Yugoslav Journal of Operations Research
28 (2018), Number 2, 153–169
DOI: https://doi.org/10.2298/YJOR180120014G

LESS IS MORE: SIMPLIFIED NELDER-MEAD METHOD FOR LARGE UNCONSTRAINED OPTIMIZATION

Kayo GONÇALVES-E-SILVA
Digital Metropolis Institute, Universidade Federal do Rio Grande do Norte, Natal, Brazil
kayo@imd.ufrn.br

Daniel ALOISE
Department of Computer and Software Engineering, École Polytechnique de Montral, Montréal, Canada
daniel.aloise@polymtl.ca

Samuel XAVIER-DE-SOUZA
Department of Computation and Automation, Universidade Federal do Rio Grande do Norte, Natal, Brazil
samuel@dca.ufrn.br

Nenad MLADENOVIĆ
Mathematical Instituite, Serbian Academy of Sciences and Arts, Belgrade, Serbia
nenad@mi.sanu.ac.rs

Received: January 2018 / Accepted: March 2018

Abstract: Nelder-Mead method (NM) for solving continuous non-linear optimization problem is probably the most cited and the most used method in the optimization literature and in practical applications, too. It belongs to the direct search methods, those which do not use the first and the second order derivatives. The popularity of NM is based on its simplicity. In this paper we propose even more simple algorithm for larger instances that follows NM idea. We call it Simplified NM (SNM): instead of generating all $n + 1$ simplex points in $\mathcal{R}^n$, we perform search using just $q + 1$ vertices, where q is usually much smaller than n. Though the results cannot be better than after performing calculations in $n+1$ points as in NM, significant speed-up allows to run many times SNM from different starting solutions, usually getting better results than those obtained by NM within the same cpu time. Computational analysis is performed on 10 classical con-

vex and non-convex instances, where the number of variables n can be arbitrarily large. The obtained results show that SNM is more effective than the original NM, confirming that LIMA yields good results when solving a continuous optimization problem.

Keywords: Continuous Optimization, Direct Search Methods, Nelder Mead Method, Less is More Approach.

MSC: 90C30, 90C56, 90C59.

1. INTRODUCTION

Let us consider the following deterministic continuous optimization problem

$$\min\{f(x)|x \in X, X \subseteq \mathcal{R}^n\}, \tag{1}$$

where $\mathcal{R}^n, X, x$, and f respectively denote the *Euclidean n-dimensional space*, a *feasible set*, a *feasible solution*, and a real-valued *objective function*. A solution $x^* \in X$ is *optimal* if $f(x^*) \leq f(x)$, $\forall x \in X$. If $X = \mathcal{R}^n$, the unconstrained problem is defined. The problem is also classified as unconstrained if a feasible set X is n-dimensional hypercube, and thus represented by box constraints, i.e., $X = \{x = (x_1, .., x_j, .., x_n) \in \mathcal{R}^n \mid a_j \leq x_j \leq b_j\}$.

Optimization methods developed for solving (1) mainly depend on the properties of the objective function $f(x)$. A nonlinear objective function could be convex or non-convex, continuous or not, derivable or not, smooth or not, etc.

Nelder-Mead.. Nelder-Mead (NM) method was originally designed for solving convex non-differentiable unconstrained nonlinear optimization problems [14]. It belongs to *direct search* methods, those that use only values of the objective function in searching for the optimal solution. In NM method, $n+1$ points are constructed in each iteration, forming a simplex in $\mathcal{R}^n$. Estimation of the gradient is made as a direction that connects the worst among $n+1$ points and the centroid of the remaining n points. Four possible points are tried out along that direction, following the strict order: reflection, expansion, outer contraction, and inner contraction. The first improvement strategy is implemented: a move is made immediately after a better than the worst point is obtained; a new simplex is made by replacing the worst point by the new one; all other n points are unchanged. If none of the four points along the estimated gradient direction is better, than only the best point remains the same in the next iteration, and the rest n are moved along the diretion of their position towards the best simplex point. In that way, the volume of the simplex is drastically reduced. This fact is used for a stopping condition: stop when the simplex volume is less than some arbirtrary small number ε.

Literature review.. The NM paper has been cited thousands of times. It was qualified by the late 1970s as a Science Citation Classic. Even recently, Google Scholar displayed more than 2,000 papers published in 2012 referring to the NM method, sometimes combining it with other algorithms. During the last 50 years, many modifications of the NM method have been proposed in the literature. Most

of them add new rules and steps to improve the solution quality and to satisfy some convergence conditions (for various modifications of NM method see, for example, [17, 5]). Besides, there are also the attempts to simplify it; for example, a parameter free version, proposed in [22]. There, parameters of the method (for reflection, expansion, etc) are used at random from the given intervals.

In recent years, NM has been used for solving non-convex and non-differentiable optimization problems, mostly as a local search routine within more complex algorithms, such as metaheuristics (see e.g., [11, 9, 1] where it is used within Variable neighborhood search metaheuristic). In [21], NM was extended for solving continuous global optimization problems. It is called Restarted Modified NM (RMNM). The basic idea of (RMNM) is equally simple as the idea of NM method. When NM stops, a new initial simplex is constructed around the final point and search continues until there is no improvement in two successive big itertions.Thus, each restart of full NM can be seen as one iteration of RMNM. Similar idea is also implemented in solving non-differentiable optimization problems in [1].

Less is more approach.. Less Is More Approach (LIMA) for solving optimization problems has recently been proposed in [12]. Its main idea is to find a minimum number of search ingredients when numerically solving some particular problem. Namely, if LIMA based method simplifies a current popular and much used method, it should provide better quality solutions than the original method within the same running time. Thus, the objective is to make a method as simple as possible but, at the same time, more effective and efficient than the current state-of-the-art heuristic. In this paper we propose Simplified Nelder-Mead (SNM) method.

