©*2002, Civil-Comp Ltd., Stirling, Scotland*
Proceedings of the Third International Conference on
Engineering Computational Technology,
B.H.V. Topping and Z. Bittnar (Editors),
Civil-Comp Press, Stirling, Scotland.

Paper 65

Globalized Nelder-Mead Method for Engineering Optimization

M.A. Luersen† and R. Le Riche‡
†Mechanical Laboratory, INSA de Rouen, France and
Mechanical Department, CEFET-PR, Brazil
‡CNRS URA 1884 / SMS
Ecole des Mines de Saint Etienne, France

Abstract

One of the fundamental difficulties in engineering design is the multiplicity of local solutions. This has triggered great efforts to develop global search algorithms. Globality, however, often has a prohibitively high numerical cost for real problems. A fixed cost local search, which sequentially becomes global is developed. Globalization is achieved by probabilistic restart. A spacial probability of starting a local search is built based on past searches. An improved Nelder-Mead algorithm makes the local optimizer. It accounts for variable bounds. It is additionally made more robust by reinitializing degenerated simplexes. The resulting method, called Globalized Bounded Nelder-Mead (GBNM) algorithm, is particularly adapted to tackle multimodal, discontinuous optimization problems, for which it is uncertain that a global optimization can be afforded. Different strategies for restarting the local search are discussed. Numerical experiments are given on analytical test functions and composite laminate design problems. The GBNM method compares favorably to an evolutionary algorithm, both in terms of numerical cost and accuracy.

Keywords: global optimization, Nelder-Mead, composite laminate design.

1 Introduction

Complex engineering optimization problems are characterized by calculation intensive system simulations, difficulties in estimating sensitivities (when they exist), the existence of design constraints, and a multiplicity of local solutions.

Acknowledging the last point, much research has been devoted to global optimization (e.g., [1, 2]). The high numerical cost of global optimizers has been at the origin of subsequent efforts to speed up the search either by adding problem specific knowledge to the search, or by mixing efficient, local algorithms with global algorithms.

There are many ways in which local and global searches can cooperate.

The simplest strategy is to link the searches in series, meaning that, firstly, a global optimization of limited cost is executed, the solution of which is refined by a local search. An example of the serial hybrid is given in [3] where simulated annealing, the global optimizer, is coupled with a sequential quadratic programming and a Nelder-Mead algorithm.

A large number of parallel local-global searches have been proposed [1, 4, 5] and analyzed [6, 7]. In these cases, iterations of global and local algorithms are intertwined. One can further classify parallel hybrids into those where the local searches converge, and those where local searches may be prematurely stopped. Memetic genetic algorithms [8] and multistart methods (e.g., deterministic restart in [9], random restarts in [10]) are examples of the former. The latter are usually based on clustering steps, where local searches approaching already explored regions of the design space are abandoned [6, 11].

When considering a real engineering optimization problem, a common situation is that the affordable total number of analyses is limited, that the presence of spurious local minima is unknown, and that it is uncertain if it will be possible to complete as few as two local searches. Nevertheless, achieving global results remains an objective of the optimizer. This typically occurs when dealing with an unknown function of less than 20 variables, for which one is willing to wait for about 1000 evaluations of the objective function. In such a case, a local-global method based on restart is the safest strategy because it can terminate in a short time (the length of a single local search). The method described in this article, the Globalized Bounded Nelder-Mead algorithm, GBNM, is meant to be a black-box local-global approach to real optimization problems. A restart procedure that uses an adaptive probability density keeps a memory of past local searches. Limits on variables are taken into account through projection. Finally, GBNM can be applied to discontinuous (no gradient information needed), nonconvex functions, since the local searches are based on a variant of the Nelder-Mead algorithm [12]. Improvements to the Nelder-Mead algorithm consist of simplex degeneracy detection and handling through reinitialization.

This paper is structured as follows. The GBNM algorithm is described in Section 2, and Section 3 reports numerical experiments on analytical functions and composite laminated plate design problems. In particular, different strategies for restarting the improved Nelder-Mead search are numerically discussed. The GBNM algorithm is also compared to a steady-state evolutionary algorithm [2].

2 Globalization of a Local Search by Probabilistic Restart

Local optimizers can make up a global search when repeatedly started from different points. The simplest restart methods initialize the search either from a regular grid of points, or from randomly chosen points. In the first case, one needs to know how many restarts will be performed to calculate the size of the mesh. In the other case, knowledge of past searches is not used, so that the same local optima may be found several times, costing vast unnecessary effort. In the current work, the number of restarts is unknown beforehand because a maximum number of analyses is imposed and the cost of each local search is unknown. A grid method cannot be applied here. Also, a memory of previous local searches is kept by building a spacial probability density of starting a search.

2.1 Probabilistic Restart

The probability, $p(x)$, of having sampled a point x is described here by a Gaussian Parzen-windows approach [13]. This method can be considered as a smoothed version of the histograms techniques, the histograms being centered at selected sampled points. $p(x)$ is written,

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} p_i(x) \; , \tag{1}$$

where N is the number of points already sampled, and p_i is the normal multi-dimensional probability density function,

$$p_i(x) = \frac{1}{(2\pi)^{\frac{n}{2}}(\det(\Sigma))^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-x_i)^T\Sigma^{-1}(x-x_i)\right) \; , \tag{2}$$

n is the dimension (number of variables) and Σ the covariance matrix,

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_n^2 \end{bmatrix} \; . \tag{3}$$

The variances, σ_j^2, are estimated by the relation,

$$\sigma_j^2 = \alpha(x_j^{max} - x_j^{min}), \tag{4}$$

where α is a positive parameter that controls the length of the Gaussians, and x_j^{max} and x_j^{min} are the bounds in the jth direction. Note that, in order to keep the method as simple and cost effective as possible, the variances are kept constant. This strategy would have a cost in terms of total number of analyses. The probability density is such that $\int_{-\infty}^{\infty} p(x)\,dx = 1$, but since a bounded domain Ω is considered, a bounded probability $\tilde{p}(x)$ is introduced,

$$\tilde{p}(x) = \frac{p(x)}{M} \; , \qquad M = \int_{\Omega} p(x)\,dx \; , \tag{5}$$

so that $\int_{\Omega} \tilde{p}(x)\,dx = 1$.

