Non-Uniform Mapping in Binary-Coded Genetic Algorithms

Kalyanmoy Deb, Yashesh D. Dhebar, and N. V. R. Pavan

Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
PIN 208016, U.P., India
deb,yddhebar@iitk.ac.in,nvrpavan@yahoo.co.in

KanGAL Report No. 2012011

Abstract. Binary-coded genetic algorithms (BGAs) traditionally use a uniform mapping to decode strings to corresponding real-parameter variable values. In this paper, we suggest a non-uniform mapping scheme for creating solutions towards better regions in the search space, dictated by BGA's population statistics. Both variable-wise and vector-wise non-uniform mapping schemes are suggested. Results on five standard test problems reveal that the proposed non-uniform mapping BGA (or NBGA) is much faster in converging close to the true optimum than the usual uniformly mapped BGA. With the base-line results, an adaptive NBGA approach is then suggested to make the algorithm parameter-free. Results are promising and should encourage further attention to non-uniform mapping strategies with binary coded GAs.

1 Introduction

Genetic algorithms (GAs) were originally started with a binary-coded representation scheme [1, 2], in which variables are first coded in binary strings comprising of Boolean variables (0 and 1). Since such a string of Boolean variables resembled a natural chromosomal structure of multiple genes concatenating to a chromosome, the developers of GAs thought of applying genetic operators (recombination and mutation) on to such binary strings – hence the name genetic algorithms.

For handling problems having real-valued variables, every real-valued variable x_i is represented using a binary substring $\mathbf{s}^i = (s_1^i, s_2^i, \ldots, s_{\ell_i}^i)$ where $s_j^i \in \{0, 1\}$. All n variables are then represented by $\ell = \sum_{i=1}^{n} \ell_i$ bits. Early GA researchers suggested a uniform mapping scheme to compute x_i from a substring $\mathbf{s}^i$:

$$x_i = x_i^{(L)} + \frac{x_i^{(U)} - x_i^{(L)}}{2^{\ell_i} - 1} DV(\mathbf{s}^i), \tag{1}$$

where $\ell_i = |\mathbf{s}^i|$ is the number of Boolean variables (or bits) used to represent variable x_i, $x_i^{(L)}$ and $x_i^{(U)}$ are lower and upper bounds of variable x_i and $DV(\mathbf{s}^i)$ is the decoded value of string $\mathbf{s}^i$, given as follows: $DV(\mathbf{s}^i) = \sum_{j=1}^{\ell} 2^{j-1} s_j^i$. This mapping scheme is uniform in the range $[x_i^{(L)}, x_i^{(U)}]$, because there is a constant gap between two consecutive values of a variable and the gap is equal to $(x_i^{(U)} - x_i^{(L)})/(2^{\ell_i} - 1)$. The uniform mapping mentioned above covers the entire search space with uniformly distributed set of points. Without any knowledge about good and bad regions in the search space, this is a wise thing to do and early users of BGA have rightly used such a scheme. However, due to its uniformity, BGA requires a large number of iterations to converge to the requisite optimum. Moreover, in every generation, the GA population usually has the information about the current-best or best-so-far solution. This solution can be judiciously used to make a faster search.

In this paper, we suggest the possibility of a non-uniform mapping scheme in which the above equations are modified so that binary substrings map non-uniformly in the search space. If the biasing can be done to have more points near the good regions of the search space, such a non-uniform mapping may be beneficial in creating useful solutions quickly. The binary genetic algorithm (BGA) framework allows such a mapping and we investigate the effect of such a mapping in this paper.

In the remainder of the paper, we first describe the proposed non-uniform mapping scheme and then describe the overall algorithm (we call it as NBGA). Thereafter, we present simulation results of NBGA on a number of standard problems and compare its performance with original uniformly mapped BGA. The NBGA approach involves an additional parameter. Finally, we propose an adaptive scheme (ANBGA) that do not require the additional parameter and present simulation results with the adaptive NBGA approach as well. Conclusions of this extensive study are then made.

2 Past Efforts of Non-Uniform Mapping in BGA

Despite the suggestion of BGAs in early sixties, it is surprising that there does not exist too many studies related to non-uniform mapping schemes in coding binary substrings to real values. However, there are a few studies worth mentioning.

ARGOT [3] was an attempt to adaptively map fixed binary strings to the decoded variable space. The methodologies used in the study used several environmentally triggered operators to alter intermediated mappings. These intermediate mappings are based