Optimization Letters (2019) 13:1011–1025
https://doi.org/10.1007/s11590-018-1306-2

ORIGINAL PAPER

The Nelder–Mead simplex algorithm with perturbed centroid for high-dimensional function optimization

Iztok Fajfar[1] · **Árpád Bűrmen**[1] · **Janez Puhan**[1]

Received: 28 November 2017 / Accepted: 25 July 2018 / Published online: 28 July 2018

Abstract
It is a well known fact that the widely used Nelder–Mead (NM) optimization method soon becomes inefficient as the dimension of the problem grows. It has been observed that part of the reason for the inefficiency is that the search direction becomes practically perpendicular to the direction of the local downhill gradient. In this paper, we identify which operations are responsible for the increase of the angle between the search direction and the direction of the local downhill gradient. We show that it is possible to decrease this angle by randomly perturbing the centroid of the n best vertices around which the original NM algorithm moves the worst vertex. Using a perturbed centroid, the algorithm outperforms the standard NM method as well as the NM method with adaptive parameters for problems with higher dimensions, and works well for problems with dimensions of well over 100.

Keywords Nelder–Mead algorithm · Simplex · Optimization · High-dimensional functions · Search direction

1 Introduction

The Nelder–Mead (NM) simplex algorithm [10] is a numerical method used to find the optimum of an unconstrained objective function in a multidimensional space. Due to its simplicity, it has been used in innumerable optimization problems. At the same time, there is still a lot of open research questions regarding the convergence of the method,

✉ Iztok Fajfar
iztok.fajfar@fe.uni-lj.si

Árpád Bűrmen
arpad.buermen@fe.uni-lj.si

Janez Puhan
janez.puhan@fe.uni-lj.si

[1] I. Fajfar
Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia

whose rigorous analysis seems to be a very hard mathematical problem and only very limited results are known. For example, [6] prove convergence to a minimizer only for strictly convex functions in one dimension, and report various limited convergence results for two dimensions. On the other hand, [7] shows that there exists a family of strictly convex functions in two dimensions for which a certain class of initial simplices converge to a nonminimizer.

Another known convergence issue involves the dimension of the problem. Numerous researchers have reported that the NM algorithm can become very ineffective for high-dimensional problems (see, for example, [5,11,12]). There is, however, not much explanation as to why this is so. [5] show theoretically and numerically that, when using the quadratic function $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{x}$, the convergence of the NM algorithm drastically deteriorates as the dimension increases. This was later remedied to a certain degree by using adaptive parameters that change with the dimension of the problem [4].

In this paper, we build on the observation by [11], which partly explains why the convergence of the NM method becomes intolerably slow for large dimensional problems. Namely, [11] noted that the search direction of the NM algorithm becomes almost orthogonal to the direction of the local downhill gradient as soon as the dimension of the problem exceeds a certain limit. This limit is different for different functions, but it can be as low as eight dimensions.

Today, there has been a lot of research focus put on large-scale optimization of problems of 1000 and more dimensions, mainly using evolutionary algorithms (see, for example, [2,8,13]). However, the NM algorithm continues to stay an extremely popular optimization tool for low-dimensional problems, mainly due to its implementation simplicity and relative efficacy. In 2017 alone, Google Scholar reports more than 3000 articles connected with the NM algorithm. It would therefore be beneficial to extend the usability of the algorithm to the domain of at least moderately dimensioned problems.

In the next section, we briefly summarize the original NM algorithm. Using a quadratic function, we show in Sect. 3 how the angle between the search direction and local downhill gradient increases with increasing dimensionality, and analyze numerically how each of the four NM operations contribute to the observed phenomenon. We propose a simple random perturbation of the centroid in order to prevent the search direction from becoming orthogonal to the direction of the local downhill gradient in Sect. 4. Finally, in Sect. 5, we show how our simple modification immensely improves the convergence of the standard NM method in high dimensions on most of the standard test functions proposed by [9].

2 The original Nelder–Mead method

The NM algorithm is a heuristic direct search method used for solving the unconstrained optimization problem

$$\min f(\mathbf{x}) \text{ subject to } \mathbf{x} \in \mathbb{R}^n, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is the n-dimensional objective function. The algorithm uses an n-dimensional simplex defined by $(n+1)$ vertices $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n+1} \in \mathbb{R}^n$, which is manipulated by iteratively replacing the worst vertex (i.e., the one with the highest objective function value) by a better one. There are five different operations that the algorithm performs, four of which are simple moves in the direction of the centroid of the n best vertices:

$$\mathbf{c} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i . \tag{2}$$

Below, we summarize the version of the NM algorithm as used by [6] and [3]. This version differs from the original method [10] only in that it removes some minor ambiguities present in the original algorithm and does not importantly influence its behavior.