Motivation and Contribution.. As mentioned earlier, many metaheuristic methods have been designed for solving continuous global optimization problems. In the recent empirical study [6], around 20 metaheuristic based methods were compared on the usually used test instances (such as Ackley function, Shekel function, Rastrigin function etc.), where the dimension n can be arbitrarily increased. It appeared that the best results were reported by heuristics based on Differential Evolution (DE). On the other hand, DE based heuristics use only a few points during the search, even when the size of the problem is very large. Taking those empirical results into account, the following conclusions can be drawn: (i) we do not need $n + 1$ points to find good search direction in $\mathcal{R}^n$ as NM method does, i.e., 4 or 5 points could be enough; (ii) a better direction than that defined in NM probably cannot be found, despite of the fact that updating of mutation and cross-over parameters within DE could be succesful [7]. Thus, we got an idea to reduce the number of points in direct search method from DE based methods, although DE methods have nothing in common with NM and SNM. They just use, as SNM does, only a few points in finding discent direction.

Based on observations (i) and (ii) from above, we propose a simplification of the NM optimization method that we call Simplified NM. It simply apply NM method in R^4, where in each iteration, four out of n dimensions ($n \geq 4$) are

chosen at random. SNM is restarted until the allowed cpu time t_{max} ellapses. Note that t_{max} is defined to be equal to the time used by NM, before it stops naturally. The extensive computational results show that SNM performs better than NM, confirming that LIMA approach yelds good results in solving continuous optimization problem.

Outline.. The rest of the paper is organized as follows. In section 2, we briefly review NM and its simplified version. Section 3 provides extensive computational results regarding the usual optimization problems, both convex and non-convex. Section 4 concludes the paper.

2. SIMPLIFIED NELDER MEAD METHOD FOR UNCONSTRAINED OPTIMIZATION

SNM method is a NM that uses simplex with $q+1$ vertices in $\mathcal{R}^n$ instead of $n+1$, where $q < n$. Any of the numerious existing modifications of NM could easily have its "simplified" version. In this paper, we propose a simplification of the original NM.

The NM method was proposed for minimizing a real-valued function $f(\mathbf{x})$ for $\mathbf{x} \leq \mathcal{R}^n$ [14]. The method is able to change direction during its execution to adapt itself to a local landscape without using derivatives. Four scalar values have to be defined to execute the Nelder-Mead algorithm: coefficients of reflection (α), expansion (β), contraction (γ), and shrinkage (δ). According to the original Nelder-Mead paper [14], these parameters should satisfy $\alpha > 0$, $\beta > 1$, $0 < \gamma < 1$ and $0 < \delta < 1$. The choices used in the standard Nelder-Mead algorithm are $\alpha = 1$, $\beta = 2$, $\gamma = 1/2$ and $\delta = 1/2$. Discussions about these parameters can be found in [19**?**].

Initially, a nondegenerate initial simplex $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n+1}\}$ is given, such that $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \ldots \leq f(\mathbf{x}_{n+1})$, where $\mathbf{x}_1$ and $\mathbf{x}_{n+1}$ are know as the best and the worst vertices/solutions/points, respectively. According to [21], due to lack of information, it is customary to specify a random starting point $\mathbf{x}$ in $\mathcal{R}^n$ and generate other n vertices by the perturbation of $\mathbf{x}$ along the n coordinate directions. The Algorithm 1 shows an embodiment of this disturbance with step $\tau m \mathbf{e}_i$, where τ is a parameter with the predefined value, $\mathbf{e}_i$ the n-dimensional unit vector with one in the ith component (and zeros elsewhere), and m represents the largest absolute value among the coordinates of the starting point $\mathbf{x}$. If not feasible, the vertex has to be projected onto the hypercube defined by box constraints: $H = \{\mathbf{x} \mid \mathbf{a}_j \leq \mathbf{x}_j \leq \mathbf{b}_j\}$. In other words, if some $\mathbf{x}_j$ is greater than $\mathbf{b}_j$ or smaller than $\mathbf{a}_j$, it becomes equal to $\mathbf{b}_j$ or $\mathbf{a}_j$, respectively.

The main loop of NM method consists of consecutive runs of its one iteration. One iteration is composed of an ordering, reflection, expansion, contractions (inside and outside), and shrinkage steps. An illustration of the basic NM steps in 2 dimensions is given in Figure 1.

Algorithm 1 Initial Simplex of the Nelder-Mead Method.

procedure INITIAL-SIMPLEX($\mathbf{x}, n$)
  $m \leftarrow$ `Max coordinate`($\mathbf{x}$); Choose τ;
  if $m = 0$ **then** $m \leftarrow 1$;
  $\mathbf{x}_1 \leftarrow \mathbf{x}$;
  for $i = 2$ **to** $n + 1$ **do**
    $\mathbf{x}_i \leftarrow \mathbf{x} + \tau \cdot m \cdot \mathbf{e}_i$;
    if ($\mathbf{x_i}$ not feasible) **then** (project $\mathbf{x}_i$ onto the hypercube); **endif**
  end for
  return $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n+1}\}$;
end procedure

