Les Cahiers du GERAD ISSN: 0711–2440

A Parametric Simplex Search for Unconstrained Optimization Problem

Q. Zhao, N. Mladenović
D. Urošević

G–2011–28

June 2011

A Parametric Simplex Search for Unconstrained Optimization Problem

Qiuhong Zhao

School of Economics and Management
Beihang University
Beijing, China
qhzhao@buaa.edu.cn

Nenad Mladenović

GERAD & School of Mathematics
Brunel University-West London
Uxbridge, UB8 3PH, UK
nenad.mladenovic@brunel.ac.uk

Dragan Urošević

Mathematical Institute, SANU
Belgrade, Serbia
draganu@mi.sanu.ac.rs

June 2011

Les Cahiers du GERAD

G–2011–28

Abstract

In this paper an effective modification to the original Nelder-Mead simplex method is suggested. It is shown that the new heuristic outperforms on average the original version of NM as well as its several modifications, showing especially its robustness in solving the standard functions. This result clearly indicates benefits of introducing randomness into a deterministic search procedure.

Key Words: Unconstrained optimization; Global optimization; Nonlinear Programming, Direct methods, Nelder–Mead.

Résumé

Dans cet article, on propose une modification efficace de l'algorithme original du simplexe de Nelder-Mead. On montre que cette nouvelle heuristique a en moyenne de meilleures performances que la version originale de Nelder-Mead, ainsi que plusieurs de ses modifications, constatant en particulier la robustesse de la résolution de fonction standard. Ce résultat illustre clairement le bénéfice d'introduire la randomisation dans une procédure de recherche déterministe.

Acknowledgments: The work of the first author was partially supported by New Century Excellent Talents in University of China under Project No. NCET-07-0049 and by National Natural Science Foundation of China under Projects No. 70771001 and No.70821061. Second and third authors are partially supported by Serbian Ministary of Sciences, project number 144007.

1 Introduction

Let us consider unconstrained continuous optimization problem:

$$\min \{f(x) \mid x \in R^n\}, \tag{1}$$

where objective function $f : R^n \to R$. If the objective function is nor convex nor concave, there may have many local minima.

The simplex search methods [13] are a kind of *direct search* methods. Note that simplex contains $(n+1)$ points $x_i \in R^n$, $i = 1, ..., n+1$. Among numerous simplex methods, the Nelder-Mead (NM) algorithm is the most popular one. Since published in 1965, the NM algorithm has been used in many fields of science and technology. Despite being widely used, the NM's practical performance can be appallingly poor in some cases [15]. Recently, numerous modifications to the NM algorithm, including metaheuristic originated NM versions, have been proposed, most of which aim at improving the worst-case performance.

In this paper, we present a modification to the original NM method, for unconstrained optimization. We explore the fact that all possible improved solutions belong to the same line, i.e., the line that connects worst simplex vertex with the centroid of the simplex. Then, instead of taking only reflection, expansion and contraction points on that line, we introduce randomness to generate more trial points within one iteration. We call our modification Parametric Simplex Search (PSS) method, since it integrates the parametric search into the method.

The rest of the paper is organized as follows. In the next section we briefly outline the NM algorithm and present a *general formulation* drawn from it. In Section 3 we give details of our algorithm. Extensive computational analysis is conducted in Section 4. Finally, conclusions are drawn in Section 5.

2 The General Nelder-Mead Method

2.1 NM procedure

In this subsection, the *initial simplex*, *One iteration* of the NM algorithm and the *termination criterion* are demonstrated respectively. The overall description of NM is given at last.

Initial simplex. It is customary to specify a starting point in R^n that is taken as one of the initial simplex vertices. The other n vertices are then usually generated by perturbing the starting point by a specified step along the n coordinate directions, or by creating a regular simplex with specified edge length and orientation.

