HYBRID NELDER-MEAD ALGORITHMS FOR NONLINEAR NUMERICAL OPTIMIZATION

I. ZARI[1], R. A. KHANUM[1], S.I.A. SHAH[2], M. A. JAN[3] AND W. K. MASHWANI[3]
[1]Jinnah College for Women, University of Peshawar, KPK, Pakistan
[2]Deparment of Mathematics, Islamia College University, Peshawar
[3]Department of Mathematics, Kohat University of Science & Technology, KPK, Pakistan
E-mails: zari145@yahoo.com, adeeb_maths@yahoo.com,drinayat@icp.edu.pk, majan@kust.edu.pk, mashwanigr8@gmail.com

ABSTRACT: Classical Nelder-Mead Algorithm (NMA) is widely used for local optimization in many domains. It was proposed in 1965; since then a lot of improvements had been made to this optimizer. NMA was also hybridized with other search techniques. We developed very recently a hybrid algorithm Reproductive Nelder-Mead Algorithm (R-NMA), which showed performance improvement over NMA. This paper devises modifications to R-NMA and presents two new hybrid algorithms. In the first hybrid, JADE's mutation strategy is combined with NMA, and the crossover is kept passive. In the second hybrid, only crossover is employed in NMA, and mutation is paused. The proposed hybrids are known as Mutated Nelder-Mead Algorithm (M-NMA) and Crossover-based Nelder-Mead Algorithm (C-NMA). Additionally, this paper investigates the performance of M-NMA and C-NMA against R-NMA and NMA. The new proposed methods have been tested on ten CEC2005 contest test problems. Experimental results show that the performance of NMA is improved by these modifications in terms of solution eminence.

Keywords: Local search, Nelder -Mead Algorithm, hybridization, nonlinear unconstrained numerical optimization.

1. INTRODUCTION

The Local Search (LS) methods start from a point and use the gradient or objective function value to guide the search. Nelder-Mead Algorithm (NMA) [1, 2] is a gradient-free LS method designed for nonlinear optimization. The search done by NMA is based on geometric operators (reflection, expansion, contraction and shrinking) on a current set of points. NMA can find the solution to an optimization problem with more accuracy, if it is started from the points in the neighbourhood of the known optimum [3]. In other words, it is not optimal at global search [3]. Evolutionary Computation (EC) is nature inspired field of optimization. Evolutionary Algorithms (EAs) are stochastic optimizers in EC, which depend upon three basic operators, i.e., selection, mutation and crossover to guide it in searching the solution space [4]. EA is good in exploration, but not as good at exploitation as LS methods [4]. Thus, it makes sense to hybridize NMA with global search techniques to improve its exploration ability.

Inspired by the above pitfalls of optimization methods, we previously merged two evolutionary operators into NMA and proposed R-NMA, which produced encourageing results. In this paper, we modify R-NMA and present two new algorithms to solve the continuous unconstrained optimization problem. These algorithmsare called Mutated Nelder-Mead Algorithm (M-NMA) and Crossover-based Nelder-Mead Algorithm (C-NMA), respectively. M-NMA invokes a local search method and a reproduction operator "mutation", which is borrowed from an efficient EA, JADE [4]. Moreover, stagnation may be achieved due to an inappropriate configuration of mutation strategy and its over exploration [6]. JADE has shown performance improvement over the state-of-the-art algorithms [7, 8, 9] according to the reported results in [4]. That is why, we borrowed its mutation strategy in our new algorithm. The novelty of M-NMA is based on incorporation of mutation operator inside NMA to increase its global search ability.The C-NMA, on the other hand is based on interchanging genes of two subpopulations, and thus producing more diversified offspring for NMA.

The rest of this paper is designed as follows. Section 2 reviews briefly the related work. Section 3 is devoted to the proposed M-NMA and C-NMA. In section 4, M-NMA and C-NMA are used to solve 10 benchmark test problems. In addition, comparisons between the proposed M-NMA, C-NMA and R-NMA and NMA are presented. Finally, Section 5 concludes this paper.