One iteration of the NM algorithm

1. **Sort.** Evaluate f at all the vertices of the simplex and relabel them so that $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \ldots \leq f(\mathbf{x}_{n+1})$.
2. **Reflection.** Calculate the reflected point $\mathbf{x}_r = \mathbf{c} + \alpha(\mathbf{c} - \mathbf{x}_{n+1})$. If $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$, then replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$.
3. **Expansion.** If $f(\mathbf{x}_r) < f(\mathbf{x}_1)$, calculate the expansion point $\mathbf{x}_e = \mathbf{c} + \beta(\mathbf{x}_r - \mathbf{c})$. If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, then replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_e$, else replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$.
4. **Outer Contraction.** If $f(\mathbf{x}_n) \leq f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$, compute the outer contraction point $\mathbf{x}_{oc} = \mathbf{c} + \gamma(\mathbf{x}_r - \mathbf{c})$. If $f(\mathbf{x}_{oc}) \leq f(\mathbf{x}_r)$, replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_{oc}$, else go to step 6.
5. **Inner Contraction.** If $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$, calculate the inner contraction point $\mathbf{x}_{ic} = \mathbf{c} - \gamma(\mathbf{x}_r - \mathbf{c})$. If $f(\mathbf{x}_{ic}) < f(\mathbf{x}_{n+1})$, then replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_{ic}$, else go to step 6.
6. **Shrink.** Replace all points except the best one (i.e., $\mathbf{x}_1$) with $\mathbf{x}_i = \mathbf{x}_1 + \delta(\mathbf{x}_i - \mathbf{x}_1)$.

The originally proposed and most often used values for the algorithm parameters are $\alpha = 1$, $\beta = 2$, $\gamma = 1/2$, and $\delta = 1/2$, which we use in this paper as well.

3 The search direction

All the operations performed by the NM method (with an exception of the shrink operation) replace the worst vertex by a vertex placed along a line connecting the worst vertex and the centroid (2), and the move direction is always from the worst vertex towards the centroid. Because the NM method never uses the shrink operation on the uniformly convex functions [6], and because we will conduct our experiments using quadratic functions, we define the *search direction* of the NM algorithm as the direction of vector $(\mathbf{c} - \mathbf{x}_{n+1})$. Also, we define the *search angle* θ_{search} to be the angle between the search direction $(\mathbf{c} - \mathbf{x}_{n+1})$ and the direction of the local downhill gradient at the worst vertex $(-\bigtriangledown f(\mathbf{x}_{n+1}))$.

In this section, we show experimentally how the search angle θ_{search} approaches 90 degrees as the problem dimension increases, and investigate the contribution of

different NM operations to the observed phenomenon. We used a 5-, 10-, 20-, and 40-dimensional quadratic function $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{x}$ to perform our numeric experiments, and let the standard NM algorithm run from a randomly chosen initial point $\mathbf{x}_0$ with an Euclidean distance of 5 from the origin. The selected initial point $\mathbf{x}_0$ acted as one of the vertices of the initial simplex, while the remaining n vertices were computed as $\mathbf{x}_0 + \tau_i \mathbf{e}_i$, with $\mathbf{e}_i$ being the unit vector in the direction of the ith coordinate. We selected the parameter τ_i as

$$\tau_i = \begin{cases} 0.05 & \text{if } (\mathbf{x}_0)_i \neq 0, \\ 0.00025 & \text{otherwise.} \end{cases} \tag{3}$$

We terminated each run as soon as one of the following four termination criteria was met:

$$\begin{aligned} &\text{(T1) } \max_{2\leq i\leq n+1} |f(\mathbf{x}_i) - f(\mathbf{x}_1)| \leq \texttt{TolFun}. \\ &\text{(T2) } \max_{2\leq i\leq n+1} \|\mathbf{x}_i - \mathbf{x}_1\|_\infty \leq \texttt{TolX}. \\ &\text{(T3) Number of iterations exceeds } \texttt{MaxIter}. \\ &\text{(T4) Number of function evaluations exceeds } \texttt{MaxFunEvals}. \end{aligned} \tag{4}$$

The following termination values were used in our experiments:

$$\texttt{TolFun} = 10^{-14},\ \texttt{TolX} = 10^{-14},\ \texttt{MaxIter} = 10^5,\ \texttt{MaxFunEvals} = 10^6. \tag{5}$$

Figure 1 shows how the search angle changed with the number of iterations. We performed five runs for each of the four used dimensions, which are plotted one over the other. Let us note that all 5- and 10-dimensional runs converged towards values lower than 10^{-14}. Nevertheless, the search angles initially rose towards higher values before eventually falling to smaller values, as seen in the top two plots of Fig. 1. It seems that in either case several iterations are needed in order for the simplex to orient properly, only that the the 10-dimensional runs show much stronger tendency towards 90 degrees and need more iterations to align the simplex properly. The 20- and 40-dimensional runs, however, failed to converge towards the minimum as the search angle was unable to drop sufficiently below 90 degrees. In the 20-dimensional cases, there are still some periods showing bursts of significantly lower angles, but this is not enough for the algorithm to be able to converge towards the minimum. The median of the minimum values of five 20-dimensional runs is only 4.41×10^{-1}. Notice that, even though the algorithm did not arrive to the minimum, it stopped before the maximum number of iterations was reached because (T1) of termination criteria (4) was met. The last plot of Fig. 1 shows the 40-dimensional cases, where the search angle approaches 90 degrees very closely after only 10^4 or so iterations, and even very small decreases of the angle become more and more rare with the increasing number of iterations. Consequently, the median of the final values obtained from five 40-dimensional runs is 45.1.