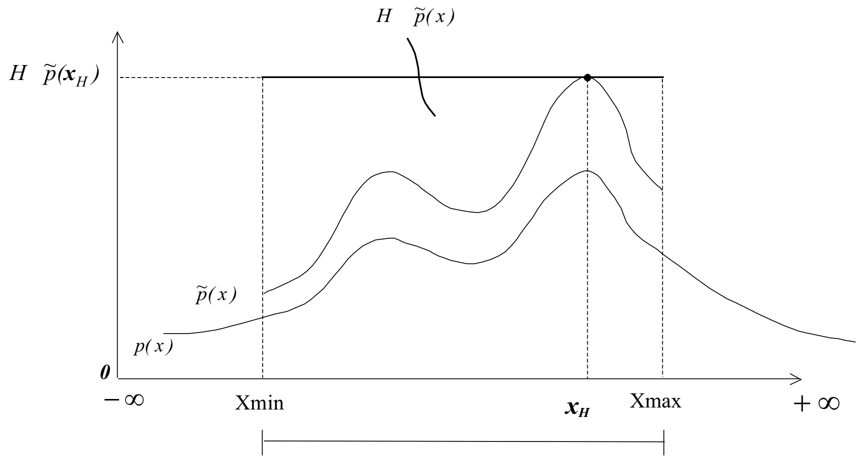


Figure 1: Probability density functions.

The probability density of sampling a new point, $\phi(x)$, is a probability density of not having sampled x before. For its estimation we adopt the following assumption: only the highest point x_H of $\tilde{p}(x)$ has a null probability of being sampled at the next iteration. So, the probability $\phi(x)$ is calculated as,

$$\phi(x) = \frac{H - \tilde{p}(x)}{\int_\Omega (H - \tilde{p}(x))\, dx} \;, \quad H = \max_{x \in \Omega} \tilde{p}(x) \;. \tag{6}$$

Fig. 1 illustrates $p(x)$, $\tilde{p}(x)$ and $H - \tilde{p}(x)$, in a unidimensional domain.

The maximization of ϕ is not performed exactly, firstly because of its numerical cost, and secondly, as will be seen in Section 3.1, because it would be detrimental to the search. Instead, N_r points are chosen randomly and the point that maximizes ϕ is selected to initiate the next search. Note that, in order to maximize ϕ, it is necessary to calculate neither M (5) nor H (6): the maximum of ϕ is the minimum of p, so p only is calculated.

Three parameters influence the probability density p and, consequently, the starting points: the points that are kept for the probability calculation, p; the number of random points used to maximize ϕ, N_r; and the Gaussians length parameter, α. Their setting is discussed in the numerical results (Section 3.1).

The probabilistic restart procedure can be applied to any local optimizer. In this case, an improved Nelder-Mead algorithm is proposed.

2.2 An Improved Nelder-Mead Search

The original Nelder-Mead algorithm [12] and the strategy for bounding variables are summarized in Appendix A. The GBNM algorithm differs from the Nelder-Mead method because of a set of restart options. The purpose of the restarts is twofold.

Firstly, probabilistic restarts based on the density p (Equation (1)) aim at repeating local searches until a fixed total cost, $C_{\max}$ has been reached. The probability of having located a global optimum increases with the number of probabilistic restarts. This is the "globalized" aspect of the method. In the current implementation of probabilistic restart, the size of the new simplex, a (defined in Equation (13)), is a uniform random variable taken between 2 and 10% of the smallest domain dimension.

Secondly, restarts are used to check and improve convergence of the algorithm. The two restart schemes that are convergence related initialize a new simplex from the current best vertex. The small and large test restarts use a small and large simplex of sizes a_s and a_l, respectively (see Equation (14)).

Convergence of the local Nelder-Mead searches is estimated through 3 criteria, the small, flat or degenerate simplex tests. The simplex is small if

$$\max_{k=1,\cdots,n+1} \left(\sum_{k=0}^{n} \left|\frac{x_i^k}{x_i^{max} - x_i^{min}}\right|\right) < \varepsilon_{s1} \ , \tag{7}$$

where x_i^k is the i-th component of the k-th edge, x_j^{max} and x_j^{max} are the bounds in the i-th direction, and ε_{s1} is a termination tolerance. The simplex is flat if

$$|f_H - f_L| < \varepsilon_{s2} \ , \tag{8}$$

where f_H and f_L are the highest and lowest objective functions in the simplex, and ε_{s2} is a tolerance value. The simplex is degenerated if it has collapsed into a subspace of the search domain. This is the most common symptom of a failed Nelder-Mead search [14] because the method cannot escape the subspace. More precisely, a simplex is called degenerated here if it is neither small, nor touches a variable bound, and one of the two following conditions is satisfied,

$$\frac{\min_{k=1,n} ||\mathbf{e}^k||}{\max_{k=1,n} ||\mathbf{e}^k||} < \varepsilon_{s3} \quad \text{or} \quad \frac{det[\mathbf{e}]}{\prod_k ||\mathbf{e}^k||} < \varepsilon_{s4} \ , \tag{9}$$

where $\mathbf{e}^k$ is the k-th edge, $\mathbf{e}$ is the edge matrix, and ε_{s3} and ε_{s4} are small positive constants.

