Engineering Optimization

ISSN: 0305-215X (Print) 1029-0273 (Online) Journal homepage: https://www.tandfonline.com/loi/geno20

Improved Nelder–Mead algorithm in high dimensions with adaptive parameters based on Chebyshev spacing points

V.K. Mehta

To cite this article: V.K. Mehta (2019): Improved Nelder–Mead algorithm in high dimensions with adaptive parameters based on Chebyshev spacing points, Engineering Optimization, DOI: 10.1080/0305215X.2019.1688315

To link to this article: https://doi.org/10.1080/0305215X.2019.1688315

Published online: 27 Nov 2019.

ENGINEERING OPTIMIZATION
https://doi.org/10.1080/0305215X.2019.1688315

Taylor & Francis
Taylor & Francis Group

Improved Nelder–Mead algorithm in high dimensions with adaptive parameters based on Chebyshev spacing points

V.K. Mehta

Department of Mechanical Engineering, Tezpur University, Tezpur, Assam, India

ABSTRACT

In spite of being one of the most popular optimization methods, Nelder–Mead's simplex search algorithm with the default choice of parameters performs poorly on high-dimensional problems. The work presented here concerns such values of the Nelder–Mead algorithm's parameters that help improve the convergence and success rate of the algorithm in high dimensions. In this work, a novel way of assigning parameters to the Nelder–Mead simplex search algorithm is proposed. The proposed scheme is based on Chebyshev spacing points and adapts itself to the dimension of the problem. The numerical experiments conducted for this study show that the proposed scheme is better not just in comparison with the original Nelder–Mead algorithm but it outperforms the other existing adaptive schemes as well.

ARTICLE HISTORY
Received 19 August 2019
Accepted 23 October 2019

KEYWORDS
Nelder–Mead algorithm; high-dimensional problems; adaptive parameters; Chebyshev spacing points; convergence

1. Introduction

The need for optimization occurs in almost all fields of science and engineering. Numerous algorithms have been suggested in the literature to solve problems of optimization. The simplex search algorithm due to Nelder and Mead is one such algorithm and has been immensely popular since the year it was proposed (Nelder and Mead 1965). The algorithm is applied in almost every possible scientific field and has been cited thousands of times (Wright 2010). Besides being a direct search method that needs only function values to operate, the underlying idea behind the algorithm is quite simple and easy to implement.

However, the algorithm has received its fair share of criticism too. Apart from issues related to the convergence of the Nelder–Mead algorithm to a non-stationary point in low-dimensional problems (Dennis and Torczon 1991; McKinnon 1998), one of the major drawbacks that researchers have pointed out is related to the working of the algorithm on high-dimensional problems (Wright 1996; Han and Neumann 2006). The focus of the present study is related to the latter issue, *i.e.* the working of the Nelder–Mead algorithm when applied to high-dimensional problems.

There are four fundamental operations in the classical Nelder–Mead Algorithm (NMA), namely reflection, expansion, contraction and shrink. Associated with these operations, there are four different parameters. With the help of these four operations and related parameters, the algorithm tries to move towards the optimal solution. For a uniformly convex function, Gao and Han (2012) showed that the expansion and contraction operations of the NMA possess a *descent* property. However, in the same work they indicated that the efficiency of these operations reduces with an increase in problem size and the working of the algorithm is mainly dominated by the points generated using the

CONTACT V.K. Mehta vkmehta@tezu.ernet.in

reflection operation alone. In order to maintain the efficiency of the expansion and contraction operations, they suggested the Adaptive Nelder–Mead Simplex (ANMS) method. In their variant of the Nelder–Mead algorithm, the algorithm's parameters are a function of the dimension of the variable space n, which otherwise were suggested as some fixed constant by Nelder and Mead (1965). Fajfar, Puhan, and Bűrmen (2017) used genetic programming to evolve an NM-like algorithm which, interestingly enough, automatically evolved into a form that used significantly fewer reflections. Kumar and Suri (2014) too have suggested another variant of NMA with adaptive parameters. Their choice of parameters is the result of sensitivity analysis, which they performed over chained Rosenbrock functions of varying dimensions. Although not adaptive, there was one more attempt to improve the NMA in higher dimensions by Fajfar, Bűrmen, and Puhan (2019). They used a randomly perturbed centroid, which they showed experimentally cumulatively corrects the angle of the search direction, thus speeding up the convergence.

The parameters suggested in this work depend on the dimensions of the variable space and are based on Chebyshev spacing points. The scheme of using Chebyshev spacing points is new in the context of the NMA and the author stumbled upon this idea while working on a different problem of function approximation using Chebyshev spacing points. Concerning the function approximation using Chebyshev spacing points, it is well established that it minimizes the maximum deviation over the domain of interest (Dasgupta 2006). As pointed out earlier, to improve the performance of the NMA in higher-dimensions, it is desirable to find ways which result in a decrease in the use of the reflection step. In a way, it can be seen as a problem in which the task is to try and reduce (*minimize*) something which is taking a large (*maximum*) value. This flow of thought led the author to ask the question: 'Can Chebyshev help Nelder–Mead converge in high dimensions?'

The work presented here is the outcome of the numerical analysis which the author has undertaken to answer the same question.

In the following sections, after a brief overview of the NMA and Chebyshev spacing points, the proposed scheme of assigning the parameter-values is discussed. The performance of the proposed scheme on a large set of benchmark problems is demonstrated and compared with the existing schemes (Nelder and Mead 1965; Gao and Han 2012; Kumar and Suri 2014; Musafer and Mahmood 2018).

2. The Nelder–Mead simplex algorithm

As stated earlier, the NMA has four fundamental operations. In an n-dimensional problem, the algorithm starts with forming a simplex which is a polytope with $n+1$ vertices. After which, in every iteration, the algorithm replaces the current worst vertex with a new one, obtained using one of the four operations mentioned earlier. For the minimization problem, an outline of the algorithm, as given in Nelder and Mead (1965), is presented below.

For the current simplex with $n+1$ vertices, carry out the following steps.

Step 1 (Ordering). Order the vertices function values and identify the worst, second worst and the best vertices having function values $f_w = f(\mathbf{x}_w)$, $f_{sw} = f(\mathbf{x}_{sw})$ and $f_b = f(\mathbf{x}_b)$, respectively.

Step 2 (Centroid). Calculate the centroid of all the vertices, other than the worst vertex: $\mathbf{x}_c = (1/n)\sum_{k=1,k\neq w}^{k=n+1} \mathbf{x}_k$.

Step 3 (Reflection). Compute the reflection point $\mathbf{x}_r = \mathbf{x}_c + \alpha(\mathbf{x}_c - \mathbf{x}_w)$ and evaluate the function value at that point, $f_r = f(\mathbf{x}_r)$. If $f_b \leq f_r < f_{sw}$, then replace $\mathbf{x}_w$ with $\mathbf{x}_r$.

Step 4 (Expansion: If $f_r < f_b$, then compute the expansion point $\mathbf{x}_e = \mathbf{x}_c + \beta(\mathbf{x}_c - \mathbf{x}_w)$, where $\beta > \alpha$. Evaluate $f_e = f(\mathbf{x}_e)$. If $f_e < f_r$, then replace $\mathbf{x}_w$ with $\mathbf{x}_e$; else replace $\mathbf{x}_w$ with $\mathbf{x}_r$.