2. RELVIEW

In recent years, a great interest has been devloped for solving optimization problems in the fields of engineering, chemical sciences and economics [10]. In real world optimization problems, quality of a product is considered. The LS methods perform very efficient to solve optimization problems without using their slopes information [11]. Therefore NMA is hybridized with many other huristics, like Genetic Algorithms (GAs) [1] and other Evoloutionary Algorithm (EAs) [12] to improve their search capabilities. Some hybrids of NMA with GA are available in [13, 14, 22]. Some researchers preferd to hybridize NMA with Honey Bees Optimization [15] and Simulated Annealing (SA) [16]. A new method is designed in [17] by combining NMA with statistical process control to deal with stochastic response optimization problems. A meta-heuristic is employed in NMA to overcome drawbacks of its slow convergence in [18]. NMA is also hybridized with artificial intelligence techniques to solve complex optimization problems [17].

Thus, various hybrid algorithms have been proposed in the past to improve NMA's performance for solving unconstrained optimization problems. Some algorithms were good for one type functions, but failed at the others. Other algorithms were good at low dimension and small population size, but faced difficulties in solving high dimension problems with large population size. Indeed, it makes room

for devising new algorithms which can solve complex real world problems with more accuracy.

We have already given the detailed description of NMA and reproduction operators in our previous work R-NMA [5], upon which this work is based. In our previous work, we employed reproduction operators, mutation and crossover together into the simplex to keep a balance between exploration versus exploitation. This combination returned fruitful results. In this work, we invesigate that if only one reproduction operator is hybridised at one time and the other is paused, what would be the effect then on the results? In the next section, we introduce our two new proposed algorithms M-NMA and C-NMA to answer the above question.

3. THE PROPOSED MODIFICATIONS

3.1 MUTATED NELDER-MEAD ALGORITHM (M-NMA)

The NMA is an old but widely used search method to solve various optimization problems. It is mostly intended for function minimization [3]. No doubt NMA has local search ability; however, it is not competent at global search [1]. Our aim is to find global minima for an optimization problem and to improve NMA's efficiency globally. Thus, it would be a good idea to hybridize mutation with NMA to improve its performance, since mutation is efficient in exploration. Thus, we designed a new hybrid algorithm M-NMA, which possesses both the global and local search properties at the same time. The M-NMA is the combination of two techniques, namely NMA and genetic alteration from genetic algorithms. The working algorithm of M-NMA is illustrated as follows.

The process of optimization described by Algorithm 1 starts with creating a sample of $n+1$random points in the search space to form a simplex S. Then, evaluate the simplex, i.e., find $f(S)$. In each iteration re-adjust the indices of S in terms of $f(S)$ values from minimum to maximum. Next, choose randomly three distinct vertices, $x_{r1,g}, x_{r2,g}$ and $x_{i,g}$ from the current simplex to perform mutation as given in the loop, steps 3-10 of Algorithm 1.

Algorithm 1: Pseudo-code of M-NMA.

00 **Control parameters:** n: problem dimension, FES: function evaluations, $\alpha, \beta, \gamma,$ and δ: NMA parameters, where α is coefficients of reflection, β is expansion, γ is contraction, and δ is shrinkage, F: mutation scale factor, and *MaxIter*: max iterations.
01 Choose n, set $FES = 0$, and create a random initial population $\{x_{i,0} \mid i = 1,2, \dots, NP\}$ to generate simplex S.
02 Evaluate S to generate $f(S)$ and set $FES = NP$.
03 **For** $k = 1: MaxIter$
04 Rank the indices of S in terms of $f(S)$ values from best to worst.
05 **For** $i = 1: NP$
07 Generate F_i for each individual i
08 Randomly collect distinct elements $x_{r1,g} \neq x_{r2,g} \neq x_{i,g}$ from the simplex.
09 Generate mutant vector as: $v_{i,g} = x_{i,g} + F_i\left(x^p_{best,g} - x_{i,g}\right) + F_i\left(x_{r1,g} - x_{r2,g}\right)$.
10 Set $S = v_{i,g}$ and find $f(S)$
11 Re-index all individuals in the current S in ascending order of their fitness values, i.e., individual $x_{n+1}$is the worst one and x_1 is the best one, in the sense of minimization.
12 **End for**
13 Apply NMA to exploit the search on these mutant vectors and terminate the current iteration.
14 **End for**
13**Output**: The point with the minimum objective value in the entire optimization.

After that compute a new simplex S on mutant points, then re-order S in ascensing order in terms of fitttnes values. i.e., $x_1 \leq x_2 \leq \cdots \leq x_{n+1}$.