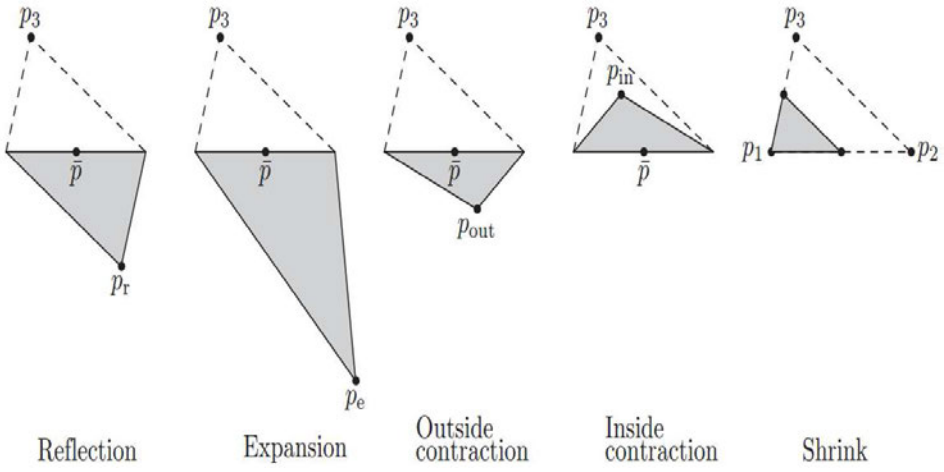

Figure 1: Graphic Representation of the Nelder-Mead method [20].

Details of one NM iteration is given in Algorithm 2. "A vertice is accepted" in Algorithm 2 means that it replaces the worst vertice of $\mathbf{X}$.

The NM method is performed by execution of an initial simplex and followed by several runs of one iteration, as shown in Algorithm 3. A stopping condition in most implementations of the NM method is based on two criteria: either the function values at the vertices are sufficiently close, or the volume of simplex becomes very small [21]. Since the method does not always converge (see e.g. [8, 10, 15]), the stopping criterion is the combination of the number of consecutive iterations without improvement it_{max} and the following condition:

$$\frac{2 \cdot |f(\mathbf{x}_{n+1}) - f(\mathbf{x}_1)|}{|f(\mathbf{x}_{n+1})| + |f(\mathbf{x}_1)| + \epsilon} \leq \epsilon \tag{2}$$

where $\epsilon > 0$. For discussions about the convergence, see also [4, 13, 15, 16].

Algorithm 2 One iteration of the Nelder-Mead Method.

procedure NM-ITERATION($\mathbf{X}, f, n$)
  Ordering. Order the $n+1$ vertices from $\mathbf{X}$ such that $f(\mathbf{x}_1) \leq \ldots \leq f(\mathbf{x}_{n+1})$.
  Centroid. Compute the centroid of the n best points $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$.
  Reflection. Compute the *reflection point* $\mathbf{x}_r = \bar{x} + \alpha(\bar{x} - \mathbf{x}_{n+1})$.
    If $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$, accept the reflection point $\mathbf{x}_r$ and terminate the iteration.
  Expansion. If $f(\mathbf{x}_r) < f(\mathbf{x}_1)$, compute the *expansion point* $\mathbf{x}_e = \bar{x} + \beta(\mathbf{x}_r - \bar{x})$.
    If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$, accept the expansion point $\mathbf{x}_e$ and terminate the interation;
    otherwise accept $\mathbf{x}_r$ and terminate the iteration.
  Contraction. Here it is certain that $f(\mathbf{x}_r) \geq f(\mathbf{x}_n)$.
    (a) Outside. If $f(\mathbf{x}_r) < f(\mathbf{x}_{n+1})$, then compute *outside contraction point*
    $\mathbf{x}_c = \bar{x} + \gamma(\mathbf{x}_r - \bar{x})$. If $f(\mathbf{x}_c) \leq f(\mathbf{x}_r)$, accept $\mathbf{x}_c$ and terminate the iteration.
    (b) Inside. If $f(\mathbf{x}_r) \geq f(\mathbf{x}_{n+1})$, then compute *inside contraction point*
    $\mathbf{x}_c = \bar{x} - \gamma(\mathbf{x}_r - \bar{x})$. If $f(\mathbf{x}_c) \leq f(\mathbf{x}_{n+1})$, accept $\mathbf{x}_c$ and terminate.
  Shrinkage. Replace all points, except $\mathbf{x_1}$, with $\mathbf{x}_i = \mathbf{x}_1 + \delta(\mathbf{x}_i - \mathbf{x}_1)$ for all $i > 1$.
end procedure

Algorithm 3 Nelder Mead Method.

procedure NELDER-MEAD(f, n)
  Get an initial vertex $\mathbf{x} \in \mathcal{R}^n$ at random;
  $\mathbf{X} \leftarrow$ `Initial-Simplex`$(\mathbf{x}, n)$;
  while convergence criterion **do**
    `NM-Iteration`$(\mathbf{X}, f, n)$;
  end while
  return x_1;
end procedure

2.1. Simplified Nelder-Mead

In the original NM method and in all of its numerius modifications, the constructed simplex contains $n+1$ vertices. In our SNM, size of the simplex is simply reduced. Besides the problem size n, the SNM contains parameter q ($q < n$) that a user can choose before running the code. In order to check if LIMA "works", we took a value of $q = 4$ for all problem size tested, i.e., the number of simplex vertices, we use in Computational results section, is equal to $q + 1 = 5$.

It is clear that the results obtained with SNM cannot be better than those

obtained by NM: estimated gradient direction derived from $n+1$ points is more precise than the gradient estimated from five points. However, calculation of objective function values in $n+1$ points can last very long and thus, moving through the solution space with only five points should be much faster. If the stopping condition for NM and SNM is based on running time, then the question is: would a multistart SNM be better within the same cpu time? Since the simplified procedure is obvious, we will not give its multi-start pseudo-code. It would be a repetition of all other algorithms given earlier, but using parameter q instead of n, for the number of points in $\mathcal{R}^n$. In Algorithm 1 for example, input parameters (x, n) in `Initial-Simplex`(x, n), should be `Initial-SimplexS`(x, n, q). In line 5 of Algorithm 1, q values of i are taken at random from $[1, n]$. So, we get set $\mathbf{X}$ that contains $q+1$ points (see Algorithm 4).