One NM iteration. Let $X = \{x_1, x_2, \dots, x_{n+1}\}$ be the current simplex, where the points are ordered according to the function values, let x_1 be the best one (with minimal function value), and x_{n+1} the worst. One NM iteration is shown in (*Algorithm 1*).

"Point x is accepted" in Algorithm 1 means that x replaces the worst point from X. According to the original Nelder-Mead paper [13], the parameters $\alpha, \beta, \gamma, \delta$ in the NM iteration should satisfy the following conditions: $\alpha > 0$, $\beta > 1$, $0 < \gamma < 1$, and $0 < \delta < 1$. Their usual values are $\alpha = 1$, $\beta = 2$,$\gamma = \frac{1}{2}$, and $\delta = \frac{1}{2}$.

Termination criterion. One of the two termination criteria is usually used: either the function values at all vertices are close, or the volume of the simplex becomes very small ($Volume(X) < \varepsilon$, where X and ε are the current simplex, and an arbitrary small number, respectively).

The overall steps of the NM algorithm is in (*Algorithm 2*).

2.2 A Generalization of the NM Procedure

Proposition 1 *Reflection, expansion and contraction points belong to the same line* $x_g = \bar{x} + g(\bar{x} - x_{n+1})$.

Algorithm 1 NM-Iteration(X, f)

```
1: Order. Order the $n+1$ vertices of $X$ to satisfy $f(x_1) \leqslant f(x_2) \cdots \leqslant f(x_{n+1})$.
2: Reflect. Compute the reflection point $x_r$ as $x_r = \bar{x} + \alpha(\bar{x} - x_{n+1})$, where $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the centroid
   of the $n$ best points (all vertices except for $x_{n+1}$). If $f(x_1) \leqslant f(x_r) < f(x_n)$ accept the reflected point
   $x_r$, terminate the iteration.
3: Expand. If $f(x_r) < f(x_1)$, calculate the expansion point $x_e = \bar{x} + \beta(x_r - \bar{x})$. If $f(x_e) \leqslant f(x_r)$, accept
   the expanded point $x_e$ and terminate; otherwise accept
   $x_r$ and terminate the iteration.
4: Contract. If $f(x_r) \geqslant f(x_n)$, perform a contraction between $\bar{x}$ and the better of $x_{n+1}$ and $x_r$.
   (a) Outside. If $f(x_r) < f(x_{n+1})$, then outside contraction: $x_c = \bar{x} + \gamma(x_r - \bar{x})$. If $f(x_c) \leqslant f(x_r)$, accept
   $x_c$ and terminate; otherwise go to Shrink step.
   (b) Inside. If $f(x_r) \geqslant f(x_{n+1})$, then inside contraction: $x_{c'} = \bar{x} - \gamma(\bar{x} - x_{n+1})$. If $f(x_{c'}) \leqslant f(x_{n+1})$,
   accept $x_{c'}$ and terminate; otherwise go to Shrink step.
5: Shrink. Evaluate $f$ at the $n$ points $v_i = x_1 + \delta(x_i - x_1), i = 2, \ldots, n+1$. The (unordered) vertices at
   the next iteration consist of $V = \{x_1, v_2, \ldots, v_{n+1}\}$; set $X = V$.
```

Algorithm 2 Nelder-Mead simplex method

```
1: Get an initial point $x \in R^n$ at random
2: $X \leftarrow$ Initial-Simplex
3: while The termination criterion is not met do
4:   NM-Iteration($X, f$)
5: end while
6: return  return $x_1$ as solution.
```

Proof. From Algorithm 1 we have

$$x_r = (1+\alpha)\bar{x} - \alpha x_{n+1}, \;\; x_e = (1+\alpha\beta)\bar{x} - \alpha\beta x_{n+1},$$

$$x_c = (1+\alpha\gamma)\bar{x} - \alpha\gamma x_{n+1}, \;\; x_{c'} = (1-\gamma)\bar{x} + \gamma x_{n+1}.$$

