2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Non-Uniform Mapping in Real-Coded Genetic Algorithms

Dhebar Yashesh*, Kalyanmoy Deb† and Sunith Bandaru‡
*Department of Mechanical Engineering, Indian Institute of Technology Kanpur, India,
Email: yddhebar@iitk.ac.in
†Department of Electrical and Computer Engineering, Michigan State University, East Lansing, USA
Email: kdeb@egr.msu.edu
‡Virtual Systems Research Centre, University of Skövde, Skövde, Sweden,
Email: sunith.bandaru@his.se

COIN Report Number 2014010

Abstract*—Genetic algorithms have been used as an optimization tool using evolutionary strategies. Genetic algorithms cover three basic steps for population refinement** *selection, cross-over and mutation. In normal Real-coded genetic algorithm(RGA), the population of real variables generated after population refinement operations, is used for the computation of the objective function. In this paper we have shown the effect made by mapping the refined population towards better solutions and thereby creating more biased search. The mapping used is non-uniform in nature and is the function of the position of the individual w.r.t. the best solution obtained so far in the algorithm, and hence the name Non-Uniform RGA or in short NRGA. Tests were performed on standard benchmark problems. The results were promising and should encourage further research in this dimension.**

I. Introduction

Genetic Algoriths (GAs) have been one of the powerful techniques for doing optimization via evolutionary computation. Genetic Algorithm is an evolutionary based strategy where the population is initialized randomly and then this population is modified by the use of certain operations to get new *better* population. These operations include - *selection, crossover and mutation*. Different schemes have been proposed in literature for performing selection. In this paper we have done selection via *tournament selection operator* where the tournament is performed between certain number of randomly selected individuals (2 in our case) and the better one is chosen for crossover. The crossover was made using SBX cross-over operator proposed by Deb. K [1] and then the obtained population was passed on for performing mutation which made minor alteration in the population by modifying certain individuals (the amount of which is controlled by the mutation probability p_{mut}.)

The above discussion on Genetic Algorithms was concerned with only the mentioned three operations. We in this paper have introduced a *non-uniform mapping operator* for making further modification in the population obtained after mutation. This operator which aims at pushing the population towards the better solution obtained so far and thereby making the search more biased as compared to the usual Real Coded Genetic Algorithms. Earlier this strategy was studied with Binary Coded GAs [3]. Figure 1 shows the work flow of RGA and NRGA:

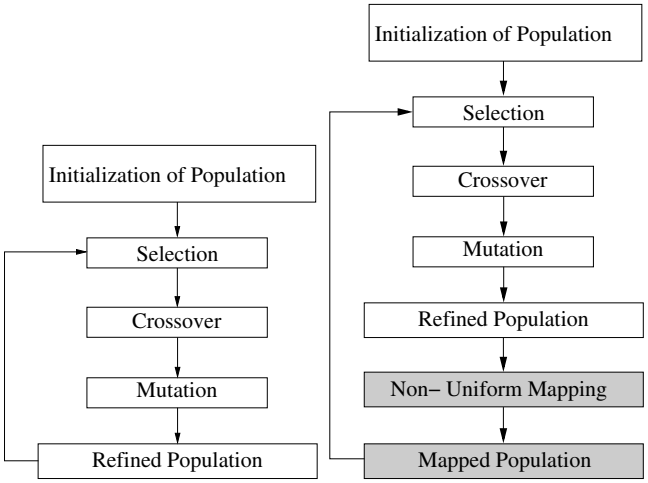


Fig. 1: Workflow of RGA and NRGA

In the remaining part of the paper we have explained the mapping strategy, the new parameter involved as a result of non uniform mapping, and its effect on the performance of the algorithm.

We performed the experiment on 5 bechmark problems: Sphere, Elipsoidal, Ackley, Schwefel and Rosenbrock. The results of different mapping-techniques were compared. Finally the runs for problems prescibed in the problem suit of CEC-14 Real-Parameter Numerical Optimization [2] were performed and the tabulated results for it are indicated in the end.

II. Previous Works

The previous works using the non-uniform *mapping* approach for generating population are observed in Binary Coded Genetic Algorithms. In case of Real Coded Algorithms, the

research in this direction is very less. Some of the notable contributions are as follows:

ARGOT(Adaptive Representation Genetic Optimizer Technique) [5] aimed at adaptively mapping the binary strings(variables) to the decoded real(variables). It used several environmentally triggered operators for altering intermediate mappings which were based on internal measurements such as parameter convergence, parameter variance and parameter positioning within the possible range of parameter values. Another work was DPE (Dynamic Parameter Encoding) [6] which was a smart search and domain control technique. The algorithm had two levels. In first level it was aimed to get in the vicinity of the optimal solution as fast as possible by making alterations in the most significant bits. After achieving it, the population entered the second level of algorithm wherein the most significant bits were dropped and new bits were introduced for getting more precision. In case of delta-coding algorithm [7], the best solution of the previous run was used as reference. The population was reinitialized and a seperate substring coding was used for each parameter as a representation of distance Δ from the corresponding parameter of the best solution mentioned earlier. Thus, a hypercube was formed around the best solution of previous generation the size of which was controlled by adjusting the number of bits used for encoding.

III. The Non-Uniform Mapping

We will explain the mathematical model of the non-uniform mapping for 1D case and then extend it to the n-dimensional space. From the statistical data, we have with us the *best so far solution* obtained as $x^{b,t}$(where b denotes the *best* and t is the *current iteration nunber*). The individual is bounded between values a and b (with a as lower bound (x_L) and b as upper bound (x_U)). The location of the individual after performing three basic operations(selection, cross-over and mutation) is x.

The mapping function used is:

$$m(\zeta) = k\zeta^{\eta}, \tag{1}$$

where $\zeta = (x-a)/(b-a)$. Using this mapping, we map x to x^*(which is nearer to the best ever solution $x^{b,t}$) as shown in the figure below.