Algorithm 4 Initial Simplex of the SNM

1: **procedure** INITIAL-SIMPLEXS($\mathbf{x}, n, q$)
2: $m \leftarrow$ `Max coordinate`$(\mathbf{x})$; Choose τ;
3: **if** $m = 0$ **then** $m \leftarrow 1$;
4: $\mathbf{x}_1 \leftarrow \mathbf{x}$;
5: **for** $i = 2$ **to** $q + 1$ **do**
6: $j \leftarrow Rand[1, n]$;
7: $\mathbf{x}_i \leftarrow \mathbf{x} + \tau \cdot m \cdot \mathbf{e}_j$;
8: **if** ($\mathbf{x_i}$ not feasible) **then** (project $\mathbf{x}_i$ onto the hypercube); **endif**
9: **end for**
10: **return** $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{q+1}\}$;
11: **end procedure**

Adaptation of Algorithm 2 is also obvious. We order $q+1$ points, find centroid of q points, reflect worst among q, etc.

3. COMPUTATIONAL EXPERIMENTS

In this section we analyze the quality of the results obtained by NM and SNM. Both methods are coded in C++ and executed on a supercomputer with 64 nodes, each one with 2 CPUs Intel Xeon Sixteen-Core E5-2698v3 2.3 GHz and 128 GB of RAM DDR4, located in High Performance Computing Center at UFRN (NPAD/UFRN), Natal, Rio Grande do Norte, Brazil. Although running on a high-performance computer, the codes were not implemented on parallel. This computer was used just to run multiple instances of the problem at the same time in order to reduce the total running time.

The algorithms are compared on test instances listed in Table 1 that contains 10 benchmark functions, which represent well the diversity in characteristics of difficulties arising in global optimization problems [21]. The name and the abbreviation of each function are given in the first two columns. The third column gives the known minimum objective function value from the literature. The fourth and

the fifth columns give the analytic form of the functions and their input domain for each variable (box constraints), respectively.

Function	Abbr.	f_{min}	Function $f(\mathbf{x})$	Input range
Dixon-Price	DP	0	$f(\mathbf{x}) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$	[-10, 10]
Griewank	GR	0	$f(\mathbf{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600, 600]
Powell	PO	0	$f(\mathbf{x}) = \sum_{i=1}^{n/4}[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4]$	[-4, 4]
Rosenbrock	RO	0	$f(\mathbf{x}) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-10, 10]
Schwefel	SC	0	$f(\mathbf{x}) = 418.9828727243379980791360139 8n - \sum_{i=1}^{n} x_i sin(\sqrt{\lvert x_i \rvert})$	[-500, 500]
Zakharov	ZA	0	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$	[-5, 5]
Rastrigin	RA	0	$f(\mathbf{x}) = 10n + \sum_{i=1}^{n}[x_i^2 - 10cos(2\pi x_i)]$	[-5.12, 5.12]
Sphere	SP	0	$f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2$	[-5.12, 5.12]
Ackley	AC	0	$f(\mathbf{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left[\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right] + 20 + e$	[-32.768, 32.768]
Noncontinuous Rastrigin	NR	0	$f(\mathbf{x}) = \sum_{i=1}^{n}[y_i^2 - 10cos(2\pi y_i) + 10]$, $y_i = \begin{cases} x_i & \text{, if } \lvert x_i \rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \text{, if } \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	[-5.12, 5.12]

Table 1: Standard test instances. The dimensionality of these functions can be adjusted with n.

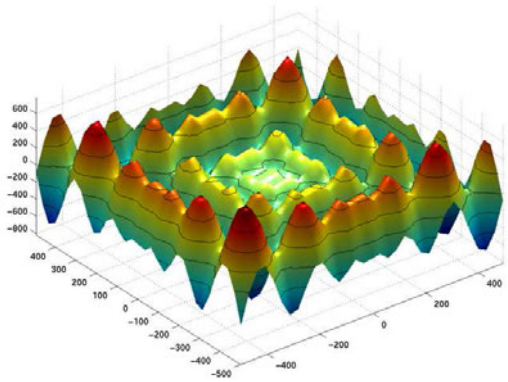

Figure 2: Schwefel Function.

For all functions in Table 1, we used $n =$ 10, 20, 40, 60, 80, 100, 250, 500, 750, and 1000. Each configuration was executed 10 times. The results comprise a total of 10 tables given in the Appendix A. The convergence criterion for all algorithms is set to be the combination of consecutive iterations without improvement ($it_{max} = 10,000$) and the condition defined in (2) with $\epsilon = 10^{-10}$. In the Multistart SNM, the maximum number of nonconsecutive improvements is set to $\zeta = 100$. In addition, the stopping criterion of the NM is at most 1000 seconds if other two criteria are not satisfied earlier. The stopping criteria for SNM is the average running time of the NM method. Standard choices for the Nelder-Mead coefficients are taken, i.e., $\alpha = 1$, $\beta = 2$, $\gamma = 1/2$ and $\delta = 1/2$, and $\tau = 4$ (for generating the initial simplex).

The average results for 10 independent restarts of each test function are given in Table 2. More precisely, each line of Table 2 reports average of 100 instances: 10 restarts for each out of 10 different n values.