Assume that the elit one is $x_1$and the worst one is x_{n+1}. Compute the centere of mass of points from $x_1$to x_n as:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (1)$$

First, produce the reflection point x_r along with its function value, $x_r = \bar{x} + \alpha(\bar{x} - x_{n+1})$. If $f(x_r)$ is better than the second worst point, then replace worst point, i.e., $x_{n+1}$by new calculated reflection point x_r. Otherwise, if $f(x_r) < f(x_1)$, determine the expansion point,

$x_e = \bar{x} + \beta(x_r - x_{n+1})$. After words, replace the worst point by the better of x_e and x_r. If $f(x_n) \leq f(x_e)$ and $f(x_r) \leq f(x_{n+1})$, then compute the contraction point, $x_c = \bar{x} + \gamma(\bar{x} - x_{n+1})$ if not, then shrink the simplex by halving the distance of vertices from best vertex x_1.

The major difference between proposed method M-NMA and classical NMA is that after the initialization of simplex, M-NMA performs mutation strategy among the vertices of the simplex. In essence, the purpose of mutation operation is to determine the searching direction and the size of the searching area. Thus, offering potential offspring to construct the new simplex, which then undergos reflection, contraction and expansion. In the NMA, there is no mutation mechanism. There the initialized simplex directly goes through reflection, contraction and expansion.

3.2 CROSSOVER-BASED NELDER-MEAD ALGORITHM (C-NMA)

Here, we combine the same NMA with another reproduction operator, known as binomial crossover [23]. The resulting new algorithm is known as Crossover-based Nelder-Mead Algorithm (C-NMA). The novelty of C-NMA lies in its

implementation of crossover in the beginning of optimization. Since C-NMA possesses the ability of both global and local search, it is expected that C-MNA will solve optimization problems better than the classical NMA.The basic idea of C-NMA is that it incorporates the binomial crossover operator [24] between the two subsets of the initial simplex. In this way, it not only preserves the diversity of the simplex, but also helps the algorithm in exploration as well. The formal algorithm of C-NMA is given in Algorithm 2.

After the vertices of the simplex are ranked according to their objective function values from best (minimum) to worst (maximum). Except the best vertex, the remaining n vertices are partitioned into two equal disjoint sub populatios, A and B, as shown in step 7 of Algorithm 2. Crossover operator interchanges the genes of each pair of vertices from populations, A and B as follows:

$$u_{j,i,g} = \begin{cases} a_{j,i,g} & , \ if\ rand(0,1) \le CR_i\ or\ j = jrand \\ b_{j,i,g}, & \text{otherwise,} \end{cases} \quad (6)$$

where $a_{j,i,g} \in$ A and $b_{j,i,g} \in$B.

In this way, n new offspring are produced which are more divercified than their parents. This operation is also illustrated in Algorithm 2 (see steps 8 to 15).

Algorithm 2: Pseudo-code of C-NMA.

00 **Control parameters:** n: problem dimension, FES: function evaluations, $\alpha, \beta, \gamma,$ and δ: NMA parameters, where α is coefficients of reflection, β is expansion, γ is contraction, and δ is shrinkage, CR: crossover probability.
01 Create a uniform and random initial population $\{x_{i,0} \mid i = 1,2, \dots, NP\}$ from the feasible solution space to generate simplex S.
02 Evaluate S to generate $f(S)$ and set $FES = NP$.
03 **For** $k = 1: MaxIter$
04 Rank the indices of $f(S)$ from best to worst.
05 **For** $i = 1: NP - 1$.
06 Generate CR_i for each individual i
07 Partition S, except the best solution into two disjoint subsets A and B.
08 **For** $j = 1: n$
09 Generate $jrand = randint(1, n)$
10 **If** $rand(0,1) \le CR_i\ or\ j = jrand$
11 $u_{j,i,g} = a_{j,i,g}$ where $a_{j,i,g} \in$ A
12 **Else**
13 $u_{j,i,g} = b_{j,i,g}$ where $b_{j,i,g} \in$ B
14 **End If**
15 **End For**
16 Combine the best vertex x_1 and $u_{i,g}$ to form new simplex S.
17 Evaluate S and sort $f(S)$ from best to worst.
18 Calculate centroid of remaining n vertices except the best vertex.
19 Implement NMA to complete the iteration.
20 **End For**
21**End For**
22 **Output:** The point with the smallest objective function value from entire search.