If we denote $G = \{\alpha, \alpha\beta, \alpha\gamma, -\gamma\}$, then we can express the *relect*, *expand* and *contract* (*inside* and *outside*) points in one NM iteration with the following *general formulation*:

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

where $g \in G$. Therefore, after simple transformation on (2), desired result is obtained. $\square$

The original NM algorithm and the most of its modifications specify *reflect*, *expand*, *contract* steps in sequence and fix the values of α, β, γ in advance. We change systematically at random an interval from which the value of g in Formulation (2) is selected. In addition, we expand the range of g (contrasting to that given in the original NM, which is $[-\frac{1}{2}, 2]$ at specified points $\{-\frac{1}{2}, \frac{1}{2}, 1, 2\}$), to search for the global optimum in a wider but reasonable interval.

3 PSS and RPSS Algorithms

3.1 The PSS Algorithm

By changing the value of parameter g in (2), creating accordingly a new vertex, the PSS algorithm follows the original version of NM, but in more flexible way. We first specify the *initial simplex* and the *stopping criteria* of the algorithm. One PSS iteration is given in (*Algorithm 3*). After the illustration of our revised *shrink* step, the overall PSS algorithm is given in (*Algorithm 4*).

Initial Simplex. Let $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,n}) \in R^n$ denotes the ith vertex of the current simplex, $i = 1, 2, ..., n+1$. The starting vertex $x_1 = (x_{1,1}, x_{1,2}..., x_{1,n}) \in R^n$ of the initial simplex X is chosen at random; otherwise x_1 is chosen by the following VNS, which is depicted in line 3 (*Algorithm 5*). The remaining vertices are created by the formulation listed in line 5 of *Algorithm 5*, where τ is a positive constant, e_i is the $n-$dimensional unit vector with one in the ith component and zeros elsewhere.

Stopping Criteria. The stopping criteria are set to be the combination of two conditions: (1) The continuous iteration number without meeting *Inequality (3)* accumulates to J:

$$f^* - f(x_1) > \rho|f^*|, \tag{3}$$

where f^* is the minimum function value ever found before, and $\rho > 0$. (2) The volume of the current simplex is small enough to meet *Inequality (4)*:

$$\frac{|f(x_1)| + |f(x_{n+1})|}{|f(x_1)| + |f(x_{n+1})| + 10^{\varepsilon_o}} \leqslant 10^{\varepsilon_o}, \tag{4}$$

where ε_o is a minus integer predefined.

One Iteration. We define I_k as the interval, which are characterized by iteration number k. For any selected, the PSS procedure chooses randomly a value $g' \in I_k$, and conducts search from a predefined interval around g'. The PSS based iteration is given in *Algorithm 3*, In Algorithm 3, according to the following formulation:

$$I_k = [d_k, u_k] = [A - \lfloor k/a \rfloor, A - \lfloor k/a \rfloor + b], \tag{5}$$

where A, a and b are the positive constants, $\lfloor z \rfloor$ the largest integer not larger than z. As shown in (5), along with the increase of k, d_k either keeps unchanged or decreases, so does u_k.

Algorithm 3 One iteration of PSS.

```
1: function OnePSS($(n, X, f)$)
2: $k \leftarrow 0$; $x_{new} \leftarrow x_{n+1}$.
3: Define $M_{g'}$ as an interval around $g' \in I_k$. Take $e$ as the step for local search.
4: while $k \leqslant k_{\max}$ do
5:    Select randomly a value $g' \in I_k$, denote $x_w = min_{f(x_g)}\{x_g = (1+g)\bar{x} - gx_{n+1} | g \in M_{g'}\}$.
6:    if $f(x_w) < f(x_{n+1})$ then
7:       $x_{new} \leftarrow x_w$
8:       break
9:    else
10:      $k \leftarrow k + 1$;
11:   end if
12: end while
13: return $x_{new}$
```

The Modified Shrink Procedure. Our modified *shrink* step is activated when we fail to find a solution better than x_{n+1} in previous step. We shrink only part of the points each time, the number q, is determined randomly and $1 \leqslant q < r < n$, where r (a parameter) is a predefined value. The detailed description of the modified *shrink* procedure is given in Algorithm 4.

