XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

A Differential Evolution and Nelder-Mead hybrid algorithm for constrained Engineering Design problems

Renan A. Chagas[1], Renan R. Sousa[1], Gustavo R. Souza[2], Michelli M. S. Loureiro[1], Felipe S. Loureiro[2]

[1]*Departamento de Ciência da Computação, Universidade Federal de São João del-Rei*
BR-494, UFSJ Campus CTan, 36301-360, São João del Rei, Minas Gerais, Brazil
[2]*Departamento de Ciências Térmicas e dos Fluidos, Universidade Federal de São João del-Rei*
Praça Frei Orlando, 36307-352, São João del-Rei, Minas Gerais, Brazil
{renan.a.c.1802, renanrsousa2018}@aluno.ufsj.edu.br, {souzagr, felipe.loureiro, michelli.loureiro}@ufsj.edu.br

Abstract. Engineering Design optimization problems frequently involve complex, nonlinear, and computationally expensive objective functions subject to various constraints. Differential Evolution (DE) is a population-based evolutionary algorithm well-suited for global optimization, whereas the Nelder-Mead (NM) simplex method offers efficient local search capabilities without requiring gradient information. This paper proposes a hybrid optimization strategy that integrates DE with the NM method to exploit their complementary strengths. The DE algorithm is initially employed to explore the global search space and identify promising regions by iteratively evolving a population of candidate solutions based on a predefined stopping criterion. Once a suitable region is identified, the NM algorithm is applied to perform local refinement, enhancing convergence toward the optimal solution. To handle constrained problems, the NM method is extended using penalty function approach, wherein constraint violations are incorporated into the objective function. The proposed hybrid algorithm is evaluated on a suite of constrained Engineering Design benchmarks using MATLAB. Results demonstrate superior performance compared to the standalone DE algorithm, yielding improved solution accuracy and a reduced number of objective function evaluations.

Keywords: Constrained Optimization, Hybrid Algorithm, Engineering Design, Differential Evolution, Nelder-Mead

1 Introduction

Engineering Design optimization problems are frequently characterized by complex, nonlinear, and multi modal objective functions, often subject to various equality and inequality constraints. These problems typically cannot be solved analytically due to the high dimensionality of the search space and the non-convex nature of the design domain. As a result, numerical optimization methods have become essential tools in modern Engineering practice.

In recent decades, metaheuristic algorithms have gained considerable attention due to their ability to address large-scale, nonlinear, and non-differentiable optimization problems. Among them, bio-inspired techniques that simulate natural phenomena, such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE), have shown significant promise. These algorithms excel in global search tasks by efficiently exploring the solution space and avoiding premature convergence to local optima.

Differential Evolution, in particular, has been widely adopted for its simplicity, robustness, and strong performance in solving real-world Engineering problems. Its population-based approach and differential mutation mechanism make it especially suitable for continuous optimization tasks. However, despite its global search capabilities, DE may exhibit slow convergence in the final optimization stages, particularly in problems that demand fine-grained solution accuracy.

To address this limitation, hybrid optimization strategies have been proposed, combining global metaheuristics with local search algorithms to improve both convergence speed and solution precision. One common local search technique used in such hybridizations is the Nelder-Mead simplex method, a derivative-free algorithm that operates by iteratively updating a simplex of points to approach a local minimum. Its low computational cost and ease of implementation make it attractive for local refinement phases.

Several hybrid approaches have been explored in the literature. Mihoub et al. [1] reviewed a range of hybrid

algorithms that integrate global and local search strategies, highlighting their advantages in solving Engineering Design problems. Zahara and Hu [2] proposed a hybrid NM-PSO method, combining the global exploration capabilities of particle swarm optimization with the local refinement efficiency of the Nelder-Mead method. Similarly, Chelouah and Siarry [3] introduced the Continuous Hybrid Algorithm (CHA), which employs a genetic algorithm for global search followed by a local search phase.

However, some of these hybrid frameworks present notable limitations. For example, CHA lacks a mechanism for constraint handling, and the NM-PSO approach proposed by Zahara requires gradient information for certain operations, which can restrict its applicability to problems with available and inexpensive derivatives.

In this context, the present work proposes a novel hybrid algorithm—referred to as DE-NM that combines the global search capabilities of Differential Evolution with the local optimization efficiency of the Nelder-Mead simplex method. The DE phase is responsible for exploring the feasible search domain and identifying promising regions through population evolution. Once a stopping criterion is met, the NM method is applied to refine the solution locally within the identified region. To handle constraints, the proposed algorithm incorporates Deb's penalty function strategy Deb [4], allowing the Nelder-Mead method to operate effectively in constrained optimization contexts. The resulting DE-NM algorithm is evaluated on a benchmark suite of Engineering Design problems, demonstrating improved performance in terms of convergence, solution quality, and computational efficiency when compared to standalone metaheuristics.