As the next step, we wanted to see which of the four operations (recall that the shrink operation is never performed on the uniformly convex functions) is most responsible for the unwanted push of the search direction away from the direction of the local

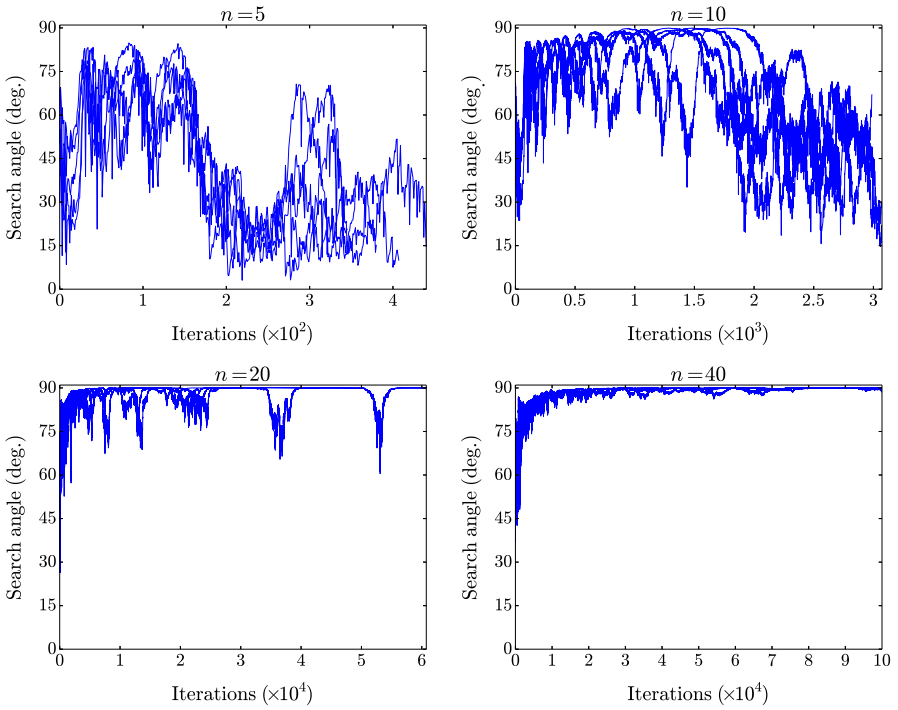

Fig. 1 Change of search angle θ_{search} with the number of iterations for 5-, 10-, 20-, and 40-dimensional quadratic functions

downhill gradient. We used the same parameters as before, and a 40-dimensional quadratic function for this experiment. For each of the iterations of the algorithm, we first identified which of the four operations was actually carried out, and then computed the difference between the search angles of the current and the following iteration. The four thus obtained distributions are shown in Fig. 2, together with the corresponding average values (μ). We can see that reflections cause the largest individual changes of the angle, but their distribution is the most symmetrical one and the average is closest to zero. In fact, the only operation that cumulatively increases the angle is expansion. Of the four operations, the average change caused by expansion is also the only one that is statistically significantly different from 0 (the two-sided p-value of the Wilcoxon rank-sum statistic was 0.0044 for expansion, and greater than 0.5 for the other three operations).

4 Random perturbation of the centroid

Following our observation that it is expansion that is responsible for increasing the search angle towards 90 degrees, we are motivated to change the direction of the move of a vertex caused by an expansion. One way of doing this is to project the original centroid (2) to a random point on a hypersphere centered at the same centroid. In order