By shrinking part of the points, we can diversify the simplex, while the information of the previous simplex can be inherited as well. In the case that q is always too small to diversify the simplex, the algorithm can avoid of converging to the local optimum and the Restarted PSS processes.

Pseudo-code for PSS. The Pseudo-code for PSS is in Algorithm 4.

3.2 The RPSS algorithm

The RPSS algorithm is characterized by restarting the PSS algorithm when its stopping condition is met.

The way of restarting the NM simplex or its modifications has already been applied ([11, 18], etc.). It is shown to be significant for the search of the global optimum. The method in [11] is called as Restarted and Revised Simplex (RRS), which consists of a three-phase application of the NM method in which: (a) the

Algorithm 4 Pseudo-code for PSS

```
function PSS($n, X, f$)
$X \leftarrow$ Initial-Simplex $(n, x)$
$shrink \leftarrow false$
while Stopping condition is not met do
  if $shrink$ then
    Select randomly a number $q < r < n$,
    $x_i = x_1 + \delta(x_i - x_1), i = n - q + 2, \ldots, n, n + 1.$
    $shrink \leftarrow false$
  else
    $x_{new} \leftarrow$ OnePSS($n, X, f$)
    if $x_{new} < x_{n+1}$ then
      $x_{n+1} \leftarrow x_{new}$
    else
      $shrink \leftarrow true$;
    end if
  end if
  reorder $X$
end while
return $x_1$ as solution.
```

ending values for one phase become the starting values for the next phase; (b) the step size for the initial simplex (respectively, the shrink coefficient) decreases geometrically (respectively, increases linearly) over the successive phases; and (c) the final estimated optimum is the best of the ending values for the three phases. In [18], the Restarted and Modified NM (RMNM) method takes the same size (volume) of the simplex in the proceeding phases as in the first one and finishes the procedure when the optimal value $f(x_1)$ in the current phase is not better than the best one (denote it as $f(x)$) obtained in the previous phase; *i.e.*, the termination criterion is $f(x_1) \geqslant f(x)$.

In this paper, we restart PSS for global searching. Whenever PSS is restarted, we diversify the initial simplex, such that the iteration process can be conducted in a wider interval if the function value is not improved.

Constructions of the initial simplex are included in Algorithm 5. If it is the first time to run PSS, the point x_1 in the initial simplex X is selected randomly; whenever PSS is restarted, the point x_1 is constructed by perturbing the best vertex gained in the previous phases, as shown in line 3, where $m > 1$. In both cases, the other points are created by the formulation listed in line 5, where τ is a positive constant, e_i is the n-dimensional unit vector with one in the ith component and zeros elsewhere. As shown in line 3, even though x_1 is chosen randomly, the probability of x_1 departuring more from the best vertex gained increases if the optimal function value keeps unimproved in the previous phases, until $k = K$.

4 Computational Analysis

The computational process is divided into two parts. In the first subsection, we run the RPSS algorithm on a series of functions, whose dimensions change steadily within a given range, and compare the results with those drawn from [18]; in the second subsection, the computational experiment focuses on the functions with specified dimensions, which were also run by some meta-heuristic algorithms recently.

In Table 1, we list the values of some parameters in the PSS algorithm. These values are classified regarding the algorithm in which they are listed.

The values of the parameters listed in the column *OnePSS* are decided based on the analysis conducted in Section 3.1; the values of the parameters listed in the columns PSS and RPSS are decided based on testing.