Step 5 (Contraction)

Outside: If $f_{sw} \leq f_r < f_w$, replace $\mathbf{x}_w$ with $\mathbf{x}_r$; compute the outside contraction point $\mathbf{x}_{oc} = \mathbf{x}_c + \gamma(\mathbf{x}_c - \mathbf{x}_w)$ and evaluate $f_{oc} = f(\mathbf{x}_{oc})$. If $f_{oc} \leq f_r$, replace replace $\mathbf{x}_w$ with $\mathbf{x}_{oc}$; else go to Step 5.

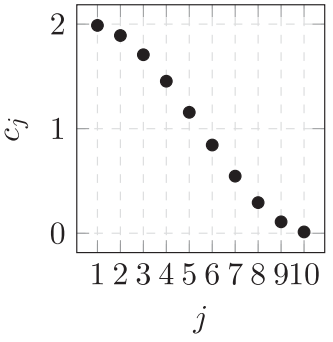

Figure 1. Chebyshev spacing points, and their values c_j, in the interval [0, 2] for $n_c = 10$.

Inside: If $f_r \geq f_w$, compute the inside contraction point $\mathbf{x}_{ic} = \mathbf{x}_c - \gamma(\mathbf{x}_c - \mathbf{x}_w)$ and evaluate $f_{ic} = f(\mathbf{x}_{ic})$. If $f_{ic} < f_w$, replace $\mathbf{x}_w$ with $\mathbf{x}_{ic}$; else go to Step 5.

Step 6 (Shrink). Reduce the simplex by pulling every point towards the best vertex by a factor δ, where $\mathbf{x}_i = \mathbf{x}_b + \delta(\mathbf{x}_i - \mathbf{x}_b)$.

These above mentioned steps are repeated until convergence up to the required tolerance is achieved. In their original study, Nelder and Mead suggested using the following values for the parameters: $\alpha = 1.0$, $\beta = 2.0$ and $\delta = \gamma = 0.5$.

3. Chebyshev spacing points

The Chebyshev spacing points are the zeros of the Chebyshev polynomial $T_{n_c}(x)$, where n_c denotes the order of the polynomial (Dasgupta 2006). These zeros are all real and lie in the interval $[-1, 1]$ at $x_j = \cos[(2j-1)\pi/2n_c]$, $j = 1, 2, 3, \ldots, n_c$. The location of these spacing points over a general interval $[a, b]$ is given by Equation (1). Figure 1 shows the Chebyshev spacing points and their locations in the interval $[0, 2]$ for $n_c = 10$.

$$c_j = \frac{b+a}{2} + \frac{b-a}{2} \cos\frac{(2j-1)\pi}{2n_c} \quad (j = 1, 2, 3, \ldots, n_c). \tag{1}$$

The Chebyshev spacing points are particularly important in the field of minimax approximation, where they are used in order to obtain an approximation of the function that minimizes the maximum deviation over the domain of interest.

4. The proposed scheme

As mentioned earlier, the proposed scheme for assigning the parameter values α, β, γ and δ is based on Chebyshev spacing points. In the proposed scheme, for an n-dimensional problem, *some* number of spacing points n_c are generated in the interval $[0, 2]$.[1] Thereafter, values at *some* of these locations of the spacing points are assigned to the parameters in such a way that $\beta > \alpha > \delta > \gamma$. Now, the obvious questions are as follows.

(i) How many Chebyshev spacing points n_c should be generated for an n-dimensional problem?
(ii) The values of which among these should be assigned to the NMA parameters?

There exist numerous ways to do this. Two versions of the proposed scheme that settle these questions differently are presented here.

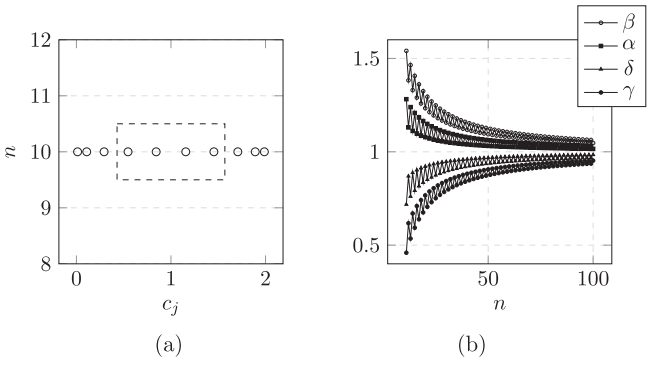

Figure 2. (a) A schematic showing the proposed (crude) scheme of assigning the NM algorithm's parameters for $n = 10$. (b) Variation of the NMA parameter values for the crude version.

Crude version

In this version of the proposed scheme, for an n-dimensional problem, n spacing points are generated in the interval $[0, 2]$, *i.e.* $n_c = n$, and values of four centrally located Chebyshev spacing points are assigned to the NMA parameters. Hence, under this crude scheme, for an n-dimensional problem, the values assigned to the parameters are given by the following expressions:

$$\begin{aligned} \beta &= 1 + \cos\frac{(n-3-(n\%2))\pi}{2n}, \quad \alpha = 1 + \cos\frac{(n-1-(n\%2))\pi}{2n}, \\ \delta &= 1 + \cos\frac{(n+1+(n\%2))\pi}{2n}, \quad \gamma = 1 + \cos\frac{(n+3+(n\%2))\pi}{2n}, \end{aligned} \tag{2}$$

where $n\%2$ denotes the remainder of dividing n by 2. For $n = 10$, the above mentioned assignment scheme is shown in Figure 2(a). Figure 2(b) shows the variation of the proposed values of parameters for $11 \leq n \leq 100$.

Refined version

During the course of the present work, it was found that the performance of the proposed scheme is very sensitive to the number of Chebyshev spacing points n_c and the choice of location of the Chebyshev nodes for parameter values. A parametric study was conducted on a quadratic problem, for $n = 11, 12, \ldots, 60$, to see how the number of Chebyshev points and their location affect the performance of the proposed scheme, and the outcome of this study is the refined version of the proposed scheme. For the refined version, for an n-dimensional problem, the expressions for obtaining the values of the four parameters are as follows:

$$\begin{aligned} \beta &= 1 + \cos\frac{(n_c-3)\pi}{2n_c}, \quad \alpha = 1 + \cos\frac{(n_c-1)\pi}{2n_c}, \\ \delta &= 1 + \cos\frac{(n_c+3)\pi}{2n_c}, \quad \gamma = 1 + \cos\frac{(n_c+5)\pi}{2n_c}, \end{aligned} \tag{3}$$

where $n_c = 2(9 + \lfloor (n-1)/5 \rfloor)$. Hence, according to the refined version of the proposed scheme, the number of Chebyshev spacing points is not equal to the problem dimension—see Table 1—whereas in the crude version it *is* equal to the problem dimension. Also, the choice for the location of the NMA parameters δ and γ is shifted one-step leftward (see Figure 3(a)). In effect, with increasing n, the rate at which parameter values decrease in the case of the refined version is slower in comparison with the crude version (see Figure 3(b)).