The linking of the three restarts and three convergence tests in the GBNM algorithm is shown in Fig. 2. A memory of past convergence locations is kept, thus preventing unnecessarily spending computations on already analyzed points (third test, *T3*, in the flow chart of Fig. 2). When the simplex is flat, a probabilistic restart is performed (*T4*). A simplex which is degenerated induces a large test iteration (*T8*). When the optimality of the convergence point is unsure, such as a convergence on a variable bound where the simplex has degenerated (*T6*), a small test, that stands for an optimality check, is performed. If the small simplex returns to the same convergence point, it is considered to be a local optimum. It should be remembered that the Kuhn and Tucker conditions of mathematical programming are not applicable to the present non-differentiable framework. The tolerances for small and degenerated simplices, ε_{s1}

and [ε_{s3}, ε_{s4}], respectively, may be difficult to tune, so that a simplex which is becoming small may be tagged as degenerated before. Thus, if a degeneration is detected twice consecutively at the same point, the point is taken as a possible optimum, and a probabilistic restart is called. Similarly, if a degeneration is detected after a small test, this point is also saved as a possible optimum, and a large test is ordered.

Once the GBNM algorithm terminates, the list of possible local (eventually global) optima makes the results of the search. In practice, the calculation of many local or global optima is a benefit of the method in comparison with global optimizers that provide a single solution (e.g., evolutionary algorithms).

3 Numerical Results

In Paragraph 3.1, the choice of GBNM parameters is discussed. Results on an analytical function are given in Paragraph 3.2 and composite laminate design problems are addressed in Paragraphs 3.3 and 3.4. The GBNM method is compared to an evolutionary algorithm (EA). The evolutionary algorithm [2] has a steady-state structure [16] with real encoding, continuous crossover, and Gaussian mutation of variance $\sigma_i^{\text{mut}} = (x_i^{\max} - x_i^{\min})^2/16$. For fair comparisons, the parameters of the EA chosen for each test are the ones that perform best in 100 independent trials among all combinations of population sizes (20 or 50), mutation probabilities (0.15 or 0.20) and crossover probabilities (0.4 or 0.5).

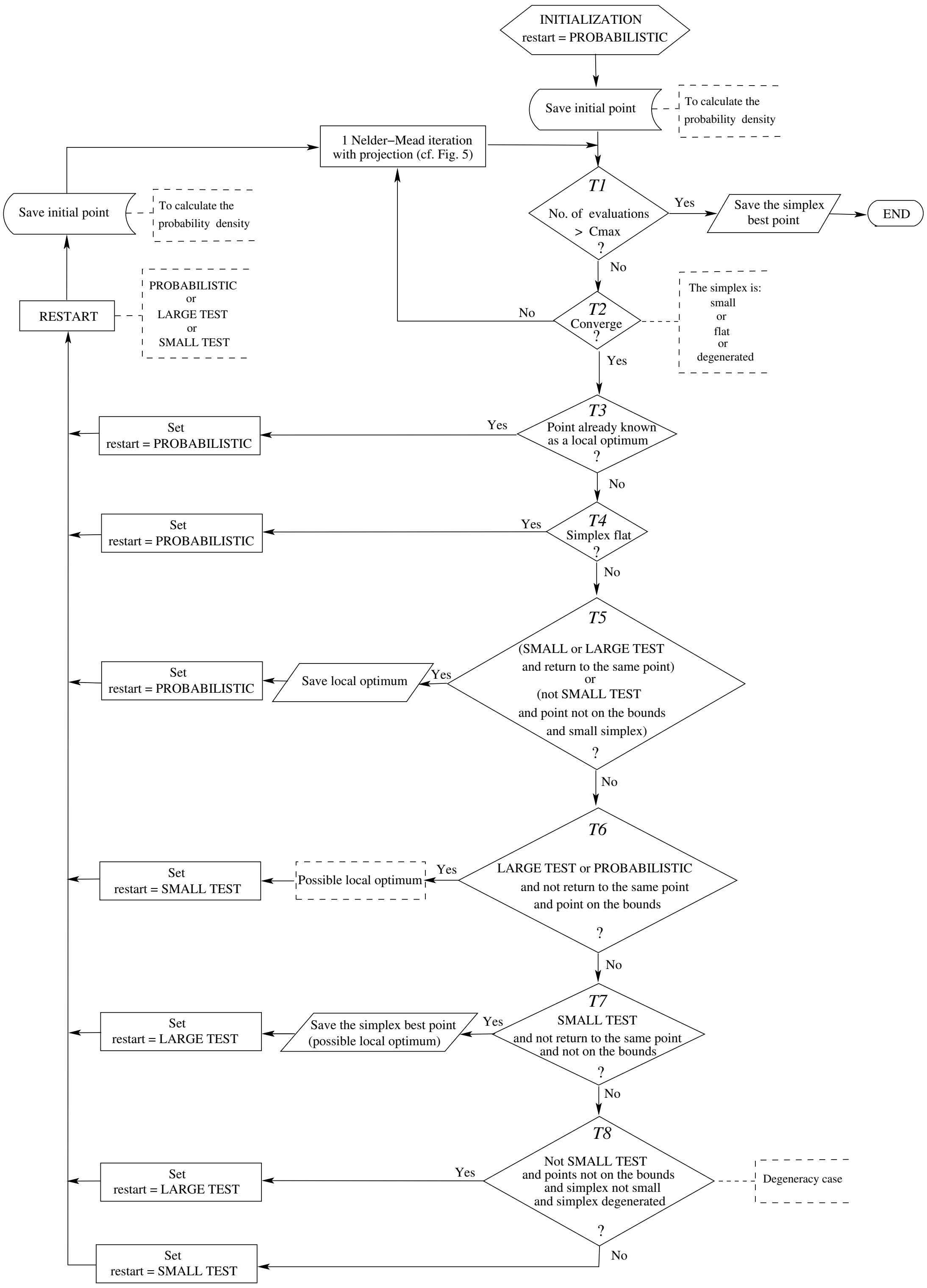
3.1 GBNM Parameters Choice

The Gaussians length parameter, α

In this work, α is set to 0.01, which means that one standard deviation away from the Gaussian mean point covers about 20% of the domain.