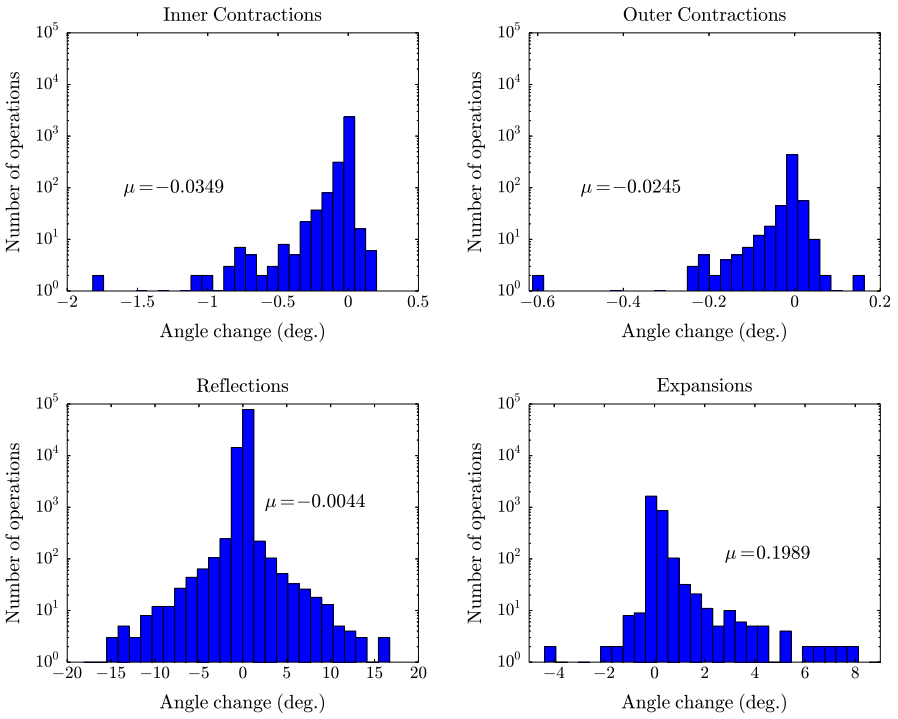

Fig. 2 The histograms show the changes of the search angle caused by individual operations performed by the NM algorithm on a 40-dimensional quadratic function

to ensure that the centroid is projected to a hypersphere with uniform probability, we generate a vector each of whose coordinates is a Gaussian variable [1]. The probability distribution for a vector $\mathbf{v}$ is given by:

$$p(\mathbf{v}) = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{v_1^2+v_2^2+\cdots+v_n^2}{2}}. \tag{6}$$

By experiment, we choose the radius of the hypersphere to be one tenth of the Euclidean distance between the best and the worst vertex, which gives us the perturbation vector

$$\Delta\mathbf{c} = 0.1 \frac{\mathbf{v}}{\|\mathbf{v}\|} \|\mathbf{x}_{n+1} - \mathbf{x}_1\|. \tag{7}$$

Finally, we add the perturbation vector (7) to the original centroid (2) to get a randomly projected centroid whose distribution is uniform over the sphere.

Our first idea was to apply this centroid perturbation only during the expansion operation. However, as the NM method tries out an expansion only after a very successful reflection, it is a good idea to perform a reflection over the same perturbed centroid over which an expansion will be tested. Below, we summarize the changes that we made to the original NM algorithm presented on page 3.

Changes to the original NM algorithm

1. **Centroid Perturbation.** (Inserted after **Sort**) Compute a perturbed centroid $\mathbf{c}' = \mathbf{c} + \Delta\mathbf{c}$.

2. **Reflection and Expansion.** Use $\mathbf{c}'$ instead of $\mathbf{c}$.

To see whether the proposed perturbation indeed effectively decreases the search angle, we performed some more experiments using the same 40-dimensional quadratic function as before. For the first experiment (shown in the left plot of Fig. 3), we saved all the simplices produced during the run of the standard NM algorithm. Then, for each of the saved simplices, we first calculated the search angle of the standard NM method. We then applied one iteration of the standard and one iteration of the perturbed NM method to the same simplex, thus obtaining two slightly different simplices. On each of the two obtained simplices, we again computed the search angles of the standard NM method and then calculated the difference between both. We repeated this procedure 100 times for each of the stored simplices in order to get the average change of the search angle caused by the centroid perturbation. The scatter plot in Fig. 3 shows individual angle changes as a function of the original search angle at which the changes occurred. We see that at angles over 80 degrees the perturbation quite effectively decreases the angle towards the local downhill gradient and this decrease rapidly grows as the search angle approaches 90 degrees.

However, the last experiment does not expose the cumulative effect of the centroid perturbation caused by gradual reorientation of the simplex. To find out how the search angle changes with the number of iterations when using the perturbed centroid, we conducted another experiment similar to the first experiment of Sect. 3, only this time we used a perturbed centroid. The results obtained from five runs using a 40-dimensional quadratic function are shown in the right plot of Fig. 3. Since the angle oscillates with high frequency, the figure shows a 1000-iteration moving average of the angle curves (solid lines) with the corresponding envelopes (dashed lines). As we can see from the plot, the average search angle is quite low and stable, and even its peak values stay below 80 degrees.

We repeated the same experiment also for dimensions up to $n = 160$ using increments of 20 [all the runs ended up with final values below 10^{-13} using termination

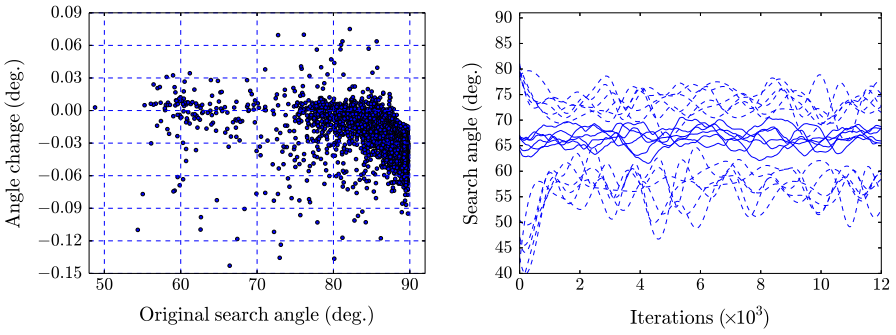

Fig. 3 A scatter plot of individual changes of the search angle caused by perturbing the centroid (left), and average search angle curves with the corresponding envelopes as functions of the number of iterations when using the perturbed centroid (right). Both plots were obtained using a 40-dimensional quadratic function