Table 1. The relationship between n and n_c for the refined version of the proposed scheme.

n	11–15	16–20	21–25	26–30	⋯	91–95	96–100	⋯
n_c	22	24	26	28	⋯	54	56	⋯

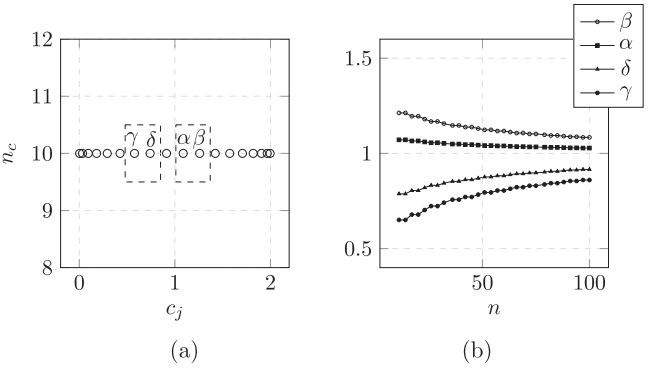

Figure 3. (a) A schematic showing the proposed (refined) scheme of assigning the NM algorithm's parameters for $n_c = 18$. (b) Variation of the NMA parameter values for the refined version.

5. Working of the proposed scheme

Both versions of the proposed scheme are first tested and compared with the Nelder–Mead implementation with the default parameter values of $\alpha = 1.0$, $\beta = 2.0$ and $\delta = \gamma = 0.5$ on a modified convex function (Equation (4)) of varying dimensions (Gao and Han 2012).

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{D}\mathbf{x} + \sigma(\mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x})^2, \tag{4}$$

where $\mathbf{D}$ is the positive definite matrix having only non-zero diagonal entries as

$$\mathbf{D} = \mathrm{diag}\left([(1+\epsilon), (1+\epsilon)^2, (1+\epsilon)^3, \ldots, (1+\epsilon)^n]\right)$$

and $\mathbf{B}$ is another positive definite matrix defined as

$$\mathbf{B} = \mathbf{U}^{\mathrm{T}}\mathbf{U}, \quad \mathbf{U} = \begin{bmatrix} 1 & \ldots & 1 \\ & \ddots & \vdots \\ & & 1 \end{bmatrix}.$$

The parameters ϵ and σ control the condition number of matrix $\mathbf{D}$ and the deviation of the function $f(\mathbf{x})$ from quadratic, respectively.

The Nelder–Mead algorithm as described in Section 2 is coded in the C language. The algorithm was run with default and proposed parameters and with termination criteria: $Tol_{Fun} = 10^{-10}$, $Tol_X = 10^{-10}$ and $Max_{FunEval} = 10^6$. The initial point $\mathbf{x}_0$ for all the cases was taken as $\mathbf{x}_0 = [1, 1, \ldots, 1] \in \mathbb{R}^n$ and the initial simplex was generated around the initial point $\mathbf{x}_0$ using Pfeffer's method (Baudin 2009). Using Pfeffer's method, one can create a unique simplex in relation to a given initial point $\mathbf{x}_0$. The first vertex of the simplex is the initial guess $\mathbf{v}_1 = \mathbf{x}_0$, and the other vertices are created using the rule given by

$$(\mathbf{v}_i)_j = \begin{cases} (\mathbf{x}_0)_j + \delta_u(\mathbf{x}_0)_j, & \text{if } j = i-1 \text{ and } (\mathbf{x}_0)_{j-1} \neq 0, \\ \delta_z, & \text{if } j = i-1 \text{ and } (\mathbf{x}_0)_{j-1} = 0, \\ (\mathbf{x}_0)_j, & \text{if } j \neq i-1, \end{cases} \tag{5}$$

for vertices $i = 2, \ldots, n+1$ and components $j = 1, \ldots, n$. Different values of δ_u and δ_z have been used/suggested in the literature (Fan 2002; Baudin 2009; Gao and Han 2012). However, the present

Table 2. Comparison of the proposed schemes with the NM algorithm and default parameters on a modified quadratic function.

				Proposed scheme			
		NM with default parameters		Crude		Refined	
	n	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}
$\epsilon = 0.0, \sigma = 0.0$	20	2.559,776e−20	9,773	3.356,880e−21	4,855	3.154,713e−21	**3,576**
	30	7.478,956e−19	32,763	3.928,473e−21	12,163	3.231,794e−21	**6,522**
	40	4.155,033e−19	39,591	4.432,467e−21	23,146	3.231,709e−21	**10,735**
	50	1.254,128e−17	90,416	4.285,665e−21	37,649	4.710,285e−21	**16,145**
	60	4.932,140e−18	149,512	5.776,418e−21	52,041	4.472,419e−21	**20,112**
	70	8.455,876e−19	59,828	5.099,626e−21	72,797	3.867,888e−21	**26,353**
	80	5.008,946e−19	93,250	4.673,399e−21	100,238	4.822,721e−21	**36,344**
	90	2.807,202e−18	129,265	4.944,304e−21	126,722	4.175,096e−21	**41,622**
	100	4.331,079e−18	226,034	4.931,429e−21	160,072	4.558,113e−21	**53,969**
$\epsilon = 0.05, \sigma = 0.0001$	20	5.059,948e−20	15,100	7.277,924e−21	4,901	5.429,537e−21	**3,834**
	30	2.227,478e−02	180,456	8.423,050e−21	12,036	7.260,932e−21	**9,038**
	40	1.519,757e−01	422,196	1.122,261e−20	20,675	7.404,530e−21	**15,187**
	50	2.125,446e−01	475,379	1.347,419e−20	35,396	1.048,766e−20	**23,289**
	60	7.931,505e+00	1,000,000	1.674,156e−20	53,847	1.422,136e−20	**31,713**
	70	1.264,732e+02	1,000,000	2.385,593e−20	76,094	1.326,168e−20	**60,164**
	80	6.564,232e+02	1,000,000	3.130,699e−20	103,744	1.833,035e−20	**87,305**
	90	1.078,628e+03	1,000,000	3.519,043e−20	158,765	2.188,087e−20	**119,501**
	100	3.163,852e+01	1,000,000	4.014,644e−20	205,849	1.993,576e−20	**168,680**

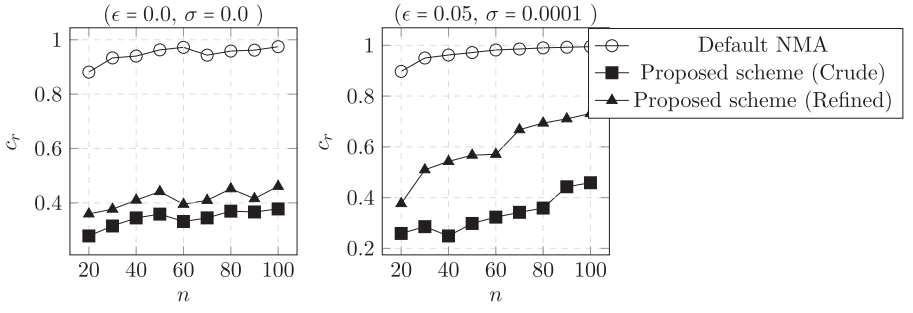

Figure 4. Contribution of reflection steps (c_r) with increase in dimension of the modified quadratic problem.

study was conducted with

$$\delta_u = \max\{1, ||\mathbf{x}_0||_\infty\}, \quad \delta_z = 0.000{,}25. \tag{6}$$

Table 2 presents the performance of NM algorithm with default parameters along with proposed schemes on the modified quadratic problem. The table shows the minimum function value $f(\mathbf{x})$ obtained by a particular algorithm at the termination along with the number of function evaluations $N_{f_{eval}}$ which it has taken to achieve that minimum value for two pairs of (ϵ, σ). As can clearly be seen from the table, in comparison with the default scheme, proposed schemes have performed well in terms of both the accuracy and the number of function evaluations. Moreover, as one can see in Figure 4, the proposed schemes have significantly reduced the contribution of reflection operation (c_r). For ($\epsilon = 0.0, \sigma = 0.0$), with the crude scheme, it is well below 40% for all the dimensions. With the refined scheme it is a bit higher than the crude scheme (well below 50% for all the dimensions), but it has resulted in less number of function evaluations than the crude scheme. Similar trend can be seen in the figure corresponding to ($\epsilon = 0.05, \sigma = 0.0001$).