The new population of n offspring is evaluated, and combined with the best vertex to form a new simplex S. This simplex is now reordered from best to worst in terms of the fitness values of its indices. Afterwards, the classical NMA is applied to terminate the current iteration. In C-NMA each iteration starts with our crossover strategy and is terminated by NMA algorithm. In this way a balance between global versus local search is maintained.

4. EXPERIMENTAL STUDY

We have run M-NMA, C-NMA, R-NMA and NMA 25 times independently with a maximum of 200 iterations per run to check the performance of these algorithms.

4.1. Parameters setting

This section describes the control parameters setting which are used in our experimental work of this paper. We keep $n = 30$ and $NP = n + 1$.

Furthermore, we have chosen mutation scalar factor F as suggested in litrature [4]. There are four fixed parameters, $\alpha, \beta, \gamma,$ and δ in NMA, M-NMA, C-NMA and R-NMA. These parameters are chosen as: $\alpha > 0, \beta > 1, 0 < \gamma < 1, 0 < \delta < 1$ as given by [2], [5]. Further, the specific used values are given in Table 1.

Table 1: Parameters values used in this work

Parameters	Parameters Name	Numerical values
F	Starting mutation scale factor	0.5
CR	Starting crossover probability	0.5
α	Coefficient of Reflection	1
β	Coefficient of	2

	Expansion	
γ	Coefficient of Contraction	0.5
δ	Coefficient of Shrinking	0.5

The algorithms M-NMA, C-NMA, R-NMA, and NMA terminate at the maximum number of iterations.

To perform the experimental work, we have taken the first 10 test instances from the CEC2005 test problems [21].These test problems support three types of complicated structures, including rotating the function, shifting the global optimum to a new position, and combining different optimization problems.These test instances are low dimensional, like 10, 30 and 50, single objective and unconstrained.

4.2. Comparison of M-NMA against NMA

For both algorithms, the average error of objective function values, $f(x) - f(x^*)$ are recorded at the termination of each experiment and are presented in Table 2.The best results are typed in bold. The M-NMA has achieved good solutions than NMA on problems, 1, 2, 3, 5, 6, 9 and 10. On 8 both the algorithms have equal performance. However, our proposed algorithm has failed on problems 4; basically 4 is made from 2 by multiplying a factor *"*$(1 + 0.4\ |\mathrm{N}(0,1)|)$*"* to its objective function. Which adds uncertainty/noise to the landscape of the problem. Thus, it may be a cause of its failure. Similarly, M-NMA is failed on 7. In case of 7, the global optima is out of the initializing range.

Table 2 shows that the performance of M-NMA is much better than NMA on 7 test functions out of 10. This better performance is due to the merging of mutation operator into NMA, which is good in global search and it increases the diversity of the population (simplex). Table 2 illustrates that the number of function evaluation used by M-NMA exceed than NMA. Since M-NMA introduces one extra strategy in each iteration, i.e., the strategy of mutation, while NMA does not. Thus, NMA utilized less function evaluations than M-NMA. However, M-NMA found superior solutions than NMA as revealed by the Table 2. Furthermore, our objective was to improve the solution quality, which is achieved, but at a higher computational cost.

4.3. Comparison of C-NMA with NMA

In Table 3 the bold results in the second column evident the fact that C-NMA outperformed NMA in terms of average objective function values, on 8 out of 10 test problems, 1-6, 9 and 10. This superior performance can be attributed to merging of crossover operator into NMA. On one test Problem, 8 both, C-NMA and NMA found the same solution, which is made italic in the same Table. However, C-NMA has failed on 7, this might be due the fact that there is no bound for variable *x*.

4.4. Comparison of M-NMA, C-NMA, R-NMA and NMA

Table 4 shows the results of these algorithms. The comparison of these four algorithms with each other shows that C-NMA is the most superior and robust algorithm than others. The minimum values of each algorithm are printed in bold. From the same table, we can see that C-NMA outperformed than other algorithms. The C-NMA shows good performance on 8 test problems out of 10, i.e., 1-6, 9 and 10. On 8 it shows almost equal results, but on 7 the performance of all algorithms are not satisfactory due to its nature. M-NMA stood second and the third comes R-NMA in this comparison.

For the purpose of fair comparison, we draw box plots of N-NMA, C-NMA, R-NMA and NMA. The red horizontal line in the middle of the box (see Figure 1) shows the median of the data. Since we are solving minimization problems, so if lower is the box, better is the performance of that algorithm. Figure 1 reveals that on 8 test problems, 1-6, 9 and 10 the C-NMA is very proficient. M-NMA is good on 8 and NMA on 7 as is evident from Figure 1.