Algorithm 5 Pseudo-code of RPSS

```
1: function RPSS($n, X, f$)
2: $x^* \leftarrow$PSS $(n, X, f)$, $x_1 \leftarrow x^*$, $k \leftarrow 0$
3: while $k \leqslant K$ do
4:     $x_1 \leftarrow x_1 \cdot (1 + \frac{k}{mK} \cdot \omega)$, $\omega \in [0, 1]$
5:     $z = max|x_{1_j}|$, $j = 1, ..., n$
6:     $x_{i+1} = x_1 + \max\{1, z\} \cdot e_i$, for $i = 1, ..., n$
7:     $x^* \leftarrow$ PSS$(n, X, f)$
8:     if $x^* < x_1$ then
9:         $x_1 \leftarrow x^*$, $k \leftarrow 0$;
10:    else
11:        $k \leftarrow k + 1$
12:    end if
13: end while
14: return $x_1$ as solution.
```

Table 1: The values of some parameters in the PSS algorithm

Parameter	*OnePSS* A	a	b	I	e	*PSS* J	ε_o	r	ρ	*RPSS* K	m	τ
Value	2.5	5	1	25	0.2	500	10^{-6}	$\frac{n}{2}$	1.5	10	5	3

4.1 Standard test functions with different dimensions

In this subsection, we conduct the comparative analyses on the functions in Table 2, with different dimensions. Besides the RPSS algorithm, three other methods —- NM, RRS and RMNM, are also considered, which are drawn from [18]. For each method, the initial simplex X is generated in the same way as the PSS method in the first phase of RPSS (that is, the point x_1 is generated randomly, the other points are generated based on the formation in line 5 of Algorithm 5, where $k = 0$). The termination criterion of NM is the same as each phase of RSS as well as RMNM, which is the combination of the iteration number without improvement (it is set to 10, 000) and Inequality (4). The values of the parameters $\alpha, \beta, \gamma, \delta$ for these methods except for RPSS are the same as what shown in Section 2.

All of these algorithms are coded in C++ and run on a Pentium 4 computer with 1400MHz processor and 256 MB of RAM. The same as in [2, 10], we use the following criterion to judge success (i.e., convergence to the global minimum) of a trial:

$$\tilde{f} - f_{min} < \epsilon_1 \mid f_{min} \mid + \epsilon_2, \tag{6}$$

where $\tilde{f}$ refers to the best function value obtained by the algorithm, f_{min} is the exact global minimum, ϵ_1 and ϵ_2 are set equal to 10^{-4} and 10^{-6}, respectively.

In Table 2, the name and the abbreviation of each test function are in the first two columns. The third column, n, denotes the dimensions and the last column f_{min} gives known minimum function values. These test functions (or some of them) can be found for examples in [3, 9, 2], etc., with diversity in characteristics of difficulties that arise in global optimization problems. By changing steadily dimension of these functions within a given range, we can well evaluate the robustness of these methods under consideration.

As shown in Table 2, for all the functions except Powell's function (which dimension should be times of 4), the dimension is changed from 10 to 100, with step 5, while the latter is from 8 to 100, with step 4. Thus, each test function consists of 19 ($\frac{100-10}{5} + 1$) instances ($\frac{100-8}{4} + 1 = 24$ instances for the Powell function).

The same as [18], to all the test functions and their dimensions considered, we run the RPSS code one time. The computational results of NM, RSS, RMNM and RPSS are in Table 3. In Table 3, for each function with $m = 19$ ($m = 24$ for Powell) time's running, we list from Column 2 to Column 6 the convergent times of each algorithm to the global minimum (Conv. Times), and from 7 to 11 the average of all the function values

Table 2: Standard test instances

Fun.	Abbr.	n	$f_{\min}$
Dixon and Price	DP	10, 15, 20, ..., 100	0
Griewank	GR	10, 15, 20, ..., 100	0
Powell	PO	8, 12, 16, ..., 100	0
Rosenbrock	RO	10, 15, 20, ..., 100	0
Schwefel	SC	10, 15, 20, ..., 100	0
Zakharov	ZA	10, 15, 20, ..., 100	0
Rastrigin	RA	10, 15, 20, ..., 100	0