2 Nelder-Mead simplex with constraints

2.1 Nelder-Mead simplex

Optimization problems, as formulated by Ackleh et al. [5], constitute a class of mathematical problems where an objective function $\phi(x)$ must be minimized with respect to its decision variables, subject to equality constraints $c_i(x)$ and inequality constraints $g_i(x)$. Those problems take the formulation presented in eq. (1):

$$\begin{aligned} \text{minimize} \quad & \phi(x) \\ \text{subject to} \quad & c_i(x) = 0, i = 1, \ldots, m_1, \\ & g_i(x) \leq 0, i = 1, \ldots, m_2, \\ \text{where} \quad & \phi : D \to \mathbb{R} \text{ and } c_i, g_i : D \to \mathbb{R}, \\ & \text{and } x = (x_1, \ldots, x_n)^T. \end{aligned} \tag{1}$$

The Nelder-Mead simplex, originally developed by Nelder and Mead [6] in 1965, is a method "for the minimization of a function of n variables, which depends on the comparison of function values at the $(n + 1)$ vertices of a general simplex, followed by the replacement of the vertex with the highest value by another point [...], and contracts on to the final minimum". It performs four geometric transformations (reflection, expansion, contraction and shrinkage) and uses four parameters (ρ, χ, γ and σ, respectively) to achieve its goal. Common values for these parameters (and the ones used in our algorithm) are $\rho = 1$, $\chi = 2$, $\gamma = 0.5$ and $\sigma = 0.5$. These transformations are better illustrated in Fig. 1. The specific Nelder-Mead implementation used in this paper follows what was documented by Lagarias et al. [7].

Following the selection of a random initial vertex x_1, the initial simplex is established according to Ackleh et al. [5] as $x_{i+1} = x_1 + \lambda e_i, i = 1, 2, \ldots, n$, where $\lambda = \max\{||(x_1, \infty||, 1\}$ as specified by Higham [8], and e_i is the identity matrix, modifying the i^{th} term of x_1 by λ.

Upon establishing the initial simplex, a reflection point x_r is computed according to eq. (2):

$$x_r = (1 + \rho)\bar{x} - \rho x_{n+1}, \tag{2}$$

where $\bar{x}$ is the centroid of the n best vertices, excluding the last one. Subsequently, the algorithm evaluates the four possible geometric transformations to determine which one will be performed during the iteration. If this reflection point and its corresponding objective function value lies between the best point x_1 and the last point f_n, the reflection was successful, and x_r is accepted as part of the simplex. Should $f(x_r) < f(x_1)$, the algorithm evaluates a potential expansion the point x_e, defined as

$$x_e = (1 + \rho\chi)\bar{x} - \rho\chi x_{n+1}. \tag{3}$$

If $f(x_e) < f_r$, then x_e becomes part of the simplex; otherwise, the reflection point x_r stays in the simplex. In the case where x_r fails both cases ($f(x_r) \geq f(x_n)$), either a contraction or a shrink step will take place, with the shrink step occurring only if the contraction fails. A contraction can take two forms: it can be an outside contraction if x_r is strictly better than x_{n+1}, evaluated as

$$x_c = (1 + \rho\gamma)\bar{x} - \rho\gamma, \tag{4}$$

or else it can be an inside contraction, evaluated as

$$x_{cc} = (1 - \gamma)\bar{x} + \gamma x_{n+1}. \tag{5}$$

If $f(x_c) \leq f(x_r)$ or $f(x_{cc}) < f(x_{n+1})$, then a shrink step will be performed at the n points $v_i = x_1 + \gamma(x_i - x_1), i = 2, \ldots, n + 1$.

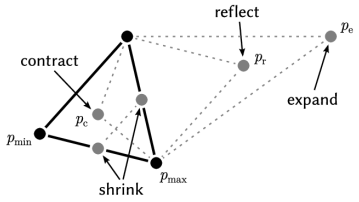


Figure 1. An iteration of the Nelder-Mead method over two-dimensional space. Courtesy of Jade Yu Cheng and Thomas Mailund, CC BY 4.0, via Wikimedia Commons.

2.2 Constraint handling

The original NM algorithm was developed for unconstrained optimization problems and does not incorporate any constraint-handling mechanisms. To address this limitation, Mehta and Dasgupta [9] successfully applied Deb's constraint-handling technique [4] to the Nelder-Mead method, which is also the strategy employed in this work.