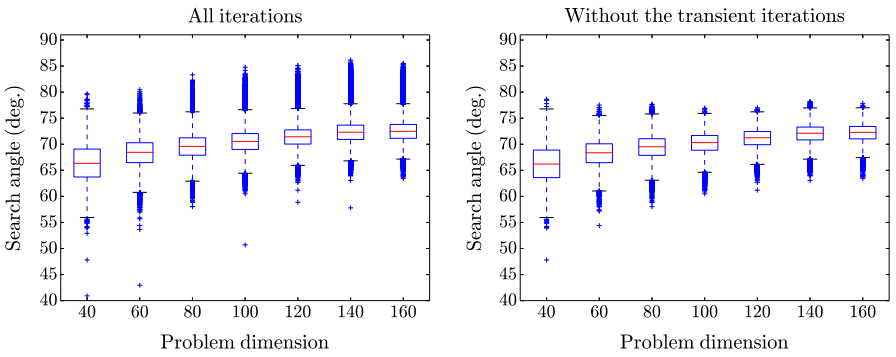

Fig. 4 Box plots of search angles observed during runs of the NM algorithm with a perturbed centroid

values (5)]. On the left of Fig. 4 there are box plots of the search angles observed during each of the runs. The average angle rises only slightly with the problem dimension and stays below 75 degrees even at $n = 160$. We can see that there are many outliers, particularly above the upper whiskers, which are placed at 1.5 IQR above the upper quartiles. The right plot of Fig. 4 shows the same data, only without the first 10% of iterations. It is interesting to observe that most of the outliers above the upper whiskers have disappeared. This implies that the centroid perturbation exerts also an additional cumulative effect on the simplex orientation that, in turn, decreases the search angle even more.

5 Results

In order to demonstrate the benefit of our proposal, we compare the NM method using the perturbed centroid (PNM) to the standard NM algorithm (SNM) and the NM algorithm with adaptive parameters for high-dimensional problems proposed by [4] (ANM). All the tests were run from the initial simplex formed according to (3) from a test-function-specific initial point. We used the following termination values in the experiments that follow:

$$\texttt{TolFun} = 10^{-8},\ \texttt{TolX} = 10^{-8},\ \texttt{MaxIter} = 10^{6},\ \texttt{MaxFunEvals} = 10^{6} \quad (8)$$

Table 1 shows the results of minimizing the modified version of quartic function from [4]:

$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{D}\mathbf{x} + \sigma(\mathbf{x}^{\mathrm{T}}\mathbf{B}\mathbf{x})^2, \quad (9)$$

where $\mathbf{D}$ is a positive definite matrix of the form

$$\mathbf{D} = \mathrm{diag}\left(\left[1+\epsilon, (1+\epsilon)^2, \ldots, (1+\epsilon)^n\right]\right) \quad (10)$$

Table 1 Comparison of standard NM (SNM), NM with adaptive parameters (ANM), and NM with perturbed centroid (PNM) using quartic problem (9) ($\epsilon = 0.05, \sigma = 0.0001$)

Dimension	SNM ANM		PNM (worst) PNM (best)	
	Fun. evals.	Final value	Fun. evals.	Final value
10	1799	2.84384×10^{-8}	1284	3.52171×10^{-7}
	1048	1.46193×10^{-8}	1151	5.70761×10^{-9}
20	21,290	1.29843×10^{1}	3818	1.26812×10^{-7}
	3507	1.81697×10^{-8}	4075	2.98315×10^{-8}
30	40,121	1.81080×10^{2}	7083	1.06510×10^{-7}
	8787	2.31123×10^{-8}	6933	1.85278×10^{-8}
40	68,271	4.38271×10^{2}	11,542	8.16516×10^{-8}
	19,851	1.72662×10^{-8}	11,760	4.61390×10^{-8}
50	120,990	7.89692×10^{2}	16,701	1.43112×10^{-7}
	37,299	3.20972×10^{-8}	18,565	5.88386×10^{-8}
60	154,326	1.49980×10^{3}	26166	1.48991×10^{-7}
	57,168	3.45578×10^{-8}	27,369	6.79145×10^{-8}
80	337,257	1.03679×10^{4}	56,032	2.16625×10^{-7}
	197,860	1.07427×10^{-7}	49,939	1.21469×10^{-7}
100	592,112	1.35030×10^{4}	89,355	2.92879×10^{-7}
	368,568	2.63321×10^{-7}	98,709	2.14714×10^{-7}
120	912,282	4.77857×10^{4}	192,329	5.47039×10^{-7}
	10^6	1.69050×10^{-4}	182,553	4.00351×10^{-7}
140	10^6	6.38875×10^{4}	378,634	1.41549×10^{-6}
	10^6	2.52566×10^{1}	391,995	9.02658×10^{-7}
160	10^6	3.67150×10^{4}	793,294	2.49111×10^{-6}
	10^6	1.19570×10^{3}	824,775	1.56876×10^{-6}