Table 3: The computational results of the test functions in Table 2

	Conv. Times				Av.Val.			
Fun.	NM	RSS	RMNM	RPSS	NM	RSS	RMNM	RPSS
DP	0	0	**3**	0	209732.08	1.83	**0.56**	0.63
GR	0	8	9	**19**	61.88	0.04	0.02	**0.00**
PO	3	23	16	**24**	38.71	0.00	**0.00**	**0.00**
RO	0	1	**4**	1	832.14	58.88	127.44	**41.35**
SC	0	0	**3**	0	12091.18	10836.81	6659.31	**1314.28**
ZA	2	3	8	**19**	37.76	0.26	**0.00**	**0.00**
RA	0	0	18	**19**	300.94	208.59	**0.00**	**0.00**
Total	5	35	61	**82**	223094.69	11106.41	6787.33	**1363.59**
Avg.	1	5	9	**12**	1616.63	80.48	49.18	**9.88**

(AV.Val.) accordingly, where the computational results except for RPSS are drawn from [18]. We show in bold the best results (Conv.Times and Av.Val.) for each function.

In Table 3, for 4 functions, the convergence times of RPSS outperforms the other methods, while RMNM dominates the others. In addition, contrasting to other methods without 100 percentage convergency to any function, RPSS converges to the optimum at 100 percentage rate for the four functions it dominates. It can be seen from the last two rows that the total (average, accordingly) convergent times of RPSS is the biggest, where $Average = \lceil Total/7 \rceil$.

Usually the global optimums are unknown for the real problems, so "Average function Value" (Av.Val.) is an important index to judge the ability as well as robustness of any solver for complex unconstrained optimizations. In Table 3, the outperform of RPSS is also shown in column AV.Val. Except for function DP, RPSS either has the minimal average function value or has it the same as the best one.

4.2 Global optimization instances with specified dimension

In this section, we analyze the efficiency of RPSS, which is tested by using a set of benchmark functions with specified dimensions. Details of these functions can be found in [5, 9]. In Table 4, we list each function the name (Fun.), abbreviation (Abbr.), dimension (n) and optimal function value ($f_{\min}$). The metaheuristic algorithms compared with RPSS in this subsection are listed in Table 5. The first six methods in Table 5 use NM as subroutine, the last two have no relation with NM, however, they also employ VNS to find the optimum.

The same as other metaheuristic algorithms compared with RPSS in this subsection, we run the code of RPSS 100 times for per function, and summarize the results by following two criteria below: the rate of successful minimization, the average of the objective function evaluation number (which is called later as *function evaluation number*), where the latter criterion relates only to the successful trials.

The computational results are listed in the middle part of Table 6, where Columns contain the number of function evaluations needed to find the first global minimum. The numbers in parentheses denote number of runs for which the method found the global minimum; in the case of 100% no number is reported. For each function, we highlight in bold the minimal function evaluation times (100% success) in bold. For all methods except for RPSS, we draw the results from the references given in Table 5, which are also listed in [18]. Since

Table 4: Global optimization test problems

Fun.	Abbr.	n	$f_{\min}$
Branin RCOS	BR	2	0.3979
Goldstein and Price	GP	2	3.0000
Hartmann	$H_{3,4}$	3	-3.8628
	$H_{6,4}$	6	-3.3224
Rosenbrock	RO_2	2	0.0000
	RO_{10}	10	0.0000
Shekel	$S_{4,5}$	4	-10.1532
Shubert	SH	2	-186.7309

Table 5: The global minimizers

Method	Reference
Genetic and Nelder-Mead (GNM)	Chelouah and Siarry [3]
Genetic with Nelder-Mead (GANM)	Fan et al. [7]
Swarm with Nelder-Mead (SNM)	Fan et al. [7]
Niche hybrid genetic algorithm (NHGA)	Wei and Zhao [16]
Tabu Search and Nelder-Mead (TSNM)	Chelouah and Siarry [4]
Continuous Genetic Algorithm (CGA)	Chenouah and Siarry [2]
Restart and Modified Nelder-Mead (RMNM)	Zhao et al. [18]
Variable Neighborhood Descent (VND)	Toksari and Guner [14]
Basic Variable Neighborhood Search (B-VNS)	Toksari and Guner [14]

some results are not available for some instances, Table 6 contains empty entries. In the last column, we list for each function the average of evaluation number of all the methods 100% succeeded in finding the global minimum.