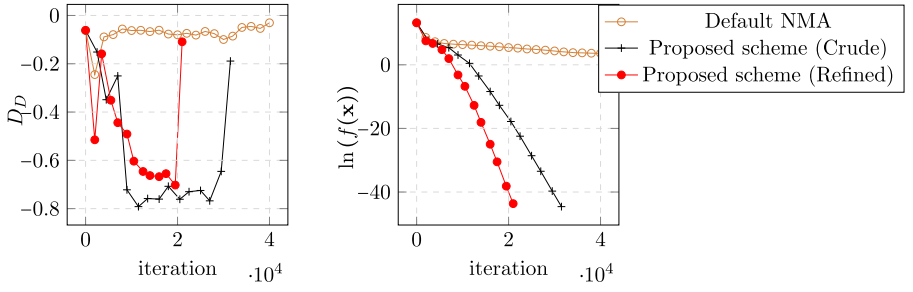

Figure 5. Typical convergence of the NMA with the default and proposed schemes for a modified quadratic with $\epsilon = 0.05$, $\sigma = 0.0001$ for $n = 60$ in terms of D_D and function value. Please note that the x-axis in the above figures is depicting iteration number which is not same as the number of function evaluations $N_{f_{eval}}$.

How does the scheme work?

To minimize a function, it is important for an algorithm to search in directions that are *descent* in nature. A direction $\mathbf{d}$ is descent if the function value decreases while moving in that direction (locally). Mathematically, a direction $\mathbf{d}$ is descent if

$$D_D = \frac{\mathbf{d}^{\mathrm{T}} \nabla f(x)}{\| \mathbf{d} \| \| \nabla f(x) \|} < 0, \tag{7}$$

where $\nabla f(x)$ is the gradient at the current point. For NMA, the search direction from the current worst point $\mathbf{x}_w$ is always along the current centroid $\mathbf{x}_c$.

From Table 2, it can be seen that for ($\epsilon = 0.05$, $\sigma = 0.0001$), the default NMA fails to converge to the true minimum, $f(x^*) = 0$, for all the dimensions except $n = 20$. With the default parameter choice, Torczon (1989) has shown numerically that the NM algorithm fails when the search direction becomes almost orthogonal to the negative gradient. This can be seen in Figure 5, wherein the typical convergence of the original and proposed schemes in terms of D_D (refer to Equation (7)) and function value (in logarithmic scale) with iteration is shown for ($\epsilon = 0.05, \sigma = 0.0001, n = 60$). One can clearly see that for the default scheme, with the algorithm's progress the search direction becomes nearly orthogonal to the gradient. However, the algorithm with the proposed parameter values maintains a sufficient component along the negative gradient with the effect that it could converge to the actual minimum with sufficient accuracy.

6. Comparison with existing schemes

A detailed numerical study on the benchmark problems suggested by Gao and Han (2012) and Moré, Garbow, and Hillstrom (1981) (the MGH problem set) is performed to measure the effectiveness of the proposed schemes in comparison with the existing schemes. The performance of the various schemes are compared using *data profiles* as suggested by Moré and Wild (2009).

Data profiles provide a way to compare the performance of different derivative-free optimization solvers from a set S which are tested on a set of P benchmark problems. For each $s \in S$, data profiles are defined in terms of a ratio of the performance measure $t_{p,s}$ and the dimension n_p of the problem $p \in P$. They are given by

$$d_s(\kappa) = \frac{1}{|P|} \text{size} \left\{ p \in P : \frac{t_{p,s}}{n_p + 1} \leq \kappa \right\}. \tag{8}$$

The performance measure $t_{p,s}$ is taken as the number of function evaluations that a particular solver $s \in S$ has taken to solve a particular problem $p \in P$. Hence, the data profile $d_s(\kappa)$ can be seen as the percentage of problems that can be solved by a particular solver s with equivalent number of κ *simplex*

Table 3. Comparison of different schemes on Gao and Han's modified quadratic function of varying dimensions.

								Proposedscheme			
		Default NMA		ANMS		HNM		Crude		Refined	
	n	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}
$\epsilon = 0.0$, $\sigma = 0.0$	10	3.15e−09	**487**	5.91e−09	898	**2.10e−301**	40,465	1.22e−08	598	3.60e−09	621
	20	5.16e−08	2,271	1.13e−08	2,259	**0.00e+00**	89,381	3.37e−09	2,326	3.14e−09	**1,776**
	30	1.43e−08	3,593	1.55e−08	4,072	**0.00e+00**	148,198	4.19e−09	5,676	3.48e−09	**3,309**
	40	1.02e−08	**4,796**	1.76e−08	7,122	**2.30e−212**	131,675	4.88e−09	10,069	4.85e−09	5,258
	50	3.11e−08	8,986	2.08e−08	9,488	**1.40e−251**	186,554	4.59e−09	16,595	3.24e−09	**7,848**
	60	3.94e−08	11,306	3.50e−08	13,754	**2.94e−218**	187,109	4.55e−09	24,178	4.05e−09	**10,367**
$\epsilon = 0.05$, $\sigma = 0.0$	10	6.61e−09	680	9.05e−09	910	N/A		**3.98e−09**	633	4.02e−09	**603**
	20	4.16e−08	2,984	1.84e−08	2,548	N/A		5.88e−09	2,289	**5.56e−09**	**1,786**
	30	7.51e−08	4,057	2.66e−08	5,067	N/A		6.73e−09	5,709	**6.38e−09**	**3,134**
	40	1.17e−07	7,988	3.68e−08	8,598	N/A		1.14e−08	10,142	**1.01e−08**	**4,966**
	50	4.74e−07	22,370	6.71e−08	13,167	N/A		1.22e−08	16,137	**1.00e−08**	**7,554**
	60	1.17e−06	29,439	6.89e−08	20,860	N/A		1.63e−08	25,245	**1.25e−08**	**10,375**
$\epsilon = 0.0$, $\sigma = 0.0001$	10	**1.79e−09**	716	1.46e−08	1,088	N/A		3.42e−09	846	2.77e−09	**664**
	20	3.71e−08	6,563	2.84e−08	4,134	N/A		**3.56e−09**	2,402	4.14e−09	**1,831**
	30	3.05e−06	32,639	4.06e−08	13,148	N/A		**3.63e−09**	5,735	4.16e−09	**4,798**
	40	0.027,3	163,546	9.301e−08	21,195	N/A		4.25e−09	10,866	**3.84e−09**	**9,924**
	50	24.431	311,197	8.17e−08	42,403	N/A		**3.98e−09**	19,842	6.66e−09	**15,189**
	60	23.225	949,219	1.05e−06	59,626	N/A		**4.58e−09**	36,554	5.60e−09	**24,503**
$\epsilon = 0.05$, $\sigma = 0.0001$	10	1.64e−08	816	6.04e−09	994	**2.38e−211**	28,223	6.32e−09	**585**	2.90e−09	631
	20	4.62e−07	4,626	1.52e−08	3,788	**4.97e−236**	63,665	4.60e−09	2,404	5.70e−09	**2,040**
	30	0.024,1	64,311	4.03e−08	10,251	**0.00e+00**	135,636	7.71e−09	6,165	8.91e−09	**5,628**
	40	0.151,9	193,158	5.74e−08	18,898	**1.05e−242**	127,958	1.23e−08	10,850	1.13e−08	**9,510**
	50	0.212,5	214,846	4.74e−07	37,282	**2.94e−218**	187,109	1.40e−08	20,179	1.10e−08	**15,713**
	60	7.931,5	730,290	2.07e−07	61,259	**1.80e−180**	148,225	2.04e−08	31,925	1.70e−08	**19,619**