and $\mathbf{B}$ a positive definite matrix of the form

$$\mathbf{B} = \mathbf{U}^{\mathrm{T}}\mathbf{U}, \quad \mathbf{U} = \begin{bmatrix} 1 & \cdots & 1 \\ & \ddots & \vdots \\ & & 1 \end{bmatrix}, \tag{11}$$

with parameters $\epsilon \geq 0$ and $\sigma \geq 0$, which control the condition number of $\mathbf{D}$ and deviation from quadratic, respectively. The used starting point was $[1, 1, \ldots, 1]^{\mathrm{T}}$.

For comparison purposes, Table 1 shows the number of function evaluations instead of iterations, because the former is a much more appropriate measure of the algorithm performance. Because the PNM method uses a random parameter (i.e., the direction of the perturbation), we applied the method 10 times for each of the dimensions, and we report only the results of the best and worst of each of the 10 runs.

Table 2 Comparison of standard NM (SNM), NM with adaptive parameters (ANM), and NM with perturbed centroid (PNM) using a set of standard test problems from [9]

Problem (dimension)	SNM ANM		PNM (worst) PNM (best)	
	Fun. evals.	Final value	Fun. evals.	Final value
Extended Rosenbrock (40)	89,050	3.78542×10^{1}	253,479	5.93614×10^{-6}
	91,746	4.25015×10^{-8}	238,398	3.89228×10^{-6}
Extended Rosenbrock (100)	912,106	6.05215×10^{1}	10^{6}	5.41668×10^{-5}
	10^{6}	1.95757×10^{-3}	10^{6}	4.92298×10^{-5}
Extended Rosenbrock (160)	10^{6}	7.81640×10^{1}	10^{6}	4.21502×10^{-2}
	10^{6}	5.38003	10^{6}	4.16270×10^{-2}
Extended Powell (40)	66,638	2.41228×10^{-3}	135,615	2.18268×10^{-5}
	26,275	1.13174×10^{-5}	138,399	1.70210×10^{-5}
Extended Powell (100)	765,580	7.75203×10^{-2}	707,221	3.81446×10^{-5}
	241,038	9.83607×10^{-5}	743,291	2.87869×10^{-5}
Extended Powell (160)	10^{6}	2.39227×10^{-2}	10^{6}	1.44137×10^{-4}
	818,704	2.37300×10^{-4}	10^{6}	1.37588×10^{-4}
Penalty I (40)	250,223	4.06726×10^{7}	22,709	3.40204×10^{-4}
	48,058	3.40450×10^{-4}	26,036	3.39528×10^{-4}
Penalty I (100)	10^{6}	4.76691×10^{10}	60,273	9.03086×10^{-4}
	582,205	9.10370×10^{-4}	67,108	9.02776×10^{-4}
Penalty I (160)	10^{6}	1.17131×10^{12}	114,009	1.47635×10^{-3}
	10^{6}	4.24852×10^{9}	121,057	1.47627×10^{-3}
Penalty II (40)	96,871	5.60319×10^{-1}	58,249	5.56913×10^{-1}
	23,032	5.56912×10^{-1}	38,375	5.56913×10^{-1}

Table 2 continued

Problem (dimension)	SNM ANM		PNM (worst) PNM (best)	
	Fun. evals.	Final value	Fun. evals.	Final value
Penalty II (100)	10^6	9.70961×10^4	145,731	9.70961×10^4
	107,788	9.70961×10^4	89,539	9.70961×10^4
Penalty II (160)	890,949	1.58057×10^{10}	457,793	1.58057×10^{10}
	484,830	1.58057×10^{10}	362,876	1.58057×10^{10}
Variably dimensioned (40)	44,698	4.05685×10^1	166,142	2.48604×10^{-6}
	110,866	2.21522×10^{-8}	162,973	1.74656×10^{-6}
Variably dimensioned (100)	282,992	1.95651×10^2	10^6	1.07760×10^1
	10^6	2.11314	10^6	5.39746
Variably dimensioned (160)	10^6	2.99588×10^2	10^6	6.08574×10^1
	10^6	1.32868×10^2	10^6	3.57239×10^1
Trigonometric (40)	10,533	4.38167×10^{-6}	8946	6.74439×10^{-6}
	16,410	1.76046×10^{-5}	11,180	1.14528×10^{-6}
Trigonometric (100)	36,109	3.61193×10^{-6}	27,122	2.09156×10^{-6}
	65,307	1.57336×10^{-6}	38,763	6.30426×10^{-7}
Trigonometric (160)	40,896	1.74094×10^{-6}	42,063	1.47542×10^{-6}
	153,553	6.56157×10^{-7}	46,881	7.65360×10^{-7}
Brown almost-linear (40)	25,682	1.87784×10^2	230,487	2.09334×10^{-5}
	54,665	5.23140×10^{-8}	19,754	2.81350×10^{-7}
Brown almost-linear (100)	39,476	1.89905×10^3	230,487	2.09333×10^{-5}
	857,806	6.85578×10^{-6}	237,803	7.11428×10^{-7}