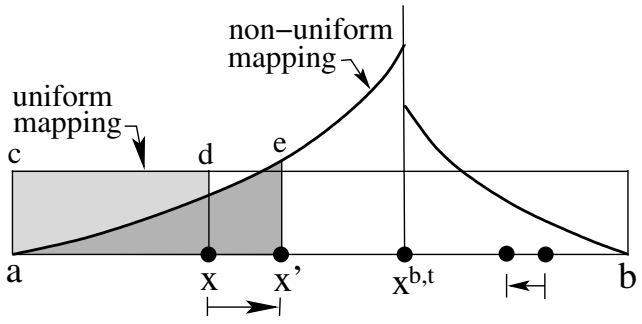


Fig. 2: The Non-Uniform Mapping

This pushing of individuals towards the best so far point via the use of the mapping function of Eq. 1 is done by equating the areas under the graph as:

$$\int_0^{(x^*-a)/(b-a)} m(\zeta)d\zeta = (x-a)/(b-a), \tag{2}$$

for $x \in (a, x^{b,t})$. The value of k used in Eq. 1 is determined by setting x and x^* in above equation to $x^{b,t}$ (as the point $x^{b,t}$ must remain stationary). From this we obtain $k = (\eta+1)\left(\frac{b-a}{x^{b,t}-a}\right)^{\eta}$ and finally substituting this value of k in Eq. 2, the mapped value x^* is :

$$x^* = a + [(x-a)(x^{b,t}-a)^{\eta}]^{\frac{1}{\eta+1}}, \tag{3}$$

Similarly, we can get the expression for x^* when $x \in (x_b, b)$.

IV. Vector-wise mapping in N-Dimensional Space

Previous section dealt with the one-dimensional scenerio. We now extend our discussion on how to handle the n-dimensional case. The point x used earlier will now be of the vector form $\boldsymbol{X}$ with its coordinates as x_i (where $i = 1...n$). We implemented 2 methods of mapping. First was the *Variable-wise* mapping approach where the components of $\boldsymbol{X}$ (i.e. x_i) were pushed towards the corresponding components of $\boldsymbol{X^{b,t}}$(i.e. ${x^{b,t}}_i$) just like in 1D case, i.e.

$${x_i}^* = x_{iL} + [(x_i - x_{iL})({x^{b,t}}_i - x_{iL})^{\eta}]^{\frac{1}{\eta+1}}$$

The second method implemented was *Vector-wise Mapping*, which is shown in Fig. 3 pictorially for 2D case:

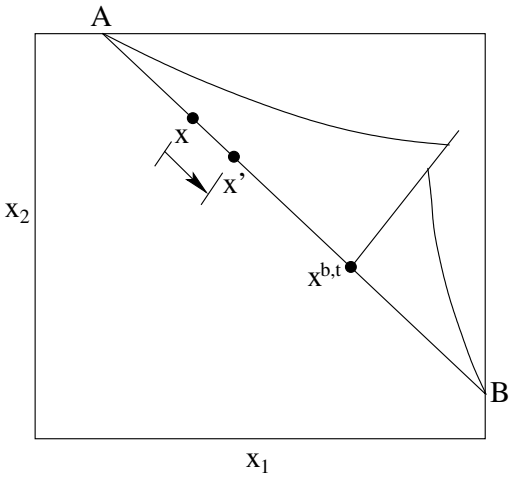


Fig. 3: The Non-Uniform Vector-wise Mapping for 2D Case

The strategy adopted here was first to extend the vector $\mathbf{X} - \mathbf{X^{b,t}}$ in both the directions so that it intersects the *hyper-boundary* enclosing the domain of $\mathbf{X}$ at points $\mathbf{A}$ and $\mathbf{B}$ (with $\mathbf{X} \in (\mathbf{A}, \mathbf{X^{b,t}})$). The line segment between $\mathbf{A}$ and $\mathbf{X^{b,t}}$ is then parameterized with parameter d. The value of d for the corresponding points of interest is tabulated in Table I.

The mapping is then performed using usual 1D mapping equation (Eq. 3) The parameterized value of the mapped point $\mathbf{X}^*$ is d^* and we arrive at it via following equation:

$$\begin{aligned} d^* &= a + [(-a)(1-a)^{\eta}]^{\frac{1}{\eta+1}} \\ \mathbf{X}^* &= \mathbf{X} + d^*(\mathbf{X^{b,t}} - \mathbf{X}) \end{aligned} \tag{4}$$

TABLE I: Values of parameter d (reducing n-Dimensional mapping to a 1D mapping)

Point of Interest	d	Description
$\mathbf{X}$	0	Current Point
$\mathbf{X^{b,t}}$	1	Best so fart Point
$\mathbf{A}$	a	negative value
$\mathbf{B}$	b	positive value

V. HANDELING η

The amount of pushing done and the speed of convergence depends on the value of non-uniform mapping parameter η. Several strategies were adopted to study the effect caused by η on the performance of the algorithm. It is clearly noticed that higher the value of η, more will be the pushing done and so our solutions will start accumulating nearer and nearer to the best ever point ($\mathbf{X^{b,t}}$). If the value of η is set too large right from the start, then the amount of *exploration* will be reduced and there will be higher chances of getting premature convergence.