gradient estimates (Moré and Wild 2009). By fixing κ to a particular value, one can compare different solvers on the basis of $d_s(\kappa)$.

The above mentioned comparison takes into account a convergence test given by

$$f(\mathbf{x}_0) - f(\mathbf{x}) \geq (1 - \tau)(f(\mathbf{x}_0) - f_L), \tag{9}$$

where $f(\mathbf{x}_0)$ is the function value at the starting point, $f(\mathbf{x})$ is the minimum function value obtained by a particular solver s and f_L is the smallest estimate of the global minimizer obtained by any of the solvers. The convergence test ensures that the reduction $f(\mathbf{x}_0) - f(\mathbf{x})$ obtained by a particular solver is at least $(1 - \tau)$ times the best possible reduction. If a certain pair (p, s) fails to pass the convergence test, then the value of the performance measure is taken to be infinity, *i.e.* $t_{p,s} = \infty$.

First, the performances of the proposed schemes are compared with those of NMA with the default parameters (Nelder and Mead 1965), ANMS (Gao and Han 2012) and HNM (Musafer and Mahmood 2018) on the modified convex function.

Table 3 presents the comparative performances of the proposed schemes along with those of ANMS and HNM on the modified convex function. The results for the schemes ANMS and HNM are taken from the respective studies. For the default and proposed schemes, the initial point $\mathbf{x}_0$ for all cases is taken as $\mathbf{x}_0 = [1, 1, \ldots, 1] \in \mathbb{R}^n$. The termination criteria used are as follows: $Tol_{Fun} = 10^{-4}$, $Tol_X = 10^{-4}$ and $Max_{FunEval} = 10^6$.

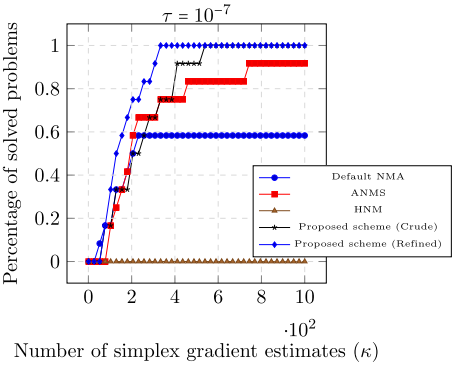

Figure 6. Data profile $d_s(\kappa)$ of different schemes for Gao and Han's function.

For each of the problems, two values are reported for every scheme: the minimum function value obtained $f(\mathbf{x})$, and the number of function evaluations $N_{f_{eval}}$ used for obtaining the minimum value. An ellipse around the minimum function value obtained by a particular algorithm signifies that the minimum value obtained by the algorithm could not pass the convergence test given by Equation (9).

On comparing these values, one can see that, as far as closeness to the actual minimum ($f(\mathbf{x}^*) = 0$) is concerned, the algorithm HNM is the most desirable. However, the cost of obtaining such close estimates of actual minima is too high in the case of HNM. This can be clearly seen in Figure 6, which shows a comparison of the variation of the data profile $d_s(\kappa)$ for the default NMA, ANMS, HNM and the proposed schemes. As is evident from Figure 6, the algorithm HNM cannot solve any of Gao and Han's modified quadratic problems even after 1000 simplex gradient estimates. Even NMA with the default parameters performs better than HNM for 1000 simplex gradient estimates. On the other hand, the proposed schemes perform much better in comparison with all the other schemes on these problems. The refined version of the proposed scheme can solve all the problems with a tolerance of $\tau = 10^{-7}$ well within 400 simplex gradient estimates, whereas the crude version can manage the same within 600 simplex gradient estimates. The ANMS can manage to solve only 90% of the problems in 1000 simplex gradient estimates.

Table 4 shows the comparative performance of the proposed schemes along with the default NMA, ANMS and HNM on the MGH problem set. For the problems of this set, the initial point $\mathbf{x}_0$ is also prescribed. The same is used as the initial point in the present study. Here, again for most of the problems, the accuracy of the HNM is better in comparison with the other schemes. However, for this problem set too, HNM proves to be very costly.

Figure 7 shows the variation of data profiles for all the schemes for this problem set. At 1000 simplex gradient estimates, the HNM can solve only 30% of the total number of problems, which is inferior to the default NMA. The default NMA can solve approximately 40% of the total number of problems. The ANMS does better than the default NMA and HNM as it can solve 50% of the total number of problems with those many simplex gradient estimates. However, the proposed schemes do much better as they can solve close to 70% of the problems with 1000 simplex gradient estimates. Even after 5000 simplex gradient estimates, both versions of the proposed schemes are well ahead of the existing schemes.

The average performance of the proposed schemes is also compared with the ANMS of Gao and Han (2012)—equation set (10)—and the adaptive scheme of Kumar and Suri (2014)—equation set (11). For different schemes, the variation of parameter values with problem dimension n is depicted in Figure 8.

Table 4. Comparison of the different schemes with the Moré–Garbow–Hilstrom (MGH) problem set.