Points kept for the probability calculation

Three strategies have been compared in terms of probability of not finding at least one local minimum, P_{nfm}, on three multimodal functions, varying N_r from 1 to 1000. The x_i's used in Equations (1) and (2) are the starting points, or the starting and local convergence points, or all the points sampled during the search. One should remember that local convergence points are never duplicated (test T3 on Fig. 2). The functions tested are,



INITIALIZATION
restart = PROBABILISTIC
Save initial point
To calculate the probability density
1 Nelder−Mead iteration with projection (cf. Fig. 5)
T1
No. of evaluations > Cmax ?
Yes
No
Save the simplex best point
END
Save initial point
To calculate the probability density
RESTART
PROBABILISTIC or LARGE TEST or SMALL TEST
T2
Converge ?
The simplex is: small or flat or degenerated
T3
Point already known as a local optimum ?
Set restart = PROBABILISTIC
T4
Simplex flat ?
Set restart = PROBABILISTIC
T5
(SMALL or LARGE TEST and return to the same point) or (not SMALL TEST and point not on the bounds and small simplex) ?
Save local optimum
Set restart = PROBABILISTIC
T6
LARGE TEST or PROBABILISTIC and not return to the same point and point on the bounds ?
Possible local optimum
Set restart = SMALL TEST
T7
SMALL TEST and not return to the same point and not on the bounds ?
Save the simplex best point (possible local optimum)
Set restart = LARGE TEST
T8
Not SMALL TEST and points not on the bounds and simplex not small and simplex degenerated ?
Degeneracy case
Set restart = LARGE TEST
Set restart = SMALL TEST

Figure 2: Restarts and convergence tests linking in GBNM.

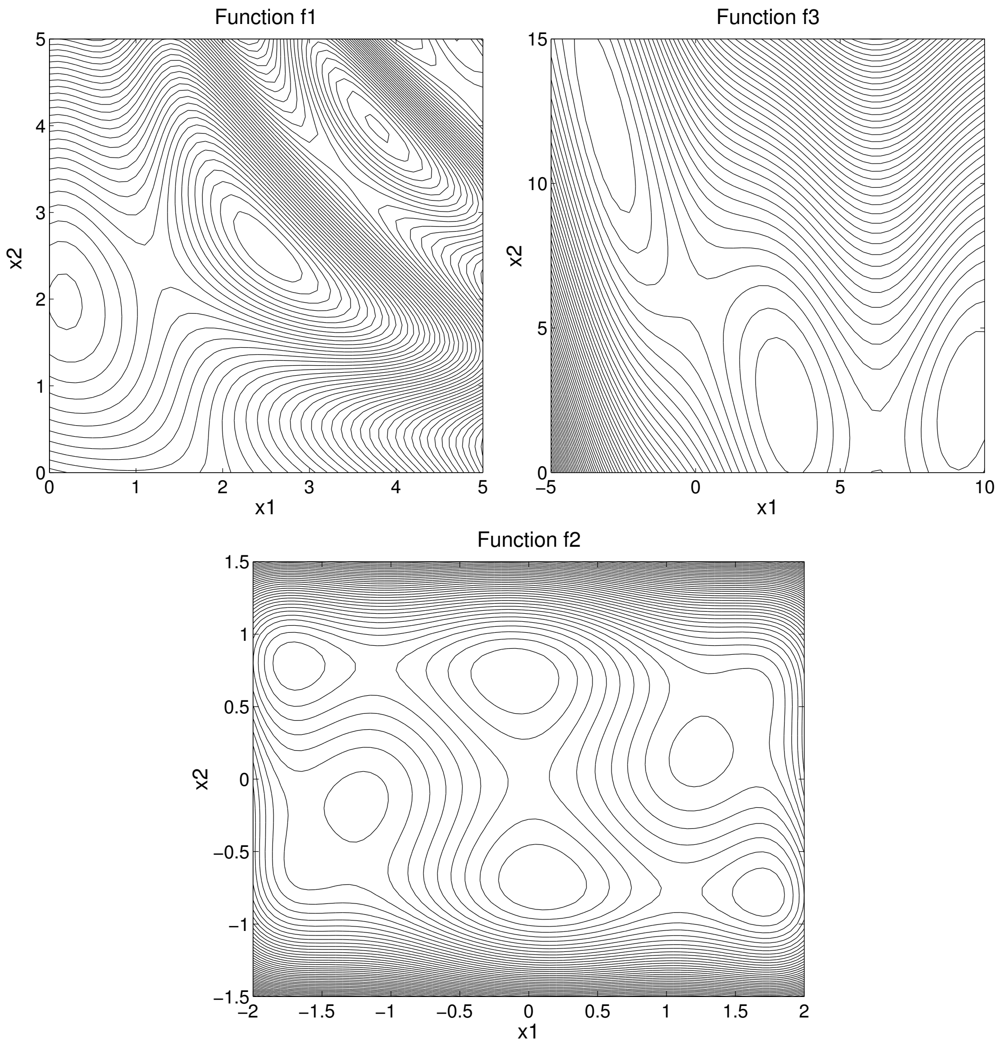


Figure 3: Test functions: contour plots.

$$
\begin{cases}
f_1(x_1,x_2) = & 2 + 0.01(x_2 - x_1^2)^2 + (1-x_1)^2 + 2(2-x_2)^2 \\
& + \sin(0.5x_1)\sin(0.7x_2x_1) \qquad x_1, x_2 \in [0,5] \\
f_2(x_1,x_2) = & (4 - 2.1x_1^2 + \frac{1}{3}x_1^4)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 \quad x_1, x_2 \in [-3,3] \\
f_3(x_1,x_2) = & (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10 \\
& x_1 \in [-5,10],\ x_2 \in [0,15]
\end{cases}
\tag{10}
$$

f_1 has 4 local minima, f_2 6, and f_3 3 (see Fig. 3). f_2 is known as the Six Humps Camel Back function and f_3 as the Branin's rcos function. The last option is memory and time consuming with degraded performance. For the first and the second strate-