In this approach, the vertices of the simplex are classified into three categories based on feasibility: infeasible region (all points violate some constraint), feasible region (no constraint violations), and boundary region, with feasible and infeasible points. The evaluation of each vertex depends on its classification: feasible vertices are assigned their actual objective function values; infeasible vertices receive the sum of their constraint violation as a penalty; and in the case of a boundary simplex, feasible vertices retain their objective values, while infeasible vertices are penalized with the sum of their constraint violations plus the objective function value of the worst feasible vertex within the simplex. This ensures that feasible solutions are prioritized and guides the simplex towards feasible regions of the search space.

As described by Lagarias et al. [7], the Nelder-Mead method is particularly appealing for Engineering optimization problems involving expensive function evaluations due to its relatively low computational cost. Specifically, the algorithm requires one function evaluation during the reflection step, two evaluations during expansion or contraction steps, and $(n + 2)$ evaluations during the shrinkage step.

It is important to note that the algorithm implementation described in Subsection 2.1 incorporates constraint handling internally. Every time the objective function is evaluated, the algorithm verifies in which of the previously described categories the candidate point lies and also if it is within the lower and upper bounds of the problem; if not, then the objective function value will be infinite (as per Takenaga et al. [10]).

3 Hybrid Algorithm

The Differential Evolution (DE) algorithm, introduced by Storn and Price [11], is an effective stochastic global optimization method. Starting from an initial population of candidate solutions distributed across the search domain, DE generates new candidate solutions by applying **mutation** operators to perturb existing points and combines these through **crossover** to produce offspring. A **selection** mechanism then retains the best candidates for the next generation. While DE is robust for global exploration, its convergence typically requires a large number of function evaluations, which can be computationally expensive for Engineering problems involving costly objective

functions. On the other hand, the deterministic Nelder-Mead (NM) algorithm, described in Subsection 2.1, while demanding fewer function evaluations than DE, can suffer from slow convergence rates and sensitivity to initial conditions.

In this work, we propose a hybrid algorithm that combines the global search capabilities of DE with the local refinement efficiency of the NM simplex method, modified to handle constraints as described previously. This hybrid approach aims to leverage the global exploration strength of DE to locate promising regions in the search space, followed by the local convergence of NM to fine-tune the solution. The overall workflow of the proposed algorithm is illustrated in Fig. 2.

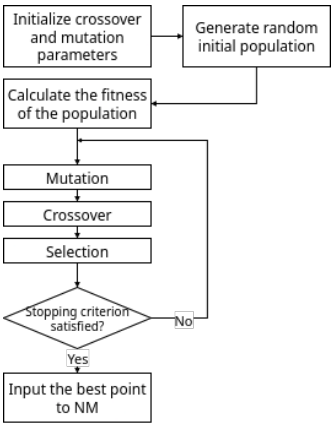


Figure 2. Diagram for the hybrid algorithm

The stopping criterion used to transition from the DE to the NM algorithm is based on the work of Chelouah and Siarry [3]. However, it should be noted that this criterion may be insufficient when applied to PlatEMO's [12] DE implementation, potentially causing premature termination before the NM phase begins. Therefore, it is recommended to empirically tune the number of generations allocated to DE based on the specific problem characteristics and available computational resources. Despite this, considering the high computational cost of function evaluations in Engineering problems, the proposed hybrid algorithm typically requires significantly fewer function evaluations than running a standalone DE algorithm until full convergence.

The proposed hybrid algorithm was implemented in MATLAB. For the DE component, we utilized the Differential Evolution implementation available in the PlatEMO platform [12], with a crossover rate of 0.9 and a mutation factor of 0.5. Algorithm performance was assessed on three Engineering Design benchmark problems: the welded beam design problem and Deb's test problem 1, both present in Deb [4]; and the tension-compression spring design problem from Zahara [2]. In the following tables, **DE*** refers to the DE population at the stopping point, just before switching to the NM algorithm; **DE-NM** denotes the proposed hybrid algorithm; **DE** represents the standalone DE algorithm executed with the same number of function evaluations as DE-NM.