Table 2: Comparison statistics obtained by M-NMA and NMA

Test Instances	$error = f(x) - f(x^*)$		*FES*		*T / seconds*	
	M-NMA	**NMA**	**M-NMA**	**NMA**	**M-NMA**	**NMA**
1	**8.7104e+04**	8.9700e+04	318565	164089	43.36	31.41
2	**8.2495e+04**	8.6078e+04	318813	165020	91.20	71.01
3	**2.2720e+09**	2.4288e+09	318765	164714	164.74	125.73
4	1.1582e+05	**1.0833e+05**	318391	161667	213.25	157.34
5	**5.9846e+04**	6.7035e+04	317295	164218	258.70	191.83
6	**4.3417e+10**	4.4773e+10	318563	164620	304.57	226.63
7	8.5760e+03	**6.2220e+02**	317207	163569	373.78	278.95
8	*1.1824e+02*	*1.1838e+02*	315517	160948	437.32	321.64
9	**1.8297e+02**	1.9442e+02	317846	163381	482.63	353.42
10	**5.5491e+02**	5.7708e+02	318063	163413	555.17	404.11

5. CONCLUSION

In this paper, we first presented a brief review of work in the field of NMA optimization. Some of the pitfalls of current methods are also identified. Next, we described M-NMA and C-NMA, which combined only one genetic alteration procedure in NMA. In our previous work R-NMA, we

inserted two gentic strategies togather into NMA. Which performed superior than NMA. In this experiment we inlaid two operator separately and studied the effects on the performance of NMA. The following conclusions can be drawn from the experimental results displayed in this paper:

1) In the above results M-NMA and C-NMA, in comparison with NMA, performed better in terms of solution quality on seven and eight out of ten CEC2005 test instances, respectively. M-NMA was worse on two problems as well. The superior performance of M-NMA can be attributed to its exploration skills. However, the reason for the inferiority of M-NMA on two test problems might be the complicated nature of these problems.
2) We gave the comparison of all proposed algorithms, i.e., M-NMA, C-NMA and R-NMA against NMA. The C-NMA performed well on 8 out of 10 representative test problems. So the achievement of C-NMA is more valuable than the other algorithms.
3) The numerical results further unveiled that these combinations are computationally
4) costly than NMA

.Table 2: Comparison statistics obtained by C-NMA and NMA

Test instances	$error = f(x) - f(x^*)$		*FES*		*T/seconds*	
	C-NMA	**NMA**	**C-NMA**	**NMA**	**C-NMA**	**NMA**
1	**8.4023e+03**	8.9700e+04	239752	164089	46.65 sc	31.41sc
2	**5.9293e+04**	8.6078e+04	239784	165020	98.28 sc	71.01sc
3	**3.9200e+08**	2.4288e+09	239756	164714	171.06sc	125.73sc
4	**9.5505e+04**	1.0833e+05	238622	161667	219.47sc	157.34sc
5	**1.2312e+04**	6.7035e+04	239408	164218	267.76sc	191.83sc
6	**2.4027e+09**	4.4773e+10	239650	164620	316.47sc	226.63sc
7	2.2266e+03	**6.2220e+02**	237526	163569	382.61sc	278.95sc
8	*1.1833e+02*	*1.1838e+02*	235663	160948	441.86sc	321.64sc
9	**38.1242**	1.9442e+02	239812	163381	491.19sc	353.42sc
10	**57.3365**	5.7708e+02	239509	163413	562.69sc	404.11sc

Table 4: ***Experimental results of M-NMA,C-NMA,R-NMA and NMA over 25 independent runs along with function evaluations and time elapsed.***

Test instances	$error = f(x) - f(x^*)$			
	M-NMA	**C-NMA**	**R-NMA**	**NMA**
1	8.9700e+04	**8.4023e+03**	8.3608e+04	8.9700e+04
2	8.6078e+04	**5.9293e+04**	8.1990e+04	8.6078e+04
3	2.4288e+09	**3.9200e+08**	2.2132e+09	2.4288e+09
4	1.0833e+05	**9.5505e+04**	1.0637e+05	1.0833e+05
5	6.7035e+04	**1.2312e+04**	5.6205e+04	6.7035e+04
6	4.4773e+10	**2.4027e+09**	4.0835e+10	4.4773e+10
7	**6.2220e+02**	2.2266e+03	9.5902e+03	**6.2220e+02**
8	1.1838e+02	1.1833e+02	1.1837e+02	1.1838e+02
9	1.9442e+02	**38.1242**	1.8251e+02	1.9442e+02
10	5.7708e+02	**57.3365**	5.1495e+02	5.7708e+02