N_r	f_1	f_2	f_3
1 (random restart)	0.96904	0.99679	0.49420
2	0.93201	0.99597	0.33697
3	0.92247	**0.99549**	0.27986
10	0.87097	0.99752	0.15167
20	0.82177	0.99876	0.14210
50	**0.77436**	0.99960	0.11170
1000	0.79485	0.99974	**0.08178**

Table 1: Probability P_{nfm} of not finding at least one of the local minima (C_{max}=500 analyses, 1000 runs, only the starting points are kept for the probability density calculation)

N_r	f_1	f_2	f_3
1 (random restart)	0.96904	0.99679	0.49420
2	0.93323	**0.99444**	0.29337
3	0.91236	0.99860	0.23685
10	0.84521	0.99862	0.09115
20	0.79950	0.99945	0.07130
50	0.74687	0.99971	0.04735
1000	**0.71452**	0.99982	**0.02681**

Table 2: Probability P_{nfm} of not finding at least one of the local minima (C_{max}=500 analyses, 1000 runs, starting and convergence points are kept for the probability density calculation)

gies, results after 500 analysis and based on 1000 independents runs are presented in Table 1 and Table 2, respectively. The second strategy performs best, independently of N_r. It shows that the starting and local convergence points efficiently summarize the topology of the basins of attraction. This scheme is chosen to update p.

Number of random points, N_r

If N_r is equal to 1, the reinitialization is random. If N_r is large, the initial points are distributed based on the initial and convergence points of past searches, which induces a regular grid-like pattern. Setting N_r to a small number, larger than 1, gives a biased-random reinitialization. It should be seen as a compromise between the grid and the random strategies. Optimum value of N_r depends on the test function: if the basins of attraction are regularly distributed, restarts following a regular pattern (i.e., N_r large) are optimal, and vice versa. From the tests results on the three multimodal functions presented in Table 2, the optimal strategy is N_r large for f_1 and f_3, while it is $N_r = 2$ for f_2. $N_r = 10$ is chosen as a compromise for general function optimization.

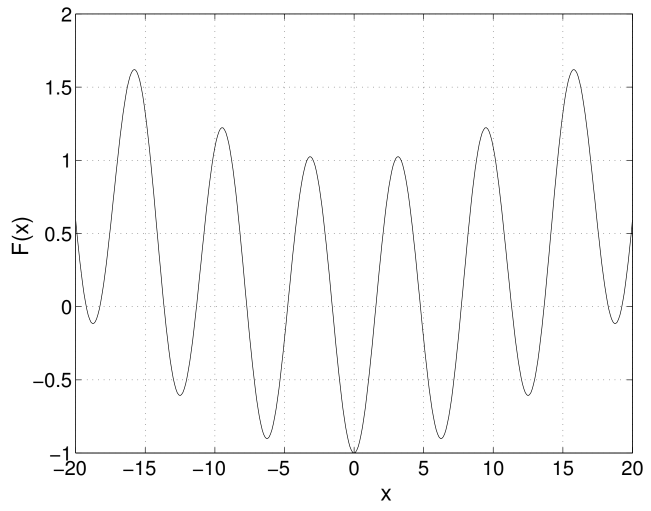


Figure 4: Griewank's test function (n=1).

3.2 Griewank's Function Minimization

Consider the minimization of the Griewank's test function,

$$F(x_1, \cdots, x_n) = \frac{1}{400n} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) \qquad x_i \in [-1000, 1000] \ , \tag{11}$$

where the dimension is n=12 and the global minimum is -1.0 at x_i=0.0, $i = 1, n$. This function has many local minima. Figure 4 shows it in the one-dimensional case (n=1), $x \in [-20, 20]$. Table 3 compares GBNM and the best point in the population of an evolutionary algorithm (EA) at 200, 1000, 5000 and 10000 function calls. Table 3 averages 100 independent runs where the starting point of the GBNM is randomly selected. The following criterion is used to consider the EA has congerved to the global minimum:

$$\frac{1}{n} ||\hat{x} - x^*|| < 1 \ , \tag{12}$$

where $\hat{x}$ is the best point found, and x^* the global minimum.

The main observation is that the GBNM method finds, on average, better objective function values, with a higher probability of finding the global minimum than the EA does. The advantage of the GBNM method is substantial at a low number of analyses, and slowly decreases as numerical cost grows.

3.3 Buckling Load Maximization

Composite laminates are made of stacked layers where each layer has oriented fibers melted in an isotropic matrix (see sketch in Fig. 5). The design problems adressed here aim at finding the optimal orientation of the fibers within each layer i, θ_i, where

	200 analyses		1,000 analyses		5,000 analyses		10,000 analyses	
	Minimum function value	Prob. of finding the global minimum	Minimum function value	Prob. of finding the global minimum	Minimum function value	Prob. of finding the global minimum	Minimum function value	Prob. of finding the global minimum
GBNM	19.321 $\pm$26.709	0/100	-0.526 $\pm$0.499	0/100	-0.947 $\pm$0.074	15/100	-0.982 $\pm$0.024	30/100
EA	36.497 $\pm$15.537	0/100	4.861 $\pm$1.920	0/100	0.090 $\pm$0.096	2/100	-0.157 $\pm$0.221	29/100

Table 3: Comparison of GBNM (N_r=10) and the best of an EA population on Griewank's function, 100 runs (average$\pm$standard deviation)

θ_i is a continuous variable bounded by $0°$ and $90°$. The plates are analyzed using the classical lamination theory and an elastic linear buckling model (see [19]).

Consider a simply supported carbon-epoxy square plate, subjected to in-plane compressive loads $N_x = N_y$, as shown in Fig. 5. The plate is balanced and symmetric and has 32 layers, each of which are 0.125mm thick. The elastic material properties of the layers are $E_1 = 115\ GPa$, $E_2 = 5\ GPa$, $G_{12} = 5\ GPa$, $\nu_{12} = 0.35$. The laminate is designed to maximize its buckling load. Since the plate is balanced and symmetric, there are 8 continuous design variables, the ply orientations, which are bounded between 0 and 90^o. This problem has a unique minimum, all plies oriented at $45°$. Note also that the outermost plies have more influence on the buckling behavior than the innermost plies.