3.1 Deb's Test Problem 1

This constrained optimization problem, proposed by Deb [4], represents a modified Himmelblau function [13] with two circular constraints:

$$\begin{aligned} \text{Minimize} \quad & f_1(\bar{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \\ \text{Subject to} \quad & g_1(\bar{x}) = 4.84 - (x_1 - 0.05)^2 - (x_2 - 2.5)^2 \geq 0 \\ & g_2(\bar{x}) = x_1^2 + (x_2 - 2.5)^2 - 4.84 \geq 0 \\ & 0 \leq x_1 \leq 6, \quad 0 \leq x_2 \leq 6 \end{aligned} \tag{6}$$

Table 1. Statistical results of the different methods for Deb's test problem 1, 20 iterations

Method	Avg. No. of FE	Best	Mean	Median	Worst	St. Dev.
DE	585.78	13.65184	14.63740	14.46408	20.35607	1.296210
DE*	400	13.60607	14.96578	14.14550	19.28800	1.778469
DE-NM	585.78	13.59084	13.59084	13.59084	13.59084	2.1575e-12

Table 2. Comparison of the best solution for Deb's test problem 1, 20 iterations

Method	$x(1)$	$x(2)$	$f(x)$
DE	2.241159	2.397733	13.65184
DE*	2.246023	2.375628	13.60606
DE-NM	2.246826	2.381635	13.59084
Exact	2.246826	2.381865	13.59085

The proposed DE-NM algorithm demonstrates exceptional performance, achieving a best objective function value of 13.59084 compared to 13.65184 for standalone DE and 13.60607 for DE*. The DE-NM algorithm also shows convergence consistency, with a standard deviation of 2.1575×10^{-12}, indicating virtually identical solutions across all 50 runs.

3.2 Welded beam problem

This structural optimization problem, as formulated by Deb [4], aims to minimize the cost of a welded beam subject to five constraints on shear stress $\tau(x)$, bar bending stress $\sigma(x)$, bar buckling load $P_c(x)$, and bar deflection $\delta(x)$:

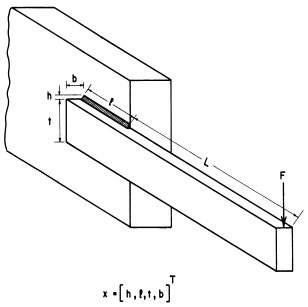


Figure 3. A welded beam.

$$
\begin{aligned}
\text{Minimize} \quad & f_w(\bar{x}) = 1.10471h^2l + 0.04811tb(14.0 + l) \\
\text{Subject to} \quad & g_1(\bar{x}) = 13,600 - \tau(\bar{x}) \geq 0 \\
& g_2(\bar{x}) = 30,000 - \sigma(\bar{x}) \geq 0 \\
& g_3(\bar{x}) = b - h \geq 0 \\
& g_4(\bar{x}) = P_c(\bar{x}) - 6,000 \geq 0 \\
& g_5(\bar{x}) = 0.25 - \delta(\bar{x}) \geq 0 \\
& 0.125 \leq h \leq 10, \quad 0.1 \leq l, t, b \leq 10,
\end{aligned}
\tag{7}
$$

where the four decision variables are $\bar{x} = (h, l, t, b)$ and:

$$
\begin{aligned}
\tau(\bar{x}) &= \sqrt{\frac{\tau'(\bar{x})^2 + \tau''(\bar{x})^2 + l \cdot \tau'(\bar{x}) \cdot \tau''(\bar{x})}{\sqrt{0.25(l^2 + (h + t)^2)}}}, \\
\sigma(\bar{x}) &= \frac{504,000}{t^2b}, \\
P_c(\bar{x}) &= 64746.022(1 - 0.0282346t)tb^3, \\
\delta(\bar{x}) &= \frac{2.1952}{t^3b}, \\
\tau'(\bar{x}) &= \frac{6,000}{\sqrt{2}hl}, \\
\tau''(\bar{x}) &= \frac{6,000(14 + 0.5l)\sqrt{0.25(l^2 + (h + t))^2}}{2\left\{0.707hl\left(\frac{l^2}{12+0.25(h+t)^2}\right)\right\}}.
\end{aligned}
\tag{8}
$$

Table 3. Statistical results of the different methods for the welded beam problem, 120 iterations

Method	Avg. No. of FE	Best	Mean	Median	Worst	St. Dev.
DE	5978.94	2.510929	2.809067	2.788051	3.335144	0.169025
DE*	4800	2.530691	2.800684	2.807971	3.248851	0.157753
DE-NM	5978.94	2.383272	2.491911	2.463295	2.783294	0.095467

Table 4. Comparison of the best solution for the welded beam problem, 120 iterations

Method	x(1)	x(2)	x(3)	x(4)	f(x)
DE	0.2238	6.0713	9.4138	0.2393	2.51093
DE*	0.2388	6.5863	8.8333	0.2418	2.53069
DE-NM	0.2435	6.2496	8.2915	0.2444	2.38327
Exact	0.2444	6.2187	8.2915	0.2444	2.38116

The DE-NM algorithm achieves a best result of 2.383272, with only 0.09% deviation from the exact solution 2.38116, outperforming DE (5.4% deviation) and DE* (6.3% deviation). The lower standard deviation also confirms greater convergence consistency.