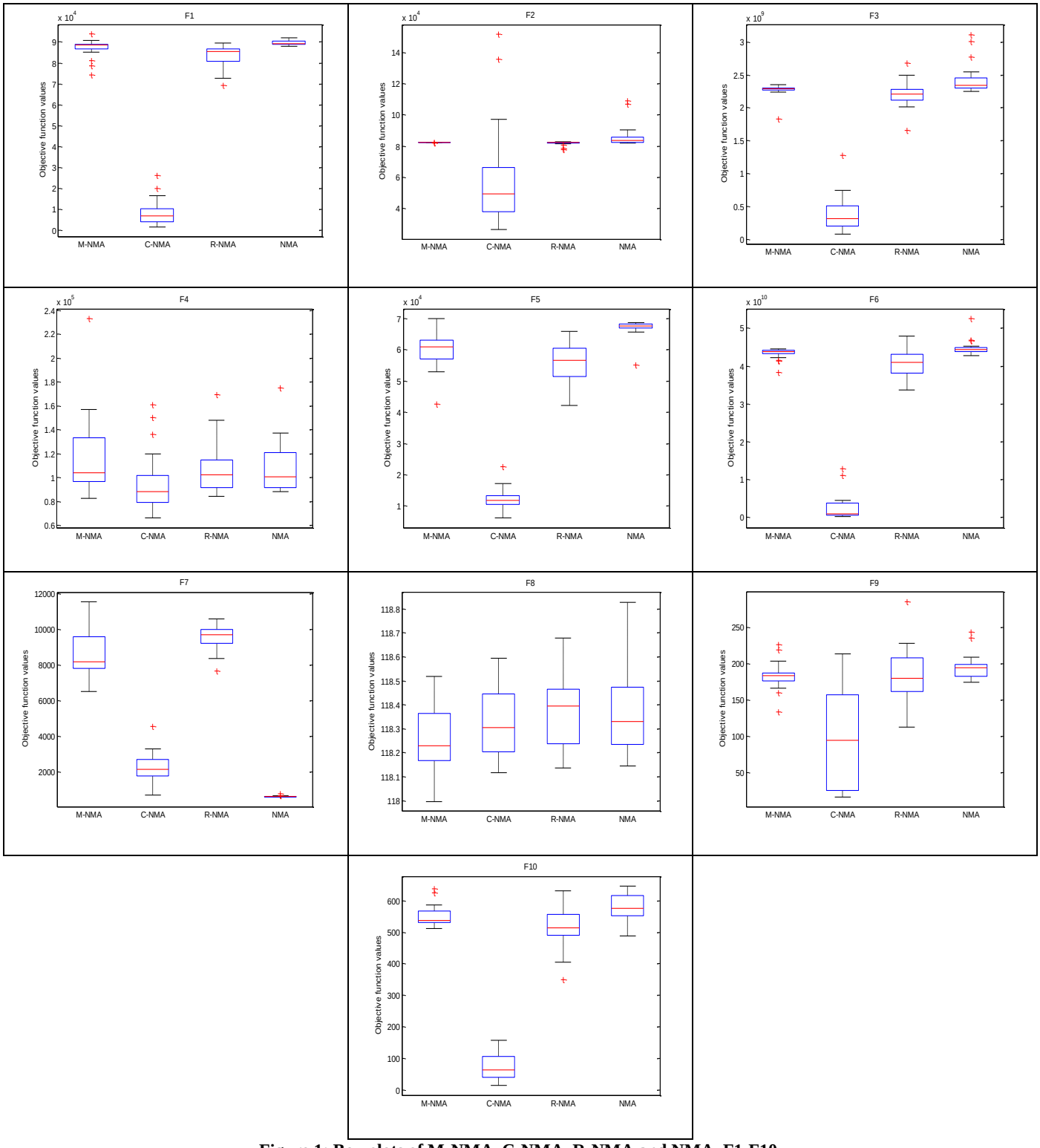

Figure 1: Box plots of M-NMA, C-NMA, R-NMA and NMA F1-F10.