Table 4 compares GBNM and the evolutionary algorithm (EA), showing the stacking sequences found, after 300, 500 and 1000 analyses based on 100 independents runs. At 300 evaluations, the buckling load of the designs proposed by GBNM and EA are equivalent. One notices that the first local search of GBNM has not always converged at that point. A steady difference between GBNM and EA designs at 300 analyses can be seen: GBNM, which is by nature a more oriented search, converges faster on the more sensitive outerplies than EA does, and vice versa on the innerplies. From 500 evaluations on, GBNM converges more accurately to the optimum than EA does.

3.4 Composite Plate Longitudinal Stiffness Maximization

A 16-ply balanced and symmetric plate, made of glass-epoxy, is to be designed by maximizing the longitudinal stiffness E_x. The elastic proprieties for the glass-epoxy layers are $E_1 = 45\ GPa$, $E_2 = 10\ GPa$, $G_{12} = 4.5\ GPa$ and $\nu_{12} = 0.31$. The plate is balanced and symmetric, so there are 4 fiber orientations to be found. This problem presents 16 local minima which are all the combinations of the $0°$ and $90°$ orientations. The global maximum has all plies oriented at $0°$. This example shows that local solutions exist for simple composite laminate design problems.

Table 5 gives the average number of local optima found after 2000 analyses based on 100 GBNM runs as a function of N_r. The first starting point is chosen randomly.

(stacking sequence: average± standard deviation)

300 analyses								
GBNM	[±45.04	±44.97	±45.02	±45.38	±45.38	±44.97	±43.28	$\pm 49.65]_s$
std. dev.	±0.47	±0.54	±0.83	±4.61	±4.37	±11.53	±17.46	±23.47
EA	[±45.09	±44.91	±45.23	±44.55	±44.78	±45.02	±45.16	$\pm 44.85]_s$
std. dev.	±1.75	±1.96	±2.67	±3.00	±3.66	±5.19	±8.80	±15.69
500 analyses								
GBNM	[±45.01	±45.03	±45.00	±45.08	±45.04	±44.97	±45.05	$\pm 45.18]_s$
std. dev.	±0.17	±0.22	±0.39	±0.40	±0.29	±0.46	±0.92	±4.22
EA	[±45.13	±44.95	±44.99	±44.95	±44.80	±45.00	±44.70	$\pm 46.45]_s$
std. dev.	±1.09	±1.28	±1.61	±1.90	±2.22	±3.22	±4.49	±11.30
1000 analyses								
GBNM	[±45.00	±45.00	±45.00	±45.00	±44.99	±45.00	±44.98	$\pm 45.02]_s$
std. dev.	±0.02	±0.02	±0.03	±0.05	±0.04	±0.06	±0.15	±0.44
EA	[±44.96	±44.98	±44.96	±45.07	±44.99	±44.92	±45.13	$\pm 45.27]_s$
std. dev.	±0.60	±0.61	±0.71	±0.95	±1.12	±1.17	±1.71	±4.95

Table 4: Buckling load maximization, 100 runs, N_r=10

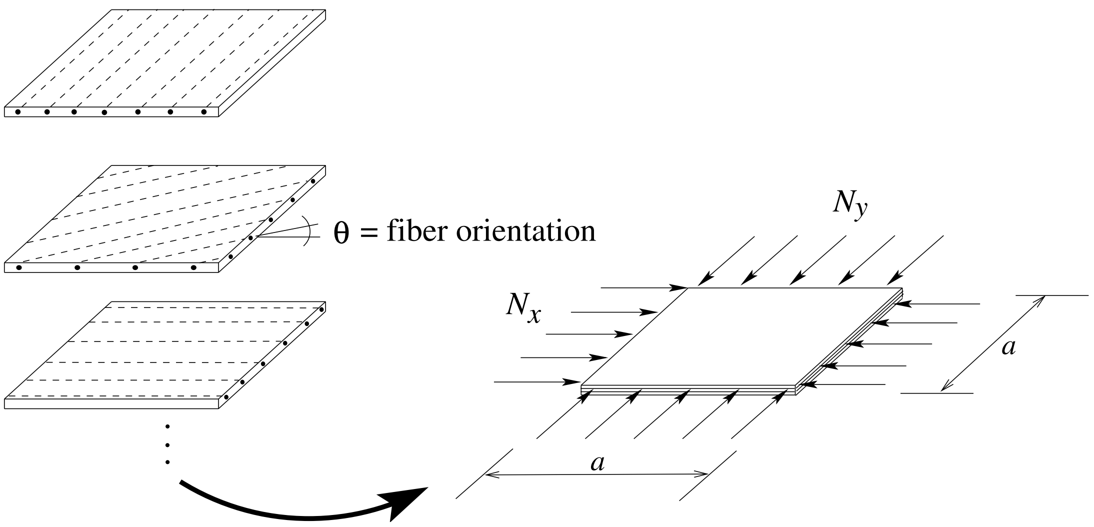


Figure 5: Simply supported rectangular plate subjected to in plane loads.

N_r	average number of optima found	standard-deviation
1	8.92	1.19
2	9.11	1.13
3	9.48	1.22
10	9.50	1.13
50	9.79	1.25
1000	10.03	1.41

Table 5: GBNM E_x maximization, 2000 analyses, 100 runs

One observes that the average number of optima found grows with N_r. This is expected since the optima are regularly distributed in the domain. Moreover, within a budget of 2000 function evaluations, the global optimum is always found because its basin of attraction is the largest one.