Table 2 continued

Problem (dimension)	SNM ANM		PNM (worst) PNM (best)	
	Fun. evals.	Final value	Fun. evals.	Final value
Brown almost-linear (160)	134,438	4.42163×10^{3}	10^{6}	1.73659×10^{-4}
	10^{6}	5.26067×10^{-1}	10^{6}	6.47152×10^{-6}
Discrete boundary value (40)	367	1.78029×10^{-5}	405	1.78029×10^{-5}
	773	1.78029×10^{-5}	383	1.78029×10^{-5}
Discrete boundary value (100)	907	1.23293×10^{-6}	980	1.23293×10^{-6}
	1913	1.23293×10^{-6}	957	1.23293×10^{-6}
Discrete boundary value (160)	1447	3.07091×10^{-7}	1553	3.07091×10^{-7}
	3053	3.07091×10^{-7}	1523	3.07091×10^{-7}
Discrete integral equation (40)	8134	1.03767×10^{-7}	6941	9.78014×10^{-8}
	6258	1.33401×10^{-8}	7205	3.45665×10^{-8}
Discrete integral equation (100)	35,314	8.42059×10^{-8}	24,541	8.18965×10^{-8}
	42,636	1.73958×10^{-8}	24,069	5.65734×10^{-8}
Discrete integral equation (160)	65,990	1.68068×10^{-7}	48,031	1.04347×10^{-7}
	128,947	2.88308×10^{-8}	51,057	7.97836×10^{-8}
Broyden tridiagonal (40)	17,517	1.29651×10^{-6}	13,338	1.74036×10^{-7}
	22,905	1.62885×10^{-8}	11,919	5.84409×10^{-8}
Broyden tridiagonal (100)	107,826	1.37883×10^{-7}	54,142	1.69908×10^{-7}
	235,282	2.80010×10^{-8}	51,669	1.16232×10^{-7}
Broyden tridiagonal (160)	359,635	9.65513×10^{-8}	98,826	1.99303×10^{-7}
	735,751	3.93550×10^{-8}	104,368	1.15375×10^{-7}

Table 2 continued

Problem (dimension)	SNM ANM		PNM (worst) PNM (best)	
	Fun. evals.	Final value	Fun. evals.	Final value
Broyden banded (40)	77,017	1.64081×10^{-3}	11,102	1.10475×10^{-7}
	6932	1.47517×10^{-8}	11,631	3.58933×10^{-8}
Broyden banded (100)	206,941	1.68527×10^{-5}	46,371	1.16127×10^{-7}
	37,171	2.18768×10^{-8}	42,636	6.57585×10^{-8}
Broyden banded (160)	192,647	1.36203×10^{-7}	96,450	1.34500×10^{-7}
	10^6	3.86459×10^{-3}	83,881	9.33087×10^{-8}
Linear—full rank (40)	84,595	5.52713×10^{-1}	10,748	1.01770×10^{-7}
	8186	1.80976×10^{-8}	10,754	3.09716×10^{-8}
Linear—full rank (100)	10^6	2.46723×10^{-3}	36,296	9.16071×10^{-8}
	39,686	2.51733×10^{-8}	38,208	6.97729×10^{-8}
Linear—full rank (160)	10^6	6.24133×10^{-5}	72,646	1.32256×10^{-7}
	97,098	1.74354×10^{-8}	76,738	7.49229×10^{-8}
Linear—rank 1 (40)	3035	9.62963	2742	9.62963
	7091	9.62963	2648	9.62963
Linear—rank 1 (100)	11,288	2.46269×10^{1}	9011	2.46269×10^{1}
	29,313	2.46269×10^{1}	8526	2.46269×10^{1}
Linear—rank 1 (160)	23,489	3.96262×10^{1}	16,012	3.96262×10^{1}
	64,631	3.96262×10^{1}	15,574	3.96262×10^{1}

We can see that the SNM method fails already at $n = 20$. The PNM method is comparable to the ANM method for dimensions from $n = 10$ to $n = 30$, but, already at $n = 40$, the PNM method is almost two times faster in terms of number of function evaluations. The PNM method works well even for dimensions well over 100, while the ANM method starts to deteriorate at these dimensions quite seriously.