Prob	NM		SNM		CPU Time
name	*Best*	*Average*	*Best*	*Average*	in seconds
DP	2.71E-13	6.69E-02	0.00E+00	1.06E-29	1.82E+01
GR	9.51E-01	1.45E+00	1.48E-09	5.23E-02	4.20E+02
PO	1.35E-15	6.78E-12	5.54E-47	9.53E-32	3.98E+01
RO	3.22E+02	3.58E+02	4.53E-09	7.90E-02	4.18E+02
SC	5.69E+04	6.30E+04	2.29E-09	2.20E+03	3.96E+02
ZA	4.14E+00	1.97E+03	2.76E-12	9.52E-12	3.02E+02
RA	2.04E+02	2.83E+02	9.16E-09	6.06E+00	4.03E+02
SP	1.02E-01	1.70E-01	2.14E-12	7.01E-12	4.16E+02
AC	2.67E+00	3.84E+00	4.28E-06	1.16E-02	3.74E+02
NR	5.52E+02	8.72E+02	2.86E-10	9.23E+00	1.54E+02

Table 2: omparison of best and average objective function values obtained by NM and SNM on 10 standard test instances from Table 1 within the same cpu times. Each instance, defined by the problem and n, is restarted 10 times. Ten different values of n are tested: 10, 20, 40, 60, 80, 100, 250, 500, 750, and 1000, so, each line reports average values of 100 runs.

Observations. The following conclusions may be derived from Table 2, and also from Tables 3 - A.13. Note that the total number of tests in each line, and each solution method of Table 2 is equal to 100 (10 test problems for 10 different values of n). Since there are 10 test instances, i.e., DP, GR, $\cdots$, NR, tables below report values obtained by 1000 runs for each method. We believe it was enough to get some reliable conclusions:

- The main conclusion is that the LIMA idea, applied on NM optimization method, is effective. Results obtained by NM method are significantly improved by simplified NM (SNM) within the same CPU time limitation.
- SNM appers to be most successful in solving all 10 test instances, which is clear after comparing values in columns 'Best' and 'Average' for NM and SNM in Table 2; for all instances, the SNM best and average values were smaller (better) than the corresponding values in NM 'Best' and Average columns of Table 2.
- Optimal solutions for all instances and for all dimensions are found by SNM at least once in 10 restarts (see column 'Best' for SNM method in Table 2). This is an unexpected result and deserves more attention in a future work. Indeed, NM was designed for solving convex nonlinear programs. However, SNM is able to find global minimum for instances where many local minima exist.

In order to emphasize that the SNM is much better than the original Nelder-Mead method, we present in Table 3 the average time in seconds needed for the SNM to reach the average solution of the NM and the number of times it has happened in 100 tests (including 10 different function and 10 different dimensions).

Problem name	Time (sec.)	Av # of cases	Problem name	Time (sec.)	Av # of cases
DP	0.002	100%	ZA	0.007	100%
GR	0.045	97%	RA	1.490	92%
PO	0.007	100%	SP	0.004	100%
RO	0.001	99%	AC	0.016	100%
SC	0.148	100%	NR	0.361	94%

Table 3: Average time needed for the SNM to reach average solution of NM and the number of cases it occurres, out of 100 tests for each instance (10 restarts for 10 different dimensions).

Table 3 shows that the SNM quickly reaches the average NM solution and the remaining time is used for improving the quality of the solution. The worst behavior of SNM, but still very good, is observed in solving non-convex Rastrigin test function (see Figure 3). There, the best NM value is reached 92 times.

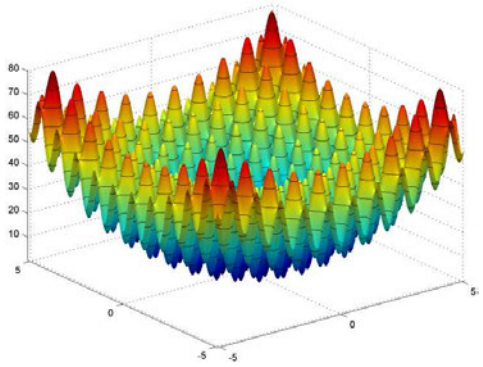

Figure 3: Rastrigin Function.

Very good results are reported in solving Ackley test function, where the NM solutions are reached always, within average time of 0.016 seconds!

4. CONCLUSIONS and SUGGESTIONS

The general idea of the recent Less is more approach (LIMA) for solving optimization problems is to consider the minimum number of ingredients in building a method, at the same time providing better quality solutions for the given problem than the solutions provided by more complex routines, or by the current state-of-the-art algorithm. [12]. Nelder-Mead method (NM) for solving continuous non-linear optimization problem [14] is probably the most cited and the most used method in optimization literature, and in practical applications, too. It belongs to direct search methods for solving unconstrained continuous optimization problems (UCOP). Such methods do not use first and second order conditions in solving UCOP. The popularity of NM is based on its simplicity. It calculates objective function values at $n+1$ (simplex) vertices in $\mathcal{R}^n$ and in each iteration, the

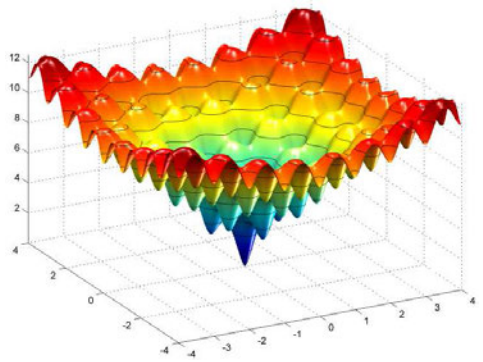

Figure 4: Ackley Function.

worst vertex is exchanged with a new, better one. When appeared, it successfully replaced direct search methods that used $2\times n$ and more points from $\mathcal{R}^n$. In this paper we propose even more simple algorithm that follows NM idea, which we call Simplified NM (SNM). Instead of generating all $n+1$ points, we perform search only by using $q+1$ vectors, where q is usually much smaller than n. Of course, our results cannot be better than the results obtained after performing calculations in $n+1$ points as in NM. However, the significant speed-up in cpu times allows to run many times SNM, so getting much better results than those obtained by NM. Significant improvements within the same running time are reported in solving non-convex instances as well.