Keeping this in mind, we made the gradual increase in η with a constant rate ($\eta = generation * rate$). To reduce the chances of premature convergence, we performed the following check:

initialize: count = 0
if *generation%5 = 0* **then**
  if $\|$*fbestprev - fbestever*$\|$ $< \epsilon$ **then**
    count += 1;
    if *count == 3* **then**
      randomize the population;
      count = 0;
    end
  end
end

Algorithm 1: Strategy 1 for reducing premature convergence (where $fbestprev$ is the *best evaluaded value of function from the population of previous generation* and $fbestever$ is the *best function value evaluated so far*. The threshold value ϵ was set to 0.001)

We performed the experiments on five benchmark problems listed below:

$$\text{Sphere:} \quad f(\mathbf{x}) = \sum_{i=1}^{n} x_i^2, \tag{5}$$

$$\text{Ellipsoidal:} \quad f(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2, \tag{6}$$

$$\text{Ackley:} \quad f(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, \tag{7}$$

$$\text{Schwefel:} \quad f(\mathbf{x}) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_i\right)^2, \tag{8}$$

$$\text{Rosenbrock:} \quad f(\mathbf{x}) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (1 - x_i^2)\right]. \tag{9}$$

All problems were evaluated for $n = 20$ variables (i.e. 20 dimensional space). Following parameter values were kept fixed for all runs:

- Population size = 100,
- Lower Bound = -10, Upper Bound = 10,
- Selection type: Tournament Selection,
- Crossover: Simulated Binary Crossover(SBX) with crossover probability = 0.9,
- Mutation: Bitwise mutation with mutation probability = 0.05
- SBX and Mutation paramters: $\eta_c = 2$, $\eta_m = 50$

The algorithm was terminated on achieving the accuracy of 10^{-2} OR when 3000 iterations (generations) were done. The η was updated linearly as below:

$$\eta(t) = \frac{t}{t_{max}}\eta_{max} \tag{10}$$

where $t_{max} = 3000$. η_{max} governed the rate of increase in η with generations and the values used for η_{max} were : 0, 20, 50, 75, 100 and 500 (value 0 means usual RGA). Total 50 runs were perfomed for each problem and for both strategies of mapping, i.e. variable-wise and vector-wise. The results obtained for variable wise mapping case are tabulated as follows in tables Tab: II to Tab: VI. In case of successful runs(S)(i.e. on achieving the desired accuracy) , the statistical data for number of Function Evaluations(FE) is represented while in case of failure(F) (i.e when the desired accuracy was not obtained), the statistical data of final objective value is shown.

TABLE II: Results of Variable-wise mapping scheme for the Sphere function.

Method	η_{max}	FE/f	S or F	min	median	max
RGA	0	FE	S = 50	6701	8901	11801
NRGA	20	FE	S = 50	3201	4001	5201
NRGA	50	FE	S = 50	2801	3501	4301
NRGA	75	FE	S = 50	2601	3401	4901
NRGA	**100**	**FE**	**S = 50**	**2,401**	**3401**	**4701**
NRGA	500	FE	S = 50	3501	5501	8801

TABLE III: Results of Variable-wise mapping scheme for the Ellipsoidal function.

Method	η_{max}	FE/f	S or F	min	median	max
RGA	0	f	F= 50	0.1581	0.2883	0.3739
NRGA	20	FE	S = 50	7701	8901	10001
NRGA	50	FE	S = 50	6301	7401	8201
NRGA	75	FE	S = 50	5401	6901	8201
NRGA	**100**	**FE**	**S = 50**	**5601**	**6701**	**8001**
NRGA	500	FE	S = 50	5901	9901	19901

From the results it is evident that the algorithm give relatively good performance for $\eta_{max} = 75$, i.e. when the rate of increase in η is $75/3000 = 0.025/generation$. But it is also true that $\eta_{max} = 75$ alone does not produce the *b*est results. The power of the non-uniform mapping approach is clearly visible by observing the plots in Fig. 4 - Fig. 8. The plots are plotted for the best run in RGA V/s the best run in NRGA with $\eta_{max} = 75$. Clearly, NRGA showed the fast rate of convergence. We also conducted the experiments

TABLE IV: Results of Variable-wise mapping scheme for the Ackley function.

Method	η_{max}	FE/f	S or F	min	median	max
RGA	0	f	F = 50	0.1533	0.2247	0.2603
NRGA	20	FE	S = 50	10401	12201	14101
NRGA	50	FE	S = 50	7801	9501	11101
NRGA	**75**	**FE**	**S = 50**	**7001**	**9001**	**10501**
NRGA	100	FE	S = 46	6901	8801	16001
		f	F = 4	1.4235	1.6462	1.6462
NRGA	500	FE	S = 1	8401	8401	8401
		f	F = 49	1.155	2.5799	3.5742

TABLE V: Results of Variable-wise mapping scheme for the Schwefel function.

Method	η_{max}	FE/f	S or F	min	median	max
RGA	0	f	F = 50	0.9945	1.643	3.0541
NRGA	20	FE	S = 50	39501	52401	64601
NRGA	50	FE	S = 50	33001	49301	66101
NRGA	**75**	**FE**	**S = 50**	**40601**	**50501**	**69401**
NRGA	100	FE	S = 50	32301	51901	68601
NRGA	500	FE	S = 43	66401	174401	271501
		f	F = 7	0.0259	0.174	0.8787

with vector-wise mapping and it was observed that $\eta_{max} = 75$ value gave better convergence than other values of η_{max}. Table VII shows the comparison for vector-wise mappping case and variable-wise mapping case with $\eta_{max} = 75$. Same parameter values were used for variable-wise and vector-wise mapping case.

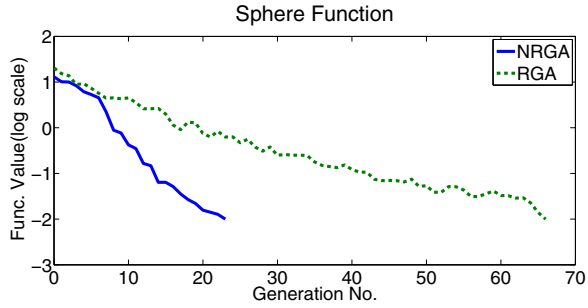


Fig. 4: Sphere Function

In case of sphere function (Fig. 4) it can be clearly seen that both RGA and NRGA showed same trend initially and infact it has to so, because for initial generations we have lower value of η(which is increasing at the rate of $0.025/generations$) and thus the mapping for both RGA and NRGA is approximately same. But later, rate of convergence for NRGA increases, thereby showing the effect caused by the non-uniform mapping.

