJ (1997) 175–180
7. Bilchev, G., Parmee, I.: Inductive Search. In: Proceedings of 1996 IEEE International Conference on Evolutionary Computation. IEEE Inc., Piscataway, NJ (1996) 832-836

8. Price, K.V.: Differential Evolution vs. the Functions of the 2nd ICEO. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation. IEEE Inc., Piscataway, NJ (1997) 153–157
9. Janka, E.: Vergleich Stochastischer Verfahren zur Globalen Optimierung. M.Sc. thesis, Vienna (1999); `http://www.solon.mat.univie.ac.at/~vpk/`; `janka@utanet.at`
10. Ingber, L.: Simulated Annealing: Practice Versus Theory. Math. Comput. Model. **18** (1993) 29–57
11. Ingber, L.: `http://www.ingber.com`
12. Ingber, L., Rosen, B.: Genetic Algorithms and Very Fast Simulated Re-annealing: A Comparison. Math. Comput. Model. **16** (1992) 87-100
13. Boender, C, Romeijn, H.: Stochastic Methods. In: Horst, R., Pardalos, P. (eds): Handbook of Global Optimization. Kluwer, Dordrecht (1995) 829–869
14. Csendes, T.: `http://www.inf.u-szeged.hu/~csendes/`
15. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd ed. Springer, Berlin (1966)
16. Michalewicz, Z.: `http://www.coe.uncc.edu/~zbyszek/`
17. Mongeau, M., Karsenty, H., Rouzé, V., Huriart-Urruty, J.-B.: Comparison of Public-domain Software for Black Box Global Optimization. Optim. Meth. and Software **13** (2000) 203–226