Incouraging results reported in this paper may be continued in several directions:

(i) The way how NM method is simplified, or depleted, is general and may be applied to other optimization techniques as well. Moreover, any numerical method, not only the optimization one, could be simplified or depleted in the way we did for NM;

(ii) SNM could be used as a local search within other meatheuristics for solving global optimization problems, such as Variable neighborhood search [3];

(iii) For solving even larger problems, a SNM modified variant could include the increment parameter q, i.e., the subspace dimension in each iteration, as it is done for example in Variable neighborhood decomposition search [18];

(iv) Simplified variants of other sucessful NM modifications could be tried out as well;

Acknowledgement: This work was partially supported by CNPq-Brazil grants 308887/2014-0 and 400350/2014-9. Work of N. Mladenović author was conducted

at National Research University Higher School of Economics and supported by RSF grant 14-41-00039. This work was also supported by NPAD/UFRN.

REFERENCES

[1] Dražić, M., Dražić, Z., Mladenović, N., Urošević, D., and Zhao, Q., H., "Continuous variable neighbourhood search with modified Nelder–Mead for non-differentiable optimization", *IMA Journal of Management Mathematics*, dpu012, 2014.

[2] Gao, F., and Han, L., "Implementing the Nelder-Mead simplex algorithm with adaptive parameters", *Computational Optimization and Applications*, 51 (1) (2012) 259-277.

[3] Hansen, P., Mladenović, N., Todosijević, R., and Hanafi, S., "Variable neighborhood search: basics and variants", *EURO Journal on Computational Optimization*, 5 (3) (2017) 423-454.

[4] Kelley, CT., "Detection and remediation of stagnation in the nelder–mead algorithm using a sufficient decrease condition", *SIAM ournal on ptimization*, 10 (1) (1999) 43-55.

[5] Kolda, T., G., Lewis, R., M., and Torczon, Virginia, "Optimization by direct search: New perspectives on some classical and modern methods", *SIAM Review*, 45 (3) (2003) 385-482.

[6] Kovačević, D., Mladenović, N., Milošević, P., Petrović, B., and Dobrić, V., *Comparative analysis of continuous global optimization methods*, Department of Computer Science, Michigan State University, G-2013-41, Montreal, Canada, 2013.

[7] Kovačević, D., Mladenović, N., Petrović, B., and Milošević, P., "DE-VNS: Self-adaptive Differential Evolution with crossover neighborhood search for continuous global optimization", *Computers & Operations Research*, 52 (part B) (2014) 157-169.

[8] Lagarias, J., C., Reeds, J., A., Wright, M., H., and Wright, P., E., "Convergence properties of the Nelder–Mead simplex method in low dimensions", *SIAM Journal on Optimization*, 9 (1) (1998) 112-147.

[9] Luangpaiboon, P., "Variable neighborhood simplex search methods for global optimization models", *Journal of Computer Science*, 8 (4) (2012) 613-620.

[10] McKinnon, Ken IM, "Convergence of the Nelder–Mead Simplex Method to a Nonstationary Point", *SIAM Journal on Optimization*, 9 (1) (1998) 148-158.

[11] Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., and Čangalović, M., "General variable neighborhood search for the continuous optimization", *European Journal of Operational Research*, 191 (3) (2008) 753-770.

[12] Mladenović, N., Todosijević, R., and Urošević, D., "Less is more: basic variable neighborhood search for minimum differential dispersion problem", *Information Sciences*, 326 (2016) 160-171.

[13] Nazareth, L., and Tseng, P., "Gilding the lily: A variant of the Nelder-Mead algorithm based on golden-section search", *Computational Optimization and Applications*, 22 (1) (2002) 133-144.

[14] Nelder, J., A., and Mead, R., "A simplex method for function minimization", *The Computer Journal*, 7 (4) (1965) 308-313.

[15] Price, C., J., Coope, I., D., and Byatt, D., "A convergent variant of the Nelder–Mead algorithm", *Journal of optimization theory and applications*, 113 (1) (2002) 5-19.

[16] Rykov, AS., "Simplex algorithms for unconstrained minimization", *Prob. Contr. Info. Theory.*, 12 (3) (1983) 195-208.

[17] Singer, S., and Nelder, J., "Nelder-Mead algorithm", http://www.scholarpedia.org/article/Nelder-Mead_algorithm, Accessed: 2017-04-04, 2009.

[18] Urošević, D., Brimberg, J., and Mladenović, N., "Variable neighborhood decomposition search for the edge weighted k-cardinality tree problem", *Computers & Operations Research*, 31 (8) (2004) 1205-1213.

[19] Wang, P., C., and Shoup, T., E., "Parameter sensitivity study of the Nelder–Mead simplex method", *Advances in Engineering Software*, 42 (7) (2011) 529-533.

[20] Wright, Margaret, H., "Nelder, Mead, and the other simplex method", *Documenta Mathematica*, 7 (2010) 271-276.

[21] Zhao, Q., H., and Urosević, D., and Mladenović, N., and Hansen, P., "A restarted and modified simplex search for unconstrained optimization", *Computers & Operations Research*, 36 (12) (2009) 3263-3271.