We conducted the next set of experiments on the standard test function set proposed by [9]. Note that some of the functions in the set are not defined for larger dimensions, so we omitted those functions from the experiment. Table 2 summarizes the results. Out of 13 test functions, the PNM method performs worse than the ANM method on seven functions at 40 dimensions, but it performs worse than the SNM method only on the Discrete boundary value function. It is, however, interesting that the SNM method performs the best of all three methods, and the ANM method the worst of all three methods on this function at all dimensions. At 100 dimensions, the PNM method performs slightly worse than the ANM method only on Extended Powell, Penalty II, Variably dimensioned, and Broyden banded functions. At the same time, we can see the PNM method performing better than the ANM method on Extended Rosenbrock, Penalty I, Trigonometric, Brown almost linear, Discrete boundary value, Discrete integral equation, Broyden tridiagonal, and Linear—rank 1 functions. Notice that in some cases the reached final value is just slightly worse for the PNM method, but the number of function evaluations is many times smaller. By decreasing the termination tolerances in (8), we mostly got better final values in these cases, while still getting much smaller number of function evaluations.

Moving to the 160-dimensional cases, we see that the PNM method is better than the ANM method on all but the Extended Powell function. But, even here, the ANM method is only 20% better regarding the number of performed function evaluations, while its final value is still slightly worse than that of the PNM method.

As a final remark, we would like to note that we carried out quite some experiments with the radius of the projection hypersphere different from 0.1 (see Eq. (7)), but did not observe any significant improvement. It seems that this value of 0.1 is close to optimal independently of the problem dimension. We even tried to combine adaptive coefficients of [4] with the centroid perturbation, varying adaptive parameters as well as the projection hypersphere radius. However, neither of both independent approaches profited from the combination.

6 Conclusion

We proposed a random perturbation of the centroid during the reflection and expansion operations of the standard NM algorithm. Numerical experiments show that this perturbation cumulatively corrects the angle of the search direction, which becomes almost perpendicular to the local downhill gradient in the standard NM method at higher dimensions. Further experiments using a standard set of test functions show that our method is much better than the standard NM method, and comparable to the adaptive NM method, for 40-dimensional problems. As the problem dimension increases, our method is becoming more and more superior to the other two methods. A great advantage of the proposed adaptation is that it requires only a minor pro-

gramming intervention in the existing optimization software. This is important as the NM algorithm is still one of the most often used direct search methods for moderately dimensioned problems. Because there are cases—mainly of dimensions below 100—where the adaptive NM method performs better, we suggest that our method be used as an addition rather than a replacement in software tools utilizing the NM optimization algorithm.

Acknowledgements This work was supported by the Ministry of Education, Science and Sport of Republic of Slovenia under Research program P2-0246—Algorithms and optimization methods in telecommunications.

References

1. Blum, A., Hopcroft, J., Kannan, R.: Foundations of Data Science. Unpublished (2017) https://www.cs.cornell.edu/jeh/book.pdf
2. Duarte, A., Martí, R., Gortazar, F.: Path relinking for large-scale global optimization. Soft Comput. **15**(11), 2257–2273 (2011). https://doi.org/10.1007/s00500-010-0650-7
3. Fajfar, I., Puhan, J., Bürmen, Á.: Evolving a nelder-mead algorithm for optimization with genetic programming. Evolut. Comput. **25**(3), 351–373 (2017). https://doi.org/10.1162/evco_a_00174
4. Gao, F., Han, L.: Implementing the Nelder–Mead simplex algorithm with adaptive parameters. Comput. Optim. Appl. **51**(1), 259–277 (2012). https://doi.org/10.1007/s10589-010-9329-3
5. Han, L., Neumann, M.: Effect of dimensionality on the Nelder–Mead simplex method. Optim. Methods Softw. **21**(1), 1–16 (2006). https://doi.org/10.1080/10556780512331318290
6. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder–Mead simplex method in low dimensions. SIAM J. Optim. **9**(1), 112–147 (1998)
7. McKinnon, K.I.M.: Convergence of the Nelder–Mead simplex method to a nonstationary point. SIAM J. Optim. **9**(1), 148–158 (1998)
8. Mohapatra, P., Nath Das, K., Roy, S.: A modified competitive swarm optimizer for large scale optimization problems. Appl. Soft Comput. **59**(C), 340–362 (2017). https://doi.org/10.1016/j.asoc.2017.05.060
9. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
10. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)
11. Torczon, V.J.: Multi-directional search: A direct search algorithm for parallel machines. Ph.D. thesis, Rice University, Houston, TX (1989)
12. Wright, M.H.: Direct search methods: once scorned, now respectable. In: Griffiths DF, Watson GA (eds) Numerical Analysis 1995, In: Proceedings of the 1995 dundee biennial conference in numerical analysis, CRC Press, Boca Raton, FL, Pitman Research Notes in Mathematics, vol 344, pp. 191–208 (1996). http://www.crcpress.com/shopping_cart/products/product_detail.asp?sku=LM7633&parent_id=&pc=/shopping_cart/search/search.asp!
13. Yang, P., Tang, K., Yao, X.: Turning high-dimensional optimization into computationally expensive optimization. IEEE Trans. Evolut. Comput. **22**(1), 143–156 (2018). https://doi.org/10.1109/TEVC.2017.2672689