4 Concluding Remarks

A local/global optimization method based on probabilistic restart has been presented. Local searches are performed by an improved Nelder-Mead algorithm where design variables can be bounded and some search failure cases prevented. The method, called Globalized and Bounded Nelder-Mead search, does not need sensitivities and constructively uses computer resources up to a given limit. It yields a list of candidate local optima, which contain, with an increasing probability in terms of computer time, global solutions.

The GBNM method is simple in its principles, and the aforementionned features make it particularly useful in an engineering design context. It has been found to compare favorably to an evolutionary optimization algorithm in terms of convergence speed, accuracy of results and ability to find local optima.

5 Acknowledgements

The first author would like to express his thanks to the Federal Center for Technological Education of Paraná (CEFET-PR), Brazil, and to the Brazilian funding agency CNPq for the financial support during this research.

References

[1] A.A. Törn and A. Zilinskas, *“Global Optimization”*, Springer-Verlag, Berlin, 1989.

[2] T. Bäck, *"Evolutionary Algorithms in Theory and Practice"*, Oxford Univ. Press, 1996.

[3] Y. Shang, Y. Wan, M.P.J. Fromherz and L. Crawford, *"Toward Adaptive Cooperation between Global and Local Solvers for Continuous Constraint Problems"*, CP'01 Workshop on cooperative Solvers in Constraints Programming, (held in Pahos, Cyprus, Decembre (2001)).

[4] N. Durand and J.-M. Alliot, *"A Combined Nelder-Mead Simplex and Genetic Algorithm"*, available at http://www.recherche.enac.fr/opti/papers/, 1999.

[5] M. Okamoto, T. Nonaka, S. Ochiai and D. Tominaga, *"Nonlinear numerical optimization with use of a hybrid Genetic Algorithm incorporating the Modified Powell method"*, Applied Mathematics and Computation, 91, 63-72, 1998.

[6] A. A. Törn, *"A Search-Clustering Approach to Global Optimization"*, Towards Global Optimization 2, 49-62, 1978.

[7] D.E. Goldberg and S. Voessner, *"Optimizing Global-Local Search Hybrids"*, GECCO 99 - Genetic and Evolutionary Computation Conference, (held in Orlando, Florida, USA), 220–228, 1999.

[8] P. Moscato, *"On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms"*, Caltech Concurrent Computation Program, C3P Report 826, 1989.

[9] J. Barhen, V. Protopopescu and D. Reister, *"TRUST: A Deterministic Algorithm for Global Constrained Optimization"*, Science, 276, 1094-1097, 16 May 1997.

[10] X. Hu, R. Shonkwiller and M.C. Spruill, em "Random Restarts in Global Optimization", Technical Report, School of Mathematics, Georgia Institute of Technology, Atlanta, January 1994.

[11] F.J. Hickernell and Y.-X. Yuan, *"A Simple Multistart Algorithm for Global Optimization"*, OR Transactions, 1:2, 1997.

[12] J.A. Nelder and R. Mead, *"A simplex for function minimization"*, Computer J., 7, 308-313, 1965.

[13] O.R. Duda, P.E. Hart and D.G. Stork, *"Pattern Classification"*. 2nd Ed., John Wiley & Sons, New York, 2001.

[14] M.H. Wright, *"Direct Search Methods: One Scorned, Now Respectable"*, Dundee Biennal Conference in Numerical Analysis, Harlow, UK, 191-208, 1996.

[15] R.T. Rockafellar, *"Lagrange multipliers in optimization"*, Nonlinear Programming, Proc. SIAM-AMS, 9, 145-168, R.W. Cottle and C.E. Lemke, eds., New York, 1976.

[16] G. Syswerda, *"A Study of Reproduction in Generational and Steady State Genetic Algorithms"*, Foundations of Genetic Algorithms, G.J.E Rawlins, ed., San Mateo, CA: Morgan Kaufmann, 1991.

[17] M. Minoux, *"Mathematical Programming: Theory and Algorithms"*, John Wiley & Sons, 1986.

[18] R. Le Riche and F. Guyon, *"Dual Evolutionary Optimization"*, Artificial Evolution, Lecture Notes in Computer Science, No. 2310, selected papers of the 5th International Conference, Evolution Artificielle, EA 2001, P. Collet, E. Lutton,

M. Schoenauer, C. Fonlupt and J.-K. Hao, eds., Le Creusot, France, 281-294, 2001.

[19] J.-M. Berthelot, *"Composite Materials: Mechanical Behavior and Structural Analysis"*, Mechanical Engineering Series, Springer, 1999.

[20] R.T. Haftka and Z. Gürdal, *"Elements of Structural Optimization"*, 3rd rev. and expanded ed., Kluwer Academic Publishers, 1992.

A A Nelder-Mead Algorithm with Bounded Variables

The Nelder-Mead method [12] is the most popular direct search method for minimizing unconstrained real functions. It is based on the comparison of function values at the n+1 vertices x_i of a simplex. A simplex of size a is initialized at x_0 based on the rule (see [20]),

$$x_i \;=\; x_0 + pe_i + \sum_{\substack{k=1\\k\neq i}}^{n} qe_k \;,\; i \;=1,n \;, \tag{13}$$

where e_i are the unit base vectors and

$$\begin{aligned} p &= \frac{a}{n\sqrt{2}}\left(\sqrt{n+1}+n-1\right) \;, \\ q &= \frac{a}{n\sqrt{2}}\left(\sqrt{n+1}-1\right) \;. \end{aligned} \tag{14}$$