[22] Zhao, Q., Mladenović, N., and Urošević, D., "A parametric simplex search for unconstrained optimization problem", *Trans Adv Res*, 8 (2012) 22-27.

Appendix A. Detailed Results

Here we give detailed results obtained by NM and SNM. All tables have the same form: the column *Best* expresses the objective function value of the best solution; the column *Average* describes the average objective function value and the column *CPU Time in seconds* shows the average running time in seconds.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	7.74E-13	2.86E-12	0.00E+00	3.03E-31	9.86E-04
20	8.96E-13	4.57E-12	0.00E+00	4.22E-30	4.19E-03
40	2.27E-13	6.69E-01	0.00E+00	1.01E-28	9.66E-03
60	4.91E-13	1.08E-12	0.00E+00	0.00E+00	3.08E-02
80	2.07E-13	8.21E-13	0.00E+00	0.00E+00	7.69E-02
100	4.36E-14	1.14E-12	0.00E+00	4.44E-32	1.30E-01
250	1.55E-14	3.00E-13	0.00E+00	5.76E-36	1.99E+00
500	2.50E-14	2.13E-13	0.00E+00	0.00E+00	1.49E+01
760	9.22E-15	1.29E-13	0.00E+00	0.00E+00	4.94E+01
1000	2.53E-14	1.27E-13	0.00E+00	0.00E+00	1.16E+02
Avg	2.71E-13	6.69E-02	0.00E+00	1.06E-29	1.82E+01

Table A.4: Detailed results for Dixon-Price Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	8.37E-02	7.77E-01	4.85E-09	2.52E-01	2.07E+00
20	9.86E-03	4.49E-01	3.54E-09	1.48E-01	3.77E+00
40	6.66E-01	1.18E+00	2.64E-09	3.40E-02	3.43E+00
60	6.33E-01	1.33E+00	1.14E-09	1.56E-02	9.68E+00
80	9.49E-01	1.34E+00	8.70E-10	5.02E-02	4.12E+01
100	3.64E-01	9.70E-01	9.68E-10	1.04E-02	1.60E+02
250	3.73E-01	8.52E-01	2.03E-10	1.32E-09	9.83E+02
500	1.34E+00	1.53E+00	2.19E-10	5.66E-10	1.00E+03
760	2.02E+00	2.39E+00	2.52E-10	1.37E-02	1.00E+03
1000	3.08E+00	3.70E+00	1.05E-10	2.75E-10	1.00E+03
Avg	9.51E-01	1.45E+00	1.48E-09	5.23E-02	4.20E+02

Table A.5: Detailed results for Griewank Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	6.88E-22	4.25E-19	2.46E-50	2.98E-31	6.24E-03
20	3.66E-21	2.43E-17	5.50E-46	2.76E-33	6.28E-03
40	1.71E-21	2.51E-16	7.88E-50	6.10E-31	3.50E-02
60	4.74E-20	2.41E-12	1.81E-48	1.04E-40	9.57E-02
80	1.27E-21	1.31E-11	8.51E-53	3.45E-32	2.18E-01
100	5.05E-20	4.74E-13	2.28E-48	6.98E-34	4.45E-01
250	1.81E-22	4.95E-12	1.92E-55	6.82E-33	8.80E+00
500	9.50E-21	2.21E-11	3.20E-52	1.57E-34	5.15E+01
760	2.40E-17	1.72E-11	9.07E-60	3.80E-49	1.72E+02
1000	1.35E-14	7.44E-12	3.34E-56	1.34E-37	1.65E+02
Avg	1.35E-15	6.78E-12	5.54E-47	9.53E-32	3.98E+01

Table A.6: Detailed results for Powell Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	1.65E-10	1.20E+00	9.14E-09	7.90E-01	5.32E-02
20	9.34E+00	3.14E+01	6.60E-09	5.23E-07	4.63E-01
40	4.23E+01	1.04E+02	9.03E-09	4.97E-07	5.87E+00
60	5.72E+01	1.13E+02	2.54E-09	2.18E-07	1.08E+01
80	7.60E+01	1.29E+02	6.74E-09	2.31E-07	3.06E+01
100	1.01E+02	1.43E+02	2.50E-09	1.17E-07	1.31E+02
250	2.38E+02	2.51E+02	1.42E-09	7.98E-08	1.00E+03
500	5.27E+02	5.58E+02	2.05E-09	1.41E-07	1.00E+03
760	8.74E+02	9.16E+02	1.50E-09	2.77E-07	1.00E+03
1000	1.30E+03	1.34E+03	3.76E-09	1.85E-07	1.00E+03
Avg	3.22E+02	3.58E+02	4.53E-09	7.90E-02	4.18E+02

Table A.7: Detailed results for Rosenbrock Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	1.01E+03	1.74E+03	3.82E-09	4.08E+02	2.83E-01
20	2.65E+03	3.80E+03	2.31E-09	3.63E+02	1.30E+00
40	6.23E+03	8.29E+03	3.02E-09	1.59E+03	5.82E+00
60	1.10E+04	1.27E+04	3.08E-09	1.25E+03	1.22E+01
80	1.37E+04	1.63E+04	2.80E-09	4.52E-09	3.31E+01
100	1.86E+04	2.21E+04	1.62E-09	5.92E+02	1.12E+02
250	4.92E+04	5.75E+04	1.82E-09	2.96E+03	8.91E+02
500	1.02E+05	1.13E+05	1.08E-09	2.96E+03	9.50E+02
760	1.53E+05	1.60E+05	1.46E-09	5.43E-09	1.00E+03
1000	2.12E+05	2.35E+05	1.86E-09	1.18E+04	9.50E+02
Avg	5.69E+04	6.30E+04	2.29E-09	2.20E+03	3.96E+02