Table 6: Comparison of RPSS with heuristics listed in Table 5

Fun.	GNM	GANM	SNM	NHGA	TSNM	CGA	RMNM	VND	B-VNS	RPSS	AVE
BR	295	356	230	–	125	–	**60**	372	308	178	241
GP	259	422	304	–	**151**	410	69(80)	294	206	215	283
$H_{3,4}$	492	688	436	–	698	–	**67**	408	521	132	430
$H_{6,4}$	**930**	–	–	–	2638	–	398(50)	2274	1244	2583	1934
SH	345	1009	753	–	279	–	275(40)	–	–	**138**	505
RO_2	459	738	440	239	369	960	**224**	–	–	363	474
RO_{10}	14563(83)	5194	**3303**	6257	–	21563	5946(95)	–	–	6333	8530
$S_{4,5}$	698(85)	2366	850	–	545(69)	610(76)	912(90)	806	**571**	3624	1643

For each function in Table 6, the rate of RPSS for success is 100 percentage. However, except for SH in Table 6, RPSS does not dominate the other functions regarding minimal function evaluation number. To further analyze the performance of RPSS, we list in Table 7 the compared results of function evaluation number with the average (which are listed in the last column of Table 6 and denoted as AVE_i). We denote Eva_{ij} as the function evaluation number of method j on function i, for example, if the method j is GNM and the function i is BR, $Eva_{ij} = 295$, and Com_{ij} as the compared results of Eva_{ij} with AVE_i, which is defined as

$$Com_{ij} = \begin{cases} + & \text{if the rate of successful minimization is} \\ & 100\% \text{ and } Eva_{ij} > AVE_i \\ - & \text{if the rate of successful minimization is} \\ & 100\% \text{ and } Eva_{ij} \leqslant AVE_i \\ U & \text{if the rate of successful minimization is} \\ & \text{less than } 100\% \\ NA & \text{if } Eva_{ij} \text{ is not available} \end{cases}$$

Table 7: Comparison of RPSS with heuristics listed in Table 5

Fun.	GNM	GANM	SNM	NHGA	TSNM	CGA	RMNM	VND	B-VNS	RPSS
BR	+	+	-	NA	-	NA	-	+	+	-
GP	-	+	+	NA	-	+	U	+	-	-
$H_{3,4}$	+	+	+	NA	+	NA	-	-	+	-
$H_{6,4}$	-	NA	NA	NA	+	NA	U	+	-	+
SH	-	+	+	NA	-	NA	U	NA	NA	-
RO_2	-	+	-	-	-	+	-	NA	NA	-
RO_{10}	U	-	-	-	NA	+	U	NA	NA	-
$S_{4,5}$	U	+	-	NA	U	U	U	-	-	+

The advantage of RPSS is obvious by observing the results in Table 7. First, contrasting to GNM and RMNM which also test all functions in Table 6, RPSS exceeds them by 100% of success and by more "−" outcomes; second, for most functions (6 out of 8), RPSS has better results than the average ones, outperforming on average the other methods.

5 Conclusions

In this paper, a modification of Nelder-Mead (NM) algorithm is proposed. Our Parametric Simplex Search (PSS) algorithm integrates all the steps (except *Shrink*) by changing systematically and randomly parameter values in the *general formulation* for one step of original NM iteration. This makes the algorithm capable of searching for the optimum in a diversified and flexible way. The *shrink* step is also revised, in the way that more information of the previous simplex can be inherited.