								Proposedscheme			
		DefaultNMA		ANMS		HNM		Crude		Refined	
	n	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}
Extended Rosenbrock function	12	3.98e+00	5,344	3.39e−09	10,015	**0.00e+00**	38,668	3.98e+00	4,146	3.64e−09	**5,318**
	18	8.26e+00	14,876	4.22e−09	**29,854**	**9.53e−23**	3,424,010	3.98e+00	10,068	3.98e+00	9,959
	24	1.07e+01	71,860	4.25e−09	50,338	**2.43e−27**	1,316,486	1.58e−08	**24,653**	2.98e−08	25,684
	30	2.41e+01	62,894	5.44e−09	156,302	**7.51e−10**	5,004,910	2.10e−08	**36,968**	3.98e+00	34,308
	36	2.73e+01	142,569	1.66e−08	**119,135**	**3.76e−28**	122,109	3.98e+00	44,169	3.98e+00	51,030
Extended Powell singular function	12	4.86e−10	**1,596**	3.94e−08	25,961	**8.74e−304**	267,683	4.41e−11	2,249	1.61e−12	1,998
	24	1.21e−09	12,878	4.87e−07	11,156	**5.68e−278**	517,754	1.02e−11	10,494	4.16e−11	**8,210**
	40	3.84e−07	47,609	9.91e−06	38,530	**6.46e−219**	705,169	4.92e−11	34,008	2.40e−10	**20,789**
	60	1.37e−05	135,469	1.91e−05	71,258	**9.85e−178**	804,590	1.89e−10	94,924	1.76e−08	**41,552**
Penalty I function											
	10	8.07e−05	1,960	**7.08e−05**	5,410	7.63e−05	**716**	7.61e−05	4,961	7.51e−05	4,803
Penalty II function											
	10	2.95e−04	**2,841**	**2.93e−04**	9,741	2.96e−04	1,556,688	2.96e−04	4,747	2.97e−04	6,624
Variably dimensioned function	12	1.81e+05	6,954	8.62e−09	4,709	**4.30e−27**	9,198	9.01e−09	6,191	5.07e−09	**4,504**
	18	1.49e+05	14,554	1.09e−08	12,815	**4.65e−17**	**8,164**	4.21e−09	14,366	3.43e−09	11,766
	24	1.58e+06	25,004	1.12e−08	35,033	**3.96e−19**	**10,419**	3.56e−09	25,421	3.17e−09	26,351
	30	1.32e+06	33,680	1.59e−08	67,717	**4.28e−24**	**17,190**	4.22e−09	38,676	4.11e−09	48,989
	36	1.36e+05	61,947	1.81e−08	209,340	**1.49e−10**	**9,782**	3.89e−09	53,840	3.92e−09	65,073
Trigono-metric function	10	4.21e−05	1,130	2.79e−05	961	**3.26e−263**	**42,592**	4.21e−05	970	4.21e−05	728
	20	7.17e−06	7,007	1.35e−06	4,194	**8.81e−145**	**47,431**	6.86e−06	2,836	6.86e−06	2,498
	30	1.93e−05	7,842	9.91e−07	8,202	**4.77e−131**	**62,440**	5.00e−06	5,318	5.30e−06	4,436
	40	7.39e−06	33,108	1.56e−06	17,674	**6.50e−228**	**136,943**	2.05e−05	6,161	6.78e−06	5,271
	50	8.76e−06	29,988	3.65e−07	19,426	**4.51e−239**	**194,040**	1.03e−05	11,009	8.26e−06	7,324
	60	7.41e−06	40,231	9.66e−07	31,789	**1.88e−152**	**119,524**	6.69e−06	14,543	5.02e−06	10,216

(*continued*).

Table 4. Continued.

		Default NMA		ANMS		HNM		Proposed scheme: Crude		Proposed scheme: Refined	
	n	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}	$f(\mathbf{x})$	N_{feval}
Discrete boundary function	10	3.32e−10	1,249	1.03e−07	1,029	**3.45e−26**	978,531	3.56e−10	1,485	1.00e−10	**1,082**
	20	1.94e−09	16,235	3.17e−10	7,535	7.10e−08	1,868,130	8.57e−11	4,534	**5.87e−11**	**4,509**
	30	7.88e−04	49,680	3.00e−05	3,860	9.99e−07	2,195,921	4.44e−11	11,445	**3.20e−11**	**9,875**
	40	6.54e−04	49,452	1.61e−05	1,029	9.94e−06	14,950	2.75e−11	23,735	**2.03e−11**	**19,499**
	50	1.40e−03	156,357	8.86e−06	1,905	5.70e−06	11,858	3.80e−11	41,909	**1.96e−11**	**30,346**
	60	4.88e−04	186,109	5.30e−06	2,125	3.59e−06	7,922	4.36e−11	65,262	**1.50e−11**	**45,834**
Discrete integral function	10	6.05e−09	**419**	9.59e−09	774	**5.00e−30**	6,243	1.03e−08	513	2.31e−09	536
	20	2.98e−08	1,878	1.08e−08	3,320	**6.22e−23**	12,686	4.07e−09	1,791	3.81e−09	**1,276**
	30	2.22e−08	3,963	2.11e−08	8,711	**8.68e−26**	18,545	3.97e−09	4,075	3.43e−09	**2,600**
	40	5.36e−08	11,028	4.07e−08	18,208	**9.33e−20**	21,829	3.85e−09	7,911	3.72e−09	**4,793**
	50	1.15e−07	17,241	4.76e−08	25,961	**2.21e−26**	28,883	4.09e−09	13,239	3.83e−09	**6,538**
	60	5.72e−07	34,180	2.26e−08	38,908	**9.21e−15**	19,539	4.75e−09	19,665	3.71e−09	**11,722**
Broyden tridiagonal function	10	1.22e−07	673	2.55e−07	740	**5.86e−30**	12,707	2.82e−07	953	4.27e−08	**597**
	20	5.07e−07	3,092	2.91e−07	3,352	**9.04e−30**	29,731	8.28e−08	2,223	5.68e−08	**1,805**
	30	3.02e−06	6,120	3.69e−07	11,343	**8.25e−29**	56,107	1.21e−07	4,772	1.00e−07	**3,205**
	40	3.17e−06	12,477	4.40e−07	23,173	**4.82e−15**	36,714	1.26e−07	8,376	1.02e−07	**4,942**
	50	1.53e−06	12,177	5.09e−07	42,013	**3.90e−28**	79,970	1.44e−07	13,496	1.07e−07	**7,199**
	60	1.91e−06	18,193	7.18e−07	64,369	**4.98e−28**	107,131	1.54e−07	19,125	1.34e−07	**9,303**
Broyden banded function	10	2.55e−06	951	2.21e−07	**741**	**8.03e−31**	11,711	1.12e−05	850	1.62e−12	3,155
	20	6.68e−09	12,228	5.19e−07	**1,993**	**6.62e−31**	35,617	1.26e−07	2,400	3.81e−04	2,226
	30	3.23e+00	52,873	6.44e−07	**3,686**	**9.23e−30**	50,482	2.53e−09	7,062	2.38e−09	4,575
	40	2.00e−07	17,035	1.08e−06	**6,060**	**7.63e−30**	76,731	3.78e−09	13,983	2.31e−09	7,004
	50	2.16e−06	56,927	1.23e−06	**8,357**	**5.66e−30**	96,263	8.01e−09	17,838	3.04e−09	9,480
	60	7.87e−08	24,169	1.00e−06	**10,630**	**6.48e−27**	95,163	7.61e−09	30,855	1.83e−09	16,783

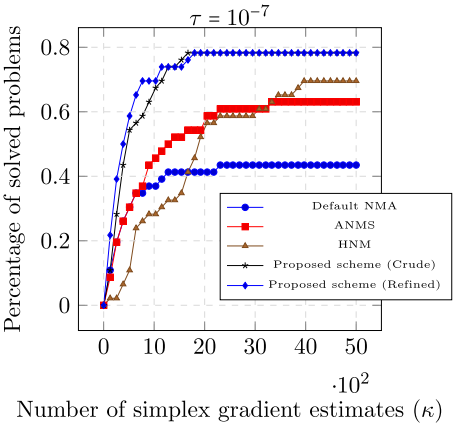

Figure 7. Data profile $d_s(\kappa)$ of different schemes on the MGH problem set.