TABLE VI: Results of Variable-wise mapping scheme for Rosenbrock function.

Method	η_{max}	FE/f	S or F	min	median	max
RGA	0	f	F = 50	21.7009	23.634	90.3395
NRGA	20	FE	S = 1	204101	204101	204101
		f	F = 49	0.0313	9.1326	14.8271
NRGA	50	f	F = 50	0.0445	6.9445	12.6175
NRGA	75	f	F = 50	0.147	5.8189	17.1752
NRGA	100	f	F = 50	0.0668	5.3279	12.3613
NRGA	500	f	F = 50	0.0107	8.3114	18.2916

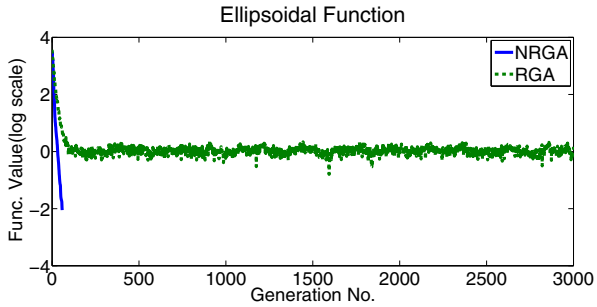


Fig. 5: Ellipsoidal Function

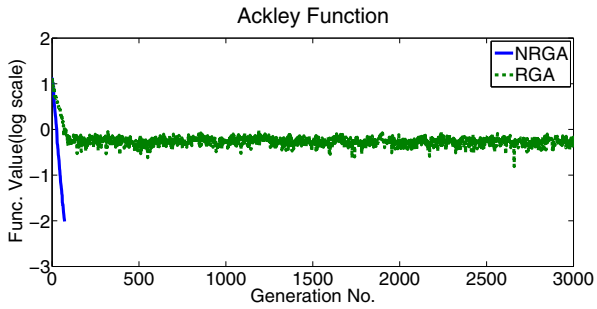


Fig. 6: Ackley Function

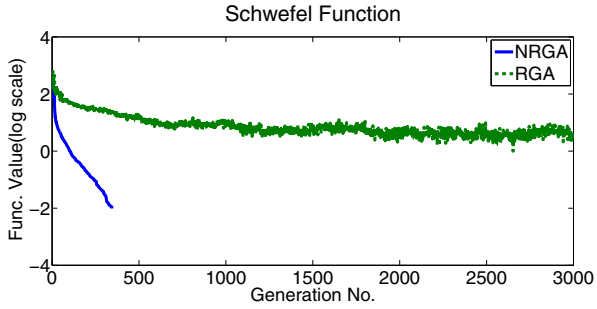


Fig. 7: Schwefel Function

The solutions in case of ackley, elipsoidal and schwefel improved monotonically (Fig. 5 - 7) for NRGA, while population was driven towards premature convergence for these objective functions in case of RGA.

The rosenbrock function is the one where we encounter *flat* zones (Fig.9) thereby making the algorithm prone to premature convergence. It is clearly observed that the plot for NRGA (Fig. 8) remained flat for considerable number of iterations thereby showing that we got stuck at some point and subsequent generations failed to generate better individuals. Here is what seemed to have happened:

- Let us call the point which was considered as the best one for several generations as ${x_{stk}}^b$
- Now the amount of pushing done is dependent on the value of η(which is $0.025 * gen$)

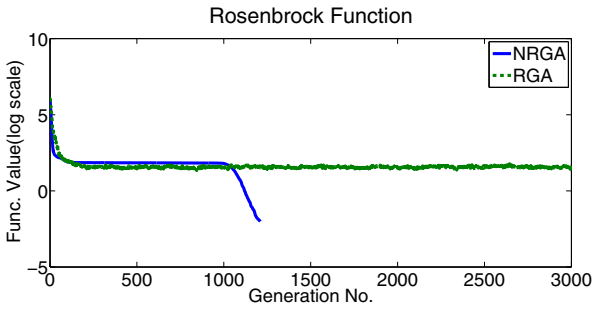
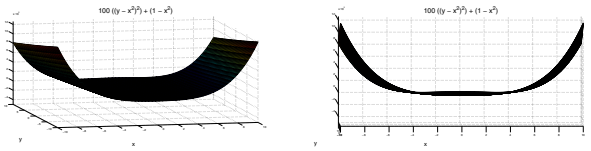


Fig. 8: Rosenbrock Function

- If the value of η is high, then even after reinitialization, the mapped population (Eq. 4) in next iteration would gather around the $\mathbf{X_{stk}}^{\mathbf{b}}$ and so the exploration is reduced.
- Hence the population will get accumulated near to the ${x_{stk}}^b$ until we get an individual which is better than ${x_{stk}}^b$.

But on its subsequent journey, the plot changes its trend of maintaining the flat nature and becomes monotonically decreasing. This thing is realised as an account of reinitialization of population made (thereby creating diversity and hence a better individual) in case of premature convergence as indicated in Algo. 1.



(a) (b)

Fig. 9: Rosenbrock Function

TABLE VII: Comparison between Vector-wise mapping and Variable-wise mapping with $\eta_{max} = 75$

Function	Mapping	FE/f	S or F	min	median	max
Sphere	Variable	FE	S = 50	2601	3401	4901
	Vector	FE	S = 50	2401	3501	4301
Elipsoidal	Variable	FE	S = 50	5401	6901	8201
	Vector	FE	S = 50	5901	6901	9001
Ackley	Variable	FE	S = 50	7001	9001	10501
	Vector	FE	S = 50	7301	8901	13901
Schwefel	Variable	FE	S = 50	40601	50501	69401
	Vector	FE	S = 50	34601	48301	64201
Rosenbrock	Variable	f	F = 50	0.147	5.8189	17.1752
	Vector	FE	S = 1	121201	121201	121201
		f	F = 49	0.0797	5.8716	13.6898

Table VII gives us an idea that vector-wise mapping strategy has tendency to give better results as compared to the variable-wise mapping.