3.3 Tension-compression spring problem

This problem with three decision variables, described by Zahara [2], seeks to minimize the weight of a spring subject to four mechanical and geometrical constraints:

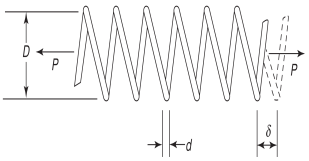


Figure 4. A tension-compression spring.

$$
\begin{aligned}
\text{Minimize} \quad & f(x) = (x_3 + 2)x_2x_1^2 \\
\text{Subject to} \quad & g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0 \\
& g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
& g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \\
& g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
& 0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15.
\end{aligned}
\tag{9}
$$

Table 5. Statistical results of the different methods for the tension-compression spring problem, 100 iterations

Method	Avg. No. of FE	Best	Mean	Median	Worst	St. Dev.
DE	3398.59	0.0126921	0.0133797	0.0132433	0.0162614	0.000538
DE*	3000	0.0127408	0.0133747	0.0133003	0.0149083	0.000488
DE-NM	3398.59	0.0126661	0.0129632	0.0127816	0.0141310	0.000330

Table 6. Comparison of the best solution for the tension-compression spring problem, 100 iterations

Method	$x(1)$	$x(2)$	$x(3)$	$f(x)$
DE	0.051641	0.355271	11.395976	0.0126921
DE*	0.052814	0.383667	9.905154	0.0127408
DE-NM	0.051466	0.351385	11.608615	0.0126661
Exact	0.051620	0.355498	11.333272	0.0126302

The DE-NM algorithm achieves a best objective value of 0.0126661, with only 0.28% deviation from the exact solution. This performance surpasses both DE (0.49% deviation) and DE* (0.87% deviation), while also showing the lowest standard deviation, confirming greater solution consistency.

4 Conclusions

The proposed DE-NM algorithm demonstrated consistent performance improvements across all tested Engineering Design problems when compared to conventional single-method approaches. It achieved superior solutions with fewer function evaluations, highlighting both efficiency and effectiveness. The statistical results confirm its robustness. Nonetheless, the current implementation requires manual parameter tuning, and further enhancements are needed – such as automating the DE-to-NM transition within PlatEMO and improving constraint-handling techniques – to increase usability and generalization across diverse problem scenarios.

Acknowledgements. The authors acknowledge the financial support from CAPES, PIBIC/CNPq and UFSJ/PPMEC/DCTEF.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] T. El-mihoub, A. Hopgood, L. Nolle, and B. Alan. Hybrid genetic algorithms: A review. *Engineering Letters*, vol. 3(2), 2006.
[2] E. Zahara and Y.-T. Kao. Hybrid nelder–mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, vol. 36, n. 2, pp. 3880–3886, 2009.
[3] R. Chelouah and P. Siarry. Genetic and nelder–mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. *European Journal of operational research*, vol. 148, n. 2, pp. 335–348, 2003.
[4] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, vol. 186, n. 2-4, pp. 311–338, 2000.
[5] A. Ackleh, E. Allen, R. Kearfott, and P. Seshaiyer. *Classical and Modern Numerical Analysis: Theory, Methods and Practice*. Chapman & Hall/CRC Numerical Analysis and Scientific Computing Series. CRC Press, 2009.
[6] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, vol. 7, n. 4, pp. 308–313, 1965.
[7] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, vol. 9, n. 1, pp. 112–147, 1998.
[8] N. J. Higham. Optimization by direct search in matrix computations. *SIAM Journal on Matrix Analysis and Applications*, vol. 14, n. 2, pp. 317–333, 1993.
[9] V. K. Mehta and B. Dasgupta. A constrained optimization algorithm based on the simplex search method. *Engineering Optimization*, vol. 44, n. 5, pp. 537–550, 2012.
[10] S. Takenaga, Y. Ozaki, and M. Onishi. Practical initialization of the nelder–mead method for computationally expensive optimization problems. *Optimization Letters*, vol. 17, n. 2, pp. 283–297, 2023.
[11] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, vol. 11, pp. 341–359, 1997.
[12] Y. Tian, R. Cheng, X. Zhang, and Y. Jin. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine*, vol. 12, n. 4, pp. 73–87, 2017.
[13] D. M. Himmelblau. *Applied nonlinear programming / [by] David M. Himmelblau.* McGraw-Hill, New York, 1972.