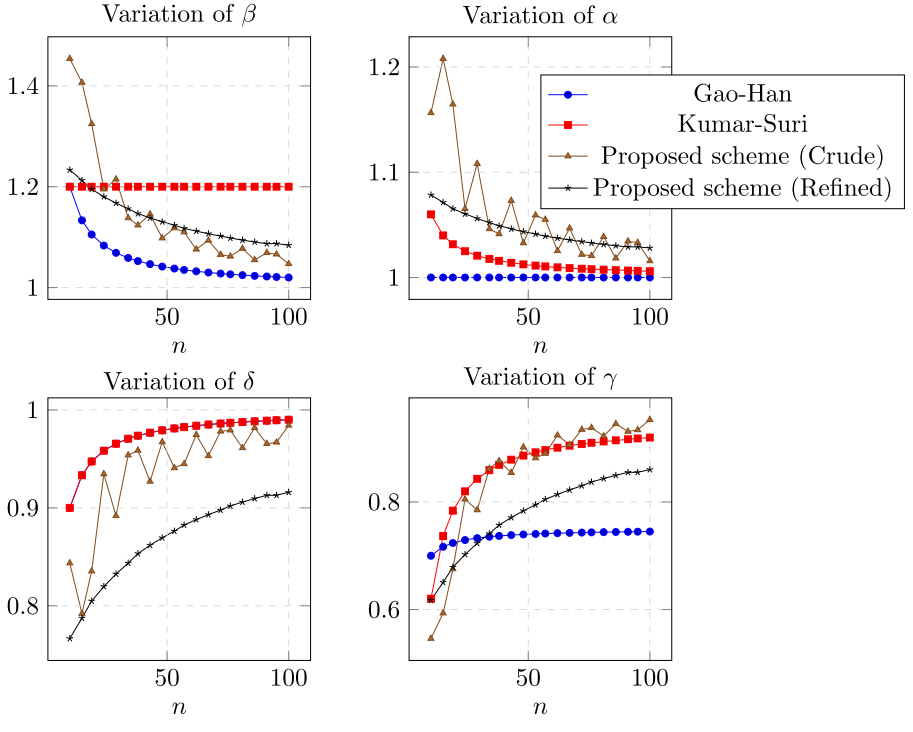

Figure 8. Variation of parameters β, α, δ and γ for different schemes.

Gao and Han's scheme :

$$\beta = 1.0 + \frac{2}{n}, \quad \alpha = 1.0,$$
$$\delta = 1.0 - \frac{1}{n}, \quad \gamma = 0.75 - \frac{1}{2n}. \tag{10}$$

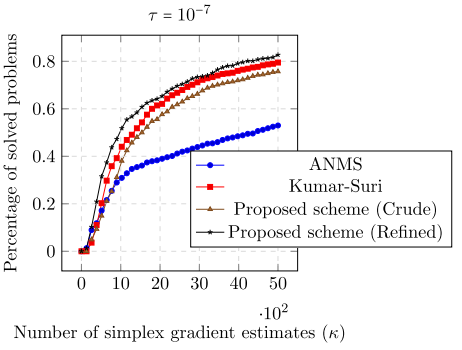

Figure 9. Data profile $d_s(\kappa)$ of different schemes for modified quadratic ($\epsilon = 0.0, \sigma = 0.0$) and MGH problem set.

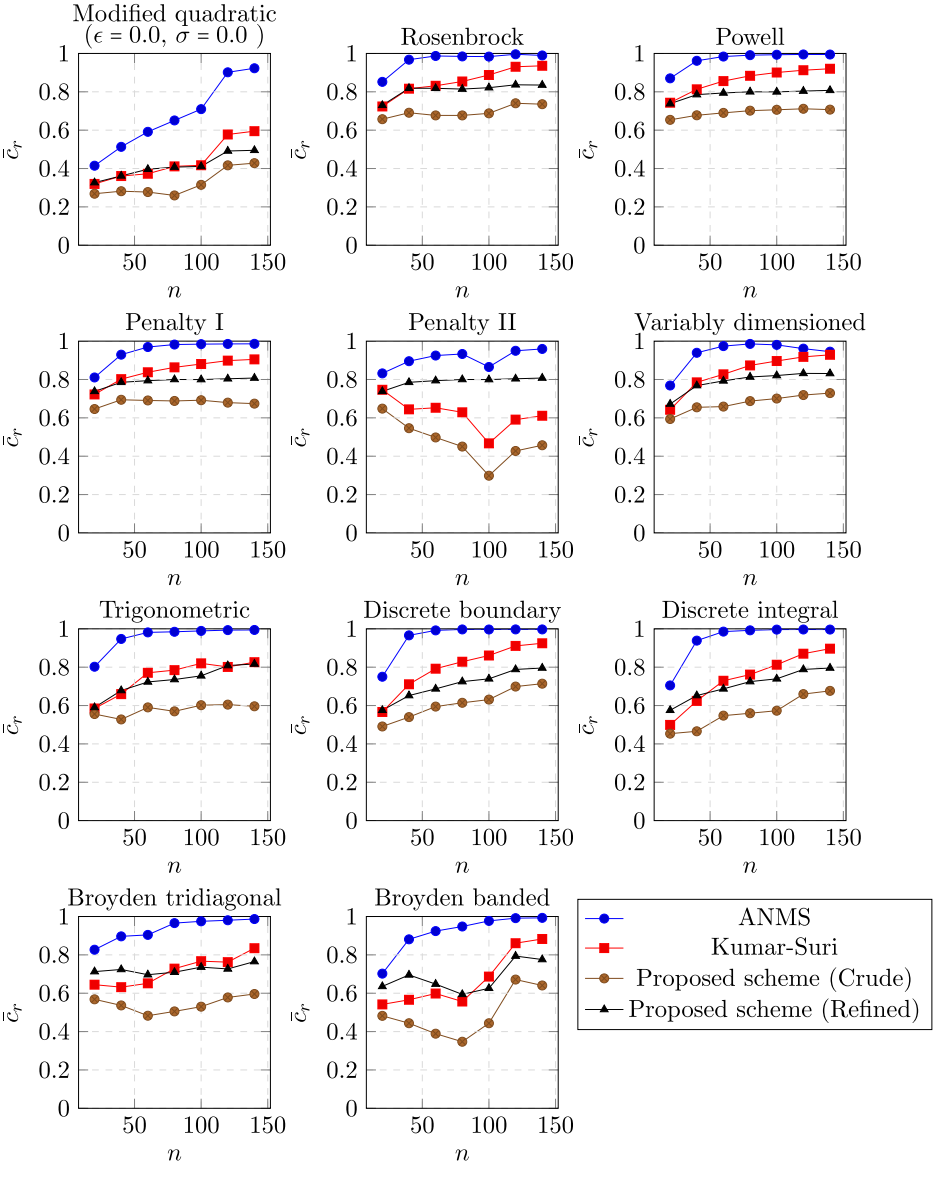

Figure 10. Variation of $\bar{c}_r$ for different schemes.