Table A.8: Detailed results for Schwefel Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	1.29E-11	8.49E-11	2.93E-12	9.44E-12	1.69E-02
20	7.24E-04	2.25E-01	3.41E-12	9.13E-12	5.06E-02
40	3.52E-01	1.12E+00	3.43E-12	7.66E-12	5.52E+00
60	7.04E-01	1.85E+00	3.13E-12	1.11E-11	8.00E+00
80	1.21E+00	1.85E+00	2.83E-12	1.19E-11	1.68E+01
100	1.04E+00	1.74E+00	2.36E-12	9.92E-12	4.18E+01
250	1.22E+00	1.96E+00	1.62E-12	5.48E-12	5.24E+02
500	3.75E+00	4.92E+00	2.08E-12	9.33E-12	8.56E+02
760	1.38E+01	1.54E+01	3.10E-12	8.50E-12	9.51E+02
1000	1.93E+01	1.97E+04	2.75E-12	1.27E-11	6.17E+02
Avg	4.14E+00	1.97E+03	2.76E-12	9.52E-12	3.02E+02

Table A.9: Detailed results for Zakharov Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	5.97E+00	1.31E+01	8.68E-08	1.27E+01	3.07E-01
20	7.10E+00	2.25E+01	7.63E-10	7.59E+00	1.38E+00
40	2.31E+01	3.43E+01	8.52E-10	1.44E+01	5.88E+00
60	2.52E+01	4.14E+01	6.40E-10	2.39E+00	1.07E+01
80	2.56E+01	4.91E+01	4.45E-10	1.61E+00	3.75E+01
100	3.61E+01	5.03E+01	4.77E-10	2.01E+00	1.41E+02
250	7.30E+01	1.28E+02	5.22E-10	3.34E-09	8.82E+02
500	2.11E+02	7.23E+02	4.33E-10	1.98E+01	9.50E+02
760	5.26E+02	6.02E+02	2.14E-10	9.95E-02	1.00E+03
1000	1.11E+03	1.17E+03	4.19E-10	3.23E-08	1.00E+03
Avg	2.04E+02	2.83E+02	9.16E-09	6.06E+00	4.03E+02

Table A.10: Detailed results for Rastrigin Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	2.12E-11	4.74E-11	3.59E-12	8.37E-12	1.80E-03
20	1.58E-10	6.42E-10	2.29E-12	8.34E-12	2.36E-02
40	1.76E-03	8.35E-02	2.39E-12	7.40E-12	9.37E-01
60	4.72E-03	1.15E-01	1.51E-12	8.10E-12	6.87E+00
80	8.61E-03	1.01E-01	1.36E-12	5.62E-12	3.83E+01
100	1.61E-05	3.18E-02	1.96E-12	6.20E-12	1.41E+02
250	2.27E-03	3.29E-02	3.29E-12	7.55E-12	9.76E+02
500	1.01E-01	1.49E-01	1.79E-12	4.82E-12	1.00E+03
760	2.94E-01	4.02E-01	1.62E-12	8.38E-12	1.00E+03
1000	6.06E-01	7.85E-01	1.64E-12	5.28E-12	1.00E+03
Avg	1.02E-01	1.70E-01	2.14E-12	7.01E-12	4.16E+02

Table A.11: Detailed results for Sphere Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	3.03E+00	7.26E+00	1.26E-05	1.16E-01	2.24E+00
20	1.84E+00	5.13E+00	7.29E-06	1.26E-05	2.26E+00
40	3.63E+00	5.12E+00	5.76E-06	9.14E-06	5.55E+00
60	4.00E+00	4.70E+00	4.34E-06	8.78E-06	9.02E+00
80	3.53E+00	4.15E+00	4.36E-06	7.81E-06	2.67E+01
100	2.82E+00	3.58E+00	3.06E-06	4.95E-06	9.26E+01
250	2.42E+00	2.63E+00	1.68E-06	3.03E-06	7.43E+02
500	1.95E+00	2.13E+00	1.25E-06	1.83E-06	1.00E+03
760	1.72E+00	1.88E+00	1.07E-06	2.11E-06	9.51E+02
1000	1.72E+00	1.85E+00	1.37E-06	1.80E-06	9.06E+02
Avg	2.67E+00	3.84E+00	4.28E-06	1.16E-02	3.74E+02

Table A.12: Detailed results for Ackley Function.

	NM		SNM		CPU Time
n	*Best*	*Average*	*Best*	*Average*	in seconds
10	7.03E+00	1.91E+01	4.05E-10	7.96E+00	2.17E-01
20	1.30E+01	3.51E+01	4.02E-10	2.25E+01	1.59E+00
40	3.87E+01	7.20E+01	4.33E-10	9.88E-10	5.02E+00
60	3.30E+01	9.05E+01	8.21E-10	5.69E+01	5.61E+00
80	7.20E+01	1.98E+02	3.45E-10	1.09E-09	6.63E+00
100	8.35E+01	2.14E+02	9.89E-11	1.19E-09	1.07E+01
250	2.04E+02	5.96E+02	4.54E-11	9.90E-10	1.81E+02
500	1.05E+03	1.63E+03	5.36E-11	5.02E-10	4.16E+02
760	1.59E+03	2.41E+03	1.94E-10	5.68E-10	4.62E+02
1000	2.43E+03	3.45E+03	6.64E-11	4.92E+00	4.50E+02
Avg	5.52E+02	8.72E+02	2.86E-10	9.23E+00	1.54E+02

Table A.13: Detailed results for Noncontinuous Rastrigin Function.