VI. Another Apporoach for Reducing Premature Convergence

It is evident from the results that for the fuctions like rosenbrock, our strategy didn't performed so well. The algorithm got stuck at one of these optima and thereby didn't give good results. Problems of similar nature were asked to solve as as part of CEC-14 problem suit on Real-Parameter optimizaton [2] and hence there was a need to make slight modification in our previous strategy in order to reduce premature convergence (Algo 1).

In the concluding portion of last section we addressed some points focusing on the point $\mathbf{X_{stk}}^b$ where our algorithm got stuck. The major catch for premature convergence was the "larger value of η" which made the population rush towards $\mathbf{X_{stk}}^{\mathbf{b}}$ thereby reducing the exploration.

To tackle this issue, apart from reintializing population we also updated η back to 0 and increased it gradually. Algorithm 2 gives the overview of the modification done.

initialize: count = 0, dec = 0
dec is incremented by 1 with every generation and:
if *generation%5 = 0* **then**
  if $\|$*fbestprev - fbestever*$\|$ $< \epsilon$ **then**
    count += 1;
    if *count == 3* **then**
      randomize the population;
      count = 0;
      dec = 1;
    end
  end
end
$\eta = dec * 0.025$ //Updation in η

Algorithm 2: Strategy 2 for reducing premature convergence

The results of this approach were promising as compared with the Strategy 1. It is to note that for lower dimensions, Strategy 1 worked well while for higher dimensions the Strategy 2 dominated. Experiment to compare these two approaches was conducted for 3 problems - *ackley, schwefel and rosenbrock*. The parameters modified were:

- Population size = 200
- Lower Bound = -100, Upper Bound = 100,
- Accuracy desired = 10^{-8}
- Maximum no. of Function Evaluations = 200000

Total 51 runs were performed and the mapping used was vector-wise mapping. Table VIII shows the resutls obtained.

VII. Final Experiments and Results

As per the directives of CEC-14 competition on *Real-Parameter Numerical Optimization, Problem suit A*[2], we ran the algorithm on specified 30 objective functions. Runs were performed for $10D$, $30D$, $50D$ and $100D$ case (where D denotes dimension). The values of key NRGA parameters taken are specified below:

- Population size = 10*D
- Lower Bound = -100, Upper Bound = 100,
- Selection type: Tournament Selection,
- Crossover: Simulated Binary Crossover(SBX) with crossover probability = 0.9,
- Mutation: mutation probability = 0.05

TABLE VIII: Comparion between Strategy 1 and Strategy 2 For reducing premature convergence

Func.	Strategy	Best	Worst	Median	Mean	Std. Dev.
Ackley	2	$5.0349e-04$	$2.0000e+01$	$1.9995e+01$	$1.6072e+01$	$8.0160e+00$
	1	$8.0000e+01$	$9.9999e+01$	$8.0005e+01$	$8.3928e+01$	$8.0160e+00$
Schwefel	2	$1.6836e-03$	$2.0579e-02$	$7.5771e-03$	$8.1766e-03$	$3.5335e-03$
	1	$8.4305e-05$	$1.5288e-03$	$3.7390e-04$	$4.1472e-04$	$2.4136e-04$
Rosenbrock	2	$1.2500e+00$	$6.7028e+01$	$1.2842e+01$	$1.1975e+01$	$9.1913e+00$
	1	$5.4141e+01$	$3.4689e+03$	$1.1020e+03$	$1.2267e+03$	$8.9078e+02$

- SBX and Mutation paramters: $\eta_c = 2$, $\eta_m = 50$

The vector-wise mapping approach was used and the rate of increase in η was set to $0.025/generation$. The issue of premature convergence was handled using the Algo 2. The accuracy level desired was 10^{-8}. The code was terminated when number of function evaluations crossed the value of $10000 * D$ or the desired accuracy was achieved.

Total 51 runs were performed for each problem and the best, worst, mean and standard deviation were recorded. The tabulated data of the results is mentioned below in tables Tab:X - Tab:XIII:

The Algorithm Complexity was determined as per the instruction given in the report [2]. The complexity of algorithm for $10D$, $30D$ and $50D$ problem is shown in Tab:IX. The computing system used for running the code was *Ubuntu Version 12.04, 32 bits with intel - i5 processor and 4GB RAM*. C-Language was used for coding the algorithm.

TABLE IX: Algorithm Complexity: $T0$, $T1$ *and* $\hat{T2}$ *are time in seconds*

Dimention	$T0$	$T1$	$\hat{T2}$	$(\hat{T2}-T1)/T0$
$D=10$	0.11	0.26	1.15	8.09
$D=30$	0.11	1.42	3.61	19.91
$D=50$	0.11	3.85	95	828.64

VIII. Conclusion and Future Work

In this paper a mapping strategy was proposed for making the exploration more biased towards the better solutions. Two mapping approaches were used - *Variable-wise* and *Vector-wise*. The mapping introduced a new parameter in our algorithm (η) which changed dynamically and its *rate of change* had crucial impact on the performance of the algorithm. Satisfactory results were obtained for contant rate of increase of η which was $0.025/generation$. The mapping approach proved to be fruitful as it gave the fast convergence as compared to normal RGA.