Kumar and Suri's scheme :

$$\beta = 1.2, \quad \alpha = 1.0 + \frac{0.6}{n},$$
$$\delta = 1.0 - \frac{1}{n}, \quad \gamma = 0.95 - \frac{3}{n} - \frac{3}{n^2}. \tag{11}$$

In their original study, Kumar and Suri (2014) tested their version on a different problem set. In order to compare the performance on the MGH problem set, the results for Kumar and Suri's version are obtained here by the author using the parameters suggested by Kumar and Suri (2014). Also, since the results provided in Gao and Han (2012) for ANMS are for a single initial point, the results for their version are obtained here by the author using the parameters suggested by Gao and Han (2012).

Ten independent runs for each of the problems of the MGH problem set and the modified quadratic function (with $\epsilon = 0.0$, $\sigma = 0.0$) are taken for the schemes of Gao–Han, Kumar–Suri and the proposed schemes with the same initial point and termination criteria of $Tol_{Fun} = 10^{-4}$, $Tol_X = 10^{-4}$ and $Max_{FunEval} = 10^6$.

The dimension n of the problem is varied from 20 to 140. Figures 9 and 10 present the outcome of this study with respect to data profiles $d_s(\kappa)$ and average contribution of the reflection step $\bar{c}_r$, respectively. As can clearly be seen, the schemes of Kumar–Suri and the proposed schemes dominate the ANMS algorithm in terms of both, by maintaining a higher percentage of solved problems represented by data profiles and keeping a low fraction of reflection steps. The performance of the scheme of Kumar and Suri and the proposed schemes are more or less comparable in terms of data profiles, as all these schemes can solve 75 to 85% of the problems with 5000 simplex gradient estimates. However, in terms of the average contribution of the reflection step $\bar{c}_r$, the proposed schemes perform better in the case of most of the problems. If one extrapolates for the trend in the increase in $\bar{c}_r$ with an increase in dimensions of the problem (Figure 10), it appears that, for n greater than 140, the proposed schemes will work with fewer reflection operations than the existing schemes.

7. Conclusion

There is no doubt that when it comes to the working of the algorithm in high dimensions, the choice of default parameter values for Nelder–Mead's algorithm as suggested in the original study is quite inefficient. This itself is a good enough reason to go beyond the choice of default parameters and search for alternatives. A novel way of using Chebyshev spacing point values as the parameters of Nelder–Mead's algorithm is presented in this article. In comparison, the proposed scheme is found to perform better than the existing schemes. The present work does not dwell on the issue of why Chebyshev spacing point values as parameters perform well in the context of NMA. The question is still open and left for future research. Moreover, there exist cases for which the proposed scheme fails to converge to the optimal solution. It would be interesting to see the effect of using the proposed scheme of parameter selection with some of the convergent variants (Kelley 1999; Tseng 1999; Price, Coope, and Byatt 2002; Bűrmen, Puhan, and Tuma 2006; Bűrmen and Tuma 2009) of the Nelder–Mead algorithm.

Note

1. The choice of this interval is influenced by the range of default values of the parameters as suggested by Nelder and Mead in their original study.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Baudin, Michaël. 2009. *Nelder–Mead User's Manual*. Rocquencourt: Consortium Scilab. https://www.scilab.org/sites/default/files/neldermead.pdf.

Bűrmen, Árpád, Janez Puhan, and Tadej Tuma. 2006. "Grid Restrained Nelder–Mead Algorithm." *Computational Optimization and Applications* 34 (3): 359–375.

Bűrmen, Árpád, and Tadej Tuma. 2009. "Unconstrained Derivative-Free Optimization by Successive Approximation." *Journal of Computational and Applied Mathematics* 223 (1): 62–74.

Dasgupta, Bhaskar. 2006. *Applied Mathematical Methods*. Delhi: Pearson Education.

Dennis, J. E., Jr, and V. Torczon. 1991. "Direct Search Methods on Parallel Machines." *SIAM Journal on Optimization* 1 (4): 448–474.

Fajfar, Iztok, Árpád Bűrmen, and Janez Puhan. 2019. "The Nelder–Mead Simplex Algorithm with Perturbed Centroid for High-Dimensional Function Optimization." *Optimization Letters* 13 (5): 1011–1025.

Fajfar, Iztok, Janez Puhan, and Árpád Bűrmen. 2017. "Evolving a Nelder–Mead Algorithm for Optimization with Genetic Programming." *Evolutionary Computation* 25 (3): 351–373.

Fan, E. 2002. "Global Optimization of Lennard–Jones Atomic Clusters." Master of Science thesis, McMaster University, Hamilton, Ontario, Canada. https://www.mat.univie.ac.at/ neum/glopt/mss/Fan02.pdf.

Gao, Fuchang, and Lixing Han. 2012. "Implementing the Nelder–Mead Simplex Algorithm with Adaptive Parameters." *Computational Optimization and Applications* 51 (1): 259–277.

Han, Lixing, and Michael Neumann. 2006. "Effect of Dimensionality on the Nelder–Mead Simplex Method." *Optimization Methods and Software* 21 (1): 1–16.

Kelley, C. T. 1999. "Detection and Remediation of Stagnation in the Nelder–Mead Algorithm Using a Sufficient Decrease Condition." *SIAM Journal on Optimization* 10 (1): 43–55.

Kumar, G. N. Sashi, and V. K. Suri. 2014. "Multilevel Nelder Mead's Simplex Method." In *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, 1–6. Piscataway, NJ: IEEE. doi:10.1109/ICIINFS.2014.7036549.

McKinnon, Ken I. M. 1998. "Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point." *SIAM Journal on Optimization* 9 (1): 148–158.

Moré, Jorge J., Burton S. Garbow, and Kenneth E. Hillstrom. 1981. "Testing Unconstrained Optimization Software." *ACM Transactions on Mathematical Software* 7 (1): 17–41. doi:10.1145/355934.355936.

Moré, Jorge J., and Stefan M. Wild. 2009. "Benchmarking Derivative-Free Optimization Algorithms." *SIAM Journal on Optimization* 20 (1): 172–191. http://dx.doi.org/10.1137/080724083.

Musafer, H. A., and A. Mahmood. 2018. "Dynamic Hassan Nelder Mead with Simplex Free Selectivity for Unconstrained Optimization." *IEEE Access* 6: 39015–39026.

Nelder, John A., and Roger Mead. 1965. "A Simplex Method for Function Minimization." *The Computer Journal* 7 (4): 308–313.

Price, Christopher John, Ian D. Coope, and David Byatt. 2002. "A Convergent Variant of the Nelder–Mead Algorithm." *Journal of Optimization Theory and Applications* 113 (1): 5–19.

Torczon, Virginia Joanne. 1989. "Multidirectional Search: A Direct Search Algorithm for Parallel Machines." PhD diss., Rice University, Houston, TX.

Tseng, Paul. 1999. "Fortified-Descent Simplicial Search Method: A General Approach." *SIAM Journal on Optimization* 10 (1): 269–288.

Wright, Margaret H. 1996. "Direct Search Methods: Once Scorned, Now Respectable." In *Numerical Analysis: Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis*, 191–208. Harlow, UK: Addison-Wesley.

Wright, Margaret H. 2010. "Nelder, Mead, and the Other Simplex Method." *Documenta Mathematica* 7: 271–276.