REFERENCES

[1] Dabiri F., Jafari R. and Nahapetain A., A unified optimal voltage selection methodology for low-power systems, IEEE Transactions on Evolutionary Computation, 210-218, 2007.

[2] Fletcher R., Practical Methods of Optimization, Wiley, 1987.

[3] Venkataraman P., Applied optimization with MATLAB programming, John Wiley and Sons, 2009.

[4] Zhang J. and Sanderson A. C., JADE: adaptive differential evolution with optional archive, IEEE

Transactions on Evolutionary Computation, 13(5): 945–958, 2009.

[5] Khanum R. A., Zari A., Jan M.A. and Mashwani W. K. Reproductive Nelder-Mead Algorithm for single objective unconstrained global optimization, Science International Journal, (accepted 2015).

[6] Tang L., Dong Y., and Liu J., Differential evolution with an individual-dependent-mechanism, IEEE, 2014.

[7] Abbass H. A., The self-adaptive Pareto differential evolution algorithm**,** In Proceeding Congress of Evolutionary Computation, 831- 836, 2002.

[8] Qin A. K., Huang V L., and Sughanthan P. N., Differential evolution algorithm with strategy adaptation for global numerical optimization, IEEE Transactions on Evolutionary Computation, 13(5):398–417, Oct. 2009.

[9] Storn R. and Price K. V., Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces**,** Journal of Global Optimization, 11(4):341–359, 1997.

[10] Aihong R. and Wang Y., A Hybrid Estimation of Distribution Algorithm and Nelder- Mead Simplex Method for Solving a Class of Nonlinear Bilevel Programming Problems, Journal of Applied Mathematics, Vol 2013, 2013.

[11] Sasadhar B. and Indrajit M., Performance Analysis of Nelder-Mead and a Hybrid Simulated Annealing for Multiple Response Quality Characteristic Optimization, Citeseer, 2010.

[12] Fan K. S. and Erwie Z., Simulation optimization using an enhanced Nelder-Mead simplex search algorithm**,** The Fifth Asia-Pacific Industrial Engineering and Management Systems Conference, Citeseer, 12 - 15, 2004.

[13] Heder A. R., Studies on Metaheuristics for Continuous Global Optimization Problems, Ph.D. dissertation, University of Tokyo, Japan, Citeseer, 2004.

[14] Nelder J. A. and Mead R., A simplex method for function minimization, *Comput. J.*, vol. 7, 308–313, 1965.

[15] Sun J. and Zhang Q., *DE/EDA:* A new evolutionary algorithm for global optimization**,** Elsevier, 169 (3): 249 – 262, 2005.

[16] Andrew L. and David A., *RSCS:* A Parallel Simplex Algorithm for the Nimrod/O Optimization Toolset**,** Scientific Programming, 14 (1): 1-11, 2006.

[17] Nicolas D. and Jean A. M., A Combined Nelder-Mead Simplex and Genetic Algorithm, Genetic and Evolutionary Computation Conference, 921- 928, 1999.

[18] Yen J., Liao J. C., Randolph D. and Lee B., A hybrid approach to modeling metabolic systems using genetic algorithm and simplex method, in Proc. 11th IEEE Conf. Artificial Intelligence Applications *(CAIA95)*, Los Angeles, CA, Feb. 1995, 277–283.

[19] Schwefel H. P., Numerical optimization of computer models, John Wiley and Sons,1981.

[20] David W. F. and Robert A P., the Box plot: A Simple Visual Method to Interpret Data, Annals of internal medicine, 110(11): 916-921, 1989.

[21] Suganthan P. N., N. Hansen J., Liang J., Deb K., Chen Y. P., Auger A. and Tiwari S., Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real- Parameter Optimization, KanGAL Report, Vol May 2005.

[22] Moraglio A. and Togelius J., Geometric Nelder-Mead Algorithm for Permutation Representation, IEEE, 1-8, 2010.

[23] Rahami H., Kaveh A., Aslani M., and Asl R. N., A hybrid modified genetic Nelder- Mead simplex algorithm for large-scale truss optimization, International Journal of Optimization and Civil Engineering, 1: 29 - 46, 2011.

[24] Ballester P. J. and Carter J. N., An effective real parameter genetic algorithm with parent centric normal crossover for multimodal optimization. Lecture Notes in Computer Science, Springer, 901- 913, 2004.