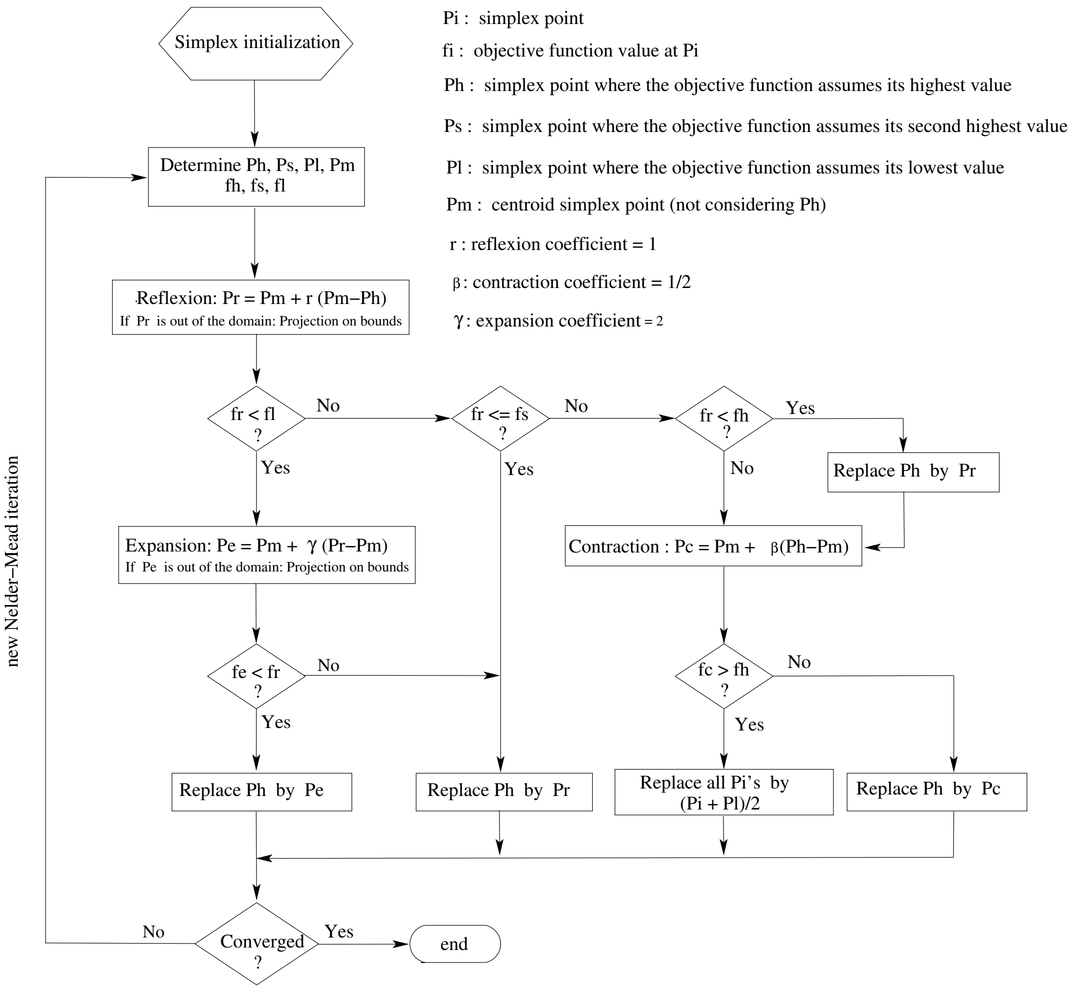
The simplex vertices are changed through *reflection*, *expansion* and *contraction* operations in order to find an improving point (see Fig. 6). The algorithm terminates when the vertices function values become similar, which is measured with the inequality,

$$\sqrt{\sum_{i=1}^{n+1}(f_i-\bar{f})^2/n} \;<\; \varepsilon \;,\; \bar{f} \;=\; \frac{1}{n+1}\sum_{i=1}^{n+1} f_i \;, \tag{15}$$

where ε is a small positive scalar. The cumulative effect of the operations on the simplex is, roughly speaking, to stretch the shape along the descent directions, and to zoom around local optima. Two comments on the properties of the algorithm are added. Firstly, the Nelder-Mead algorithm may fail to converge to a local optimum, which happens in particular when the simplex collapses into a subspace. Secondly, the method may escape a region that would be a basin of attraction for a pointwise descent search if the simplex is large enough. Ultimately, as the size of the simplex decreases, the algorithm becomes local.

The original Nelder-Mead algorithm was conceived for unbounded domain problems. With bounded variables, the points can leave the domain after either the reflection or the expansion operation. It is straightforward to account for variables bounds by projection,

$$\left\{ \begin{array}{l} \text{if } (x_i < x_i^{min}),\; x_i = x_i^{min} \;, \\ \text{if } (x_i > x_i^{max}),\; x_i = x_i^{max} \;. \end{array} \right. \tag{16}$$



Pi : simplex point
fi : objective function value at Pi
Ph : simplex point where the objective function assumes its highest value
Ps : simplex point where the objective function assumes its second highest value
Pl : simplex point where the objective function assumes its lowest value
Pm : centroid simplex point (not considering Ph)
r : reflexion coefficient = 1
β : contraction coefficient = 1/2
γ : expansion coefficient = 2
Simplex initialization
Determine Ph, Ps, Pl, Pm fh, fs, fl
Reflexion: Pr = Pm + r (Pm−Ph)
If Pr is out of the domain: Projection on bounds
fr < fl ?
fr <= fs ?
fr < fh ?
Yes
No
Replace Ph by Pr
Expansion: Pe = Pm + γ (Pr−Pm)
If Pe is out of the domain: Projection on bounds
Contraction : Pc = Pm + β(Ph−Pm)
fe < fr ?
fc > fh ?
Replace Ph by Pe
Replace Ph by Pr
Replace all Pi's by (Pi + Pl)/2
Replace Ph by Pc
Converged ?
end
new Nelder−Mead iteration

Figure 6: Nelder-Mead algorithm with bounded variables.

The flowchart of the Nelder-Mead method shown in Fig. 6 differs from the original method only in the initialization (Equation (13)) and in the bounded variables. An important side effect of accounting for the bounded variables through projection is that it tends to make the simplex collapse into the subspace of the saturated variables. A specific convergence test, based on a small simplex reinitialization at the point of convergence is then required (see “small test” in Section 2.2).