Yet, from the obtained results, the obsevation was made that for the problems with heigher complexity such as rosenbrock (which has flat terrains as indicated in Fig. 9), the approach developed was not so effective. Hence strategies (Algo 1 and Algo 2) were introduced for controlling the diversity of population and thereby avoiding premature convergence. Observation was made that on moving from lower dimesions to higher dimensions problems the performance of Algo 1 degraded while that of Algo 2 improved. Finally, the algorithm was tested with the problem suit of CEC-14 Real Paramter Numerical Optimization [2] and the results were tabulated. The complexity of algorithm was also tested which showed that on moving for higher dimensions, the algorithm speed became an issue.

As it was observed that each method suggested in this paper had its own strong sides and weak sides, the further research is encouraged to develop hybrid strategies. This work can also be used with niching techniques for tackling the problems of multi-modal optimization.

References

[1] Deb, Kalyanmoy, and Ram Bhushan Agrawal. "Simulated binary crossover for continuous search space." *Complex Systems* 9 (1994): 1-34.

[2] J. J. Liang, B-Y. Qu, P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization", **Technical Report 201311**, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, December 2013.

[3] Deb, Kalyanmoy, Yashesh D. Dhebar, and N. V. R. Pavan. "Non-Uniform Mapping in Binary-Coded Genetic Algorithms." *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*. Springer India, 2013.

[4] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: MIT Press, 1975.

[5] C. G. Shaefer. The argot strategy: Adaptive representation genetic optimizer technique. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 50–58, 1987.

[6] N. N. Schraudolph and R. K. Belew. Dynamic parameter encoding for genetic algorithms. Technical Report LAUR90-2795, Los Alamos: Los Alamos National Laboratory, 1990.

[7] D. Whitley, K. Mathias, and P. Fitzhorn. Delta coding: An iterative search strategy for genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 77–84. San Mateo, CA: Morgan Kaufmann, 1991.

TABLE X: Results for 10D

Func.	Best	Worst	Median	Mean	Std. Dev.
1	$1.0770e+03$	$1.2128e+05$	$1.8323e+04$	$2.7905e+04$	$3.1272e+04$
2	$4.7595e-05$	$5.3168e+03$	$5.1965e+02$	$9.1466e+02$	$1.1026e+03$
3	$3.1679e-01$	$6.2498e+03$	$1.2087e+03$	$1.5168e+03$	$1.5166e+03$
4	$2.4164e-03$	$3.4780e+01$	$4.3356e+00$	$1.5436e+01$	$1.7052e+01$
5	$2.0646e-04$	$2.0000e+01$	$2.0000e+01$	$1.9607e+01$	$2.8004e+00$
6	$1.5571e-01$	$5.4159e+00$	$2.4599e+00$	$2.4498e+00$	$1.2721e+00$
7	$4.6808e-02$	$4.9173e-01$	$1.9444e-01$	$2.0303e-01$	$1.0403e-01$
8	$1.9899e+00$	$2.3879e+01$	$4.9748e+00$	$5.5847e+00$	$3.7885e+00$
9	$9.9498e-01$	$2.1889e+01$	$7.9597e+00$	$8.6937e+00$	$3.9614e+00$
10	$3.6648e+00$	$3.6734e+02$	$1.2868e+02$	$1.1943e+02$	$1.0325e+02$
11	$1.8597e+01$	$1.6346e+03$	$5.6494e+02$	$5.7595e+02$	$3.0133e+02$
12	$7.6195e-03$	$3.8644e-01$	$1.1127e-01$	$1.2416e-01$	$8.3969e-02$
13	$3.7911e-02$	$3.3836e-01$	$1.4853e-01$	$1.5769e-01$	$6.2114e-02$
14	$1.2744e-01$	$3.9109e-01$	$2.5765e-01$	$2.5370e-01$	$6.8082e-02$
15	$3.7187e-01$	$2.6298e+00$	$9.2013e-01$	$1.0218e+00$	$5.0308e-01$
16	$1.4868e+00$	$3.9634e+00$	$2.7252e+00$	$2.7469e+00$	$4.9249e-01$
17	$4.0376e+02$	$6.4374e+04$	$8.0552e+03$	$1.6075e+04$	$1.7718e+04$
18	$5.5815e+01$	$2.4290e+04$	$7.1702e+03$	$7.4198e+03$	$5.1241e+03$
19	$1.0352e+00$	$4.5166e+00$	$1.7157e+00$	$2.0933e+00$	$7.8083e-01$
20	$4.3321e+00$	$7.0763e+03$	$9.2784e+02$	$1.7192e+03$	$1.9259e+03$
21	$1.4310e+02$	$1.7164e+04$	$3.0918e+03$	$4.8234e+03$	$4.2292e+03$
22	$1.0507e+00$	$1.6341e+02$	$2.0799e+01$	$3.7567e+01$	$4.0177e+01$
23	$3.2946e+02$	$3.2946e+02$	$3.2946e+02$	$3.2946e+02$	$9.4145e-06$
24	$1.1012e+02$	$2.0487e+02$	$1.2736e+02$	$1.3076e+02$	$1.5627e+01$
25	$1.3128e+02$	$2.0287e+02$	$1.9488e+02$	$1.8368e+02$	$2.0744e+01$
26	$1.0002e+02$	$1.0029e+02$	$1.0013e+02$	$1.0014e+02$	$6.2764e-02$
27	$2.4338e+00$	$4.0633e+02$	$3.4671e+02$	$2.8078e+02$	$1.5766e+02$
28	$1.0188e+02$	$7.9089e+02$	$4.7514e+02$	$4.7715e+02$	$1.0963e+02$
29	$2.5161e+02$	$5.5437e+02$	$4.1309e+02$	$4.1329e+02$	$7.2925e+01$
30	$1.0715e+03$	$2.3445e+03$	$1.7814e+03$	$1.7275e+03$	$3.1530e+02$