To make the algorithm be capable of global searching, we restart the PSS algorithm until the termination criterion is met. We call our algorithm as the *Restart and Parametric Simplex Search* NM (RPSS).

We analyze the quality of our RPSS algorithm by conducting comparative analysis with other algorithms. We evaluate the RPSS algorithm from two aspects. First, we analyze the robustness of the algorithm, which is characterized by the deviation of the function values from the optimum when the function dimension changes; second, we evaluate the effectiveness of RPSS, where we propose an index "AGAP", to well evaluate the performance of RPSS. It is shown by computational experiments that, the RPSS algorithm outperforms in average the original version of NM as well as some other recent successful modifications.

Future work may include search for better parameter values in more automatic fashion. Such an extension may be seen as Parametric Space Search approach. In addition our PSS may be used as a local search routine within some metaheuristic scheme, such as Variable Neighborhood Search [12].

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] R. Chelouah and P. Siarry, *A continuous genetic algorithm designed for the global optimization of multimodal functions. Journal of Heuristics* 6 (2000) 191–213.

[3] R. Chelouah and P. Siarry, *Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. European Journal of Operational Research* 148 (2003) 335–348.

[4] R. Chelouah and P. Siarry, *A hybrid method combining continuous tabu search and Nelder-Mead algorithms for the global optimization of multiminima functions. European Journal of Operational Research* 161 (2005) 636–654.

[5] J. Dréo and P. Siarry, *Continuous interacting ant colony algorithm based on dense heterarhy. Future Generation Computer Systems* 20 (2004) 841–856.

[6] J. Dréo and P. Siarry, *Hybrid continuous interacting ant colony aimed at enhanced global optimization. Algorithmic Operations Research* 2 (2007) 52–64.

[7] S.K.S. Fan, Y.C. Liang and E. Zahara, *A genetic algorithm and a particle swarm optimizer hybridized with Nelder-Mead simplex search. Computers and Industrial Engineering* 50 (2006) 401–425.

[8] P. Hansen and N. Mladenović, *Variable neighborhood search. In: Glover F, Kochenberger G (Eds), Handbook of Metaheuristics*. Kluwer: Dordgecht. (2003) 145–184.

[9] A.R. Hedar and M. Fukushima, *Tabu search directed by direct search methods for nonlinear global optimization. European Journal of Operational Research* 170 (2006) 329–349.

[10] F. Herrera, M. Lozano and D. Molina, *Continuous scatter search: An analysis of the integration of some combination methods and improvement strategies. European Journal of Operational Research* 169 (2006) 450–476.

[11] D.G. Humphrey and J.R. Wilson *A revised search procedure for stochastic simulation response surface optimization. INFORMS Journal on Computing* 12 (2000) 272–283.

[12] N. Mladenović and P. Hansen, *Variable neighborhood search. Computers and Operation Research* 24 (1997) 1097–1100.

[13] J.A. Nelder and R. Mead, *A simplex method for function minimization. Computer Journal* 7 (1965) 308–313.

[14] M.D. Toksari and E. Guner, *Solving the unconstrained optimization problem by a variable neighborhood search. Journal of Mathematical Analysis and Applications* 328 (2007) 1178–1187.

[15] V. Torczon, *Multi-directional Search: A direct search algorithm for parallel machines.* PhD thesis, Rice University, Houston, Texas, USA. 1989.

[16] L.Y. Wei and M. Zhao, *A niche hybrid genetic algorithm for global optimization of continuous multimodal functions. Applied Mathematics and Computation* 160 (2005) 649–661.

[17] M.H. Wright, *Direct search methods: once scorned, now respectable.* In *Numerical Analysis*(Griffiths DF and Watson GA, ed.), Addison Wesley Longman, Harlow, United Kingdom (1996) 191–208.

[18] Q.H. Zhao, D. Urošević, N. Mladenović and P. Hansen, *A restarted and modified simplex search for unconstrained optimization. Computers and Operations Research* 36 (2009) 3263–3271.