TABLE XI: Results for 30D

Func.	Best	Worst	Median	Mean	Std. Dev.
1	$3.2954e+05$	$3.1864e+06$	$1.0665e+06$	$1.3108e+06$	$7.0642e+05$
2	$2.4655e+03$	$1.9976e+04$	$8.5582e+03$	$9.2995e+03$	$3.9564e+03$
3	$4.0904e+02$	$1.5083e+04$	$3.8262e+03$	$4.9164e+03$	$3.7777e+03$
4	$1.1317e-01$	$1.4103e+02$	$8.1351e+01$	$9.3626e+01$	$3.0286e+01$
5	$2.0000e+01$	$2.0001e+01$	$2.0000e+01$	$2.0000e+01$	$1.5237e-04$
6	$1.3670e+01$	$2.1999e+01$	$1.7894e+01$	$1.7893e+01$	$2.1841e+00$
7	$4.9700e-04$	$5.1221e-02$	$1.1675e-02$	$1.6496e-02$	$1.6139e-02$
8	$1.0993e+01$	$5.8703e+01$	$2.8109e+01$	$3.0178e+01$	$8.8008e+00$
9	$2.5869e+01$	$8.6561e+01$	$4.2783e+01$	$4.5690e+01$	$1.3464e+01$
10	$5.9321e+02$	$2.8281e+03$	$1.1198e+03$	$1.2770e+03$	$5.3541e+02$
11	$2.0210e+03$	$5.2852e+03$	$3.3891e+03$	$3.4225e+03$	$6.4794e+02$
12	$5.5958e-02$	$5.3133e-01$	$1.3844e-01$	$1.6184e-01$	$8.4330e-02$
13	$1.5443e-01$	$4.7254e-01$	$2.8412e-01$	$2.8167e-01$	$5.6491e-02$
14	$1.2965e-01$	$2.4866e-01$	$1.8792e-01$	$1.8665e-01$	$2.6632e-02$
15	$5.1145e+00$	$2.9274e+01$	$1.3563e+01$	$1.4068e+01$	$4.7245e+00$
16	$9.7768e+00$	$1.2944e+01$	$1.1475e+01$	$1.1542e+01$	$6.5696e-01$
17	$5.3894e+04$	$7.7806e+05$	$3.4353e+05$	$3.3559e+05$	$1.7548e+05$
18	$4.6349e+01$	$3.3731e+03$	$2.6897e+02$	$5.5046e+02$	$7.1612e+02$
19	$1.1396e+01$	$1.6781e+01$	$1.3840e+01$	$1.4027e+01$	$1.2754e+00$
20	$3.4528e+03$	$3.9387e+04$	$1.1763e+04$	$1.2018e+04$	$5.7052e+03$
21	$6.6342e+04$	$5.3146e+05$	$1.9755e+05$	$2.1197e+05$	$1.0976e+05$
22	$1.4846e+02$	$8.4777e+02$	$4.1171e+02$	$4.2071e+02$	$1.3888e+02$
23	$3.1524e+02$	$3.1526e+02$	$3.1525e+02$	$3.1525e+02$	$2.9608e-03$
24	$2.2368e+02$	$2.4464e+02$	$2.2773e+02$	$2.2895e+02$	$4.5394e+00$
25	$2.0649e+02$	$2.1392e+02$	$2.1051e+02$	$2.1054e+02$	$1.7009e+00$
26	$1.0021e+02$	$1.0056e+02$	$1.0036e+02$	$1.0036e+02$	$9.3247e-02$
27	$4.0250e+02$	$8.5105e+02$	$6.4950e+02$	$5.8929e+02$	$1.7176e+02$
28	$8.7902e+02$	$3.1856e+03$	$1.4349e+03$	$1.6022e+03$	$5.8851e+02$
29	$1.0321e+03$	$1.9850e+03$	$1.3055e+03$	$1.3306e+03$	$2.0581e+02$
30	$2.0985e+03$	$4.4863e+03$	$3.2094e+03$	$3.2273e+03$	$5.9982e+02$

TABLE XII: Results for 50D

Func.	Best	Worst	Median	Mean	Std. Dev.
1	$9.3233e+05$	$3.4638e+06$	$2.0414e+06$	$2.1285e+06$	$5.5455e+05$
2	$4.7555e+02$	$1.0668e+04$	$3.9445e+03$	$4.6175e+03$	$2.4460e+03$
3	$4.4492e+03$	$2.1852e+04$	$1.1492e+04$	$1.1327e+04$	$3.3122e+03$
4	$6.8296e+01$	$1.8960e+02$	$1.3975e+02$	$1.3257e+02$	$2.6275e+01$
5	$2.0000e+01$	$2.0001e+01$	$2.0000e+01$	$2.0000e+01$	$7.5942e-05$
6	$2.7653e+01$	$4.1354e+01$	$3.5999e+01$	$3.5642e+01$	$3.3526e+00$
7	$3.2813e-03$	$2.5539e-02$	$1.3214e-02$	$1.3104e-02$	$5.3208e-03$
8	$4.1790e+01$	$1.0646e+02$	$6.5712e+01$	$6.6967e+01$	$1.3401e+01$
9	$5.8703e+01$	$1.6616e+02$	$9.0541e+01$	$9.3156e+01$	$1.8650e+01$
10	$1.1302e+03$	$4.0901e+03$	$2.6482e+03$	$2.5661e+03$	$6.7580e+02$
11	$4.0153e+03$	$8.3330e+03$	$6.0564e+03$	$6.1785e+03$	$9.0019e+02$
12	$8.9723e-02$	$4.4947e-01$	$2.0066e-01$	$2.0718e-01$	$6.8092e-02$
13	$3.8756e-01$	$5.8012e-01$	$4.7233e-01$	$4.7179e-01$	$4.7699e-02$
14	$2.7542e-01$	$3.5641e-01$	$3.1486e-01$	$3.1634e-01$	$2.0054e-02$
15	$6.5653e+01$	$1.4188e+02$	$9.2688e+01$	$9.5118e+01$	$1.7216e+01$
16	$1.9265e+01$	$2.2101e+01$	$2.0501e+01$	$2.0634e+01$	$7.4250e-01$
17	$1.1092e+05$	$8.9156e+05$	$3.3306e+05$	$3.4707e+05$	$1.6532e+05$
18	$1.3207e+02$	$3.1442e+03$	$8.8586e+02$	$1.0276e+03$	$6.3573e+02$
19	$2.1726e+01$	$6.6902e+01$	$2.6981e+01$	$2.9933e+01$	$8.1163e+00$
20	$7.2614e+03$	$3.3869e+04$	$1.7180e+04$	$1.7177e+04$	$5.8271e+03$
21	$1.7818e+05$	$9.5972e+05$	$4.4486e+05$	$4.6760e+05$	$1.8693e+05$
22	$3.9891e+02$	$1.5024e+03$	$1.0907e+03$	$1.0507e+03$	$2.6318e+02$
23	$3.4400e+02$	$3.4401e+02$	$3.4401e+02$	$3.4401e+02$	$3.0307e-04$
24	$2.5613e+02$	$2.8525e+02$	$2.7420e+02$	$2.7318e+02$	$6.2152e+00$
25	$2.0000e+02$	$2.2900e+02$	$2.2114e+02$	$2.1915e+02$	$8.1734e+00$
26	$1.0021e+02$	$2.0020e+02$	$1.0033e+02$	$1.2185e+02$	$4.1464e+01$
27	$1.0490e+03$	$1.3747e+03$	$1.1786e+03$	$1.1929e+03$	$7.8248e+01$
28	$3.3779e+03$	$6.8852e+03$	$4.8930e+03$	$4.8432e+03$	$7.2192e+02$
29	$1.5854e+03$	$3.1441e+03$	$2.5127e+03$	$2.4586e+03$	$4.9458e+02$
30	$1.5217e+04$	$2.0769e+04$	$1.8309e+04$	$1.8439e+04$	$1.1925e+03$

TABLE XIII: Results for 100D

Func.	Best	Worst	Median	Mean	Std. Dev.
1	$2.4183e+07$	$4.8513e+07$	$3.1610e+07$	$3.2417e+07$	$4.5619e+06$
2	$8.4452e+02$	$3.9053e+04$	$1.3851e+04$	$1.4606e+04$	$6.6768e+03$
3	$1.7794e+04$	$3.5966e+04$	$2.7092e+04$	$2.7015e+04$	$4.0642e+03$
4	$3.3040e+02$	$4.7099e+02$	$3.9537e+02$	$3.9568e+02$	$3.5115e+01$
5	$2.0000e+01$	$2.0001e+01$	$2.0000e+01$	$2.0000e+01$	$8.0875e-05$
6	$8.7948e+01$	$1.1304e+02$	$9.7877e+01$	$9.7551e+01$	$4.9272e+00$
7	$9.2296e-03$	$4.1721e-02$	$2.1302e-02$	$2.2152e-02$	$7.3382e-03$
8	$1.5323e+02$	$2.5671e+02$	$1.9707e+02$	$2.0017e+02$	$2.3871e+01$
9	$2.0098e+02$	$2.9351e+02$	$2.4078e+02$	$2.4548e+02$	$2.2234e+01$
10	$3.5154e+03$	$1.0181e+04$	$6.0405e+03$	$6.3267e+03$	$1.2752e+03$
11	$1.0120e+04$	$1.7056e+04$	$1.4028e+04$	$1.3669e+04$	$1.5581e+03$
12	$1.9985e-01$	$5.5885e-01$	$3.7161e-01$	$3.7991e-01$	$8.9085e-02$
13	$4.3925e-01$	$5.8912e-01$	$5.0285e-01$	$5.0104e-01$	$3.0011e-02$
14	$1.5143e-01$	$1.8138e-01$	$1.6187e-01$	$1.6281e-01$	$7.2109e-03$
15	$3.6164e+02$	$5.6875e+02$	$4.4969e+02$	$4.5313e+02$	$5.2330e+01$
16	$4.1126e+01$	$4.5664e+01$	$4.3330e+01$	$4.3612e+01$	$1.0370e+00$
17	$1.0304e+06$	$3.7266e+06$	$2.0981e+06$	$2.1717e+06$	$4.8270e+05$
18	$2.1530e+02$	$1.5259e+03$	$5.7270e+02$	$6.3202e+02$	$3.1093e+02$
19	$5.0736e+01$	$1.6730e+02$	$9.2625e+01$	$9.9327e+01$	$1.9690e+01$
20	$4.7750e+04$	$1.1140e+05$	$6.8687e+04$	$7.1711e+04$	$1.4154e+04$
21	$1.0533e+06$	$3.3801e+06$	$1.8279e+06$	$1.9177e+06$	$4.7684e+05$
22	$1.2238e+03$	$3.3274e+03$	$2.3040e+03$	$2.2938e+03$	$4.5285e+02$
23	$3.6349e+02$	$3.7643e+02$	$3.7006e+02$	$3.7003e+02$	$3.2964e+00$
24	$3.6860e+02$	$3.8955e+02$	$3.7540e+02$	$3.7593e+02$	$4.3245e+00$
25	$2.0001e+02$	$2.6273e+02$	$2.3765e+02$	$2.2742e+02$	$2.0651e+01$
26	$2.0019e+02$	$2.0038e+02$	$2.0028e+02$	$2.0028e+02$	$4.1800e-02$
27	$2.0107e+03$	$2.7051e+03$	$2.3428e+03$	$2.3549e+03$	$1.3405e+02$
28	$1.0174e+04$	$1.3795e+04$	$1.2203e+04$	$1.2094e+04$	$8.9545e+02$
29	$2.9018e+03$	$4.7736e+03$	$3.7789e+03$	$3.8139e+03$	$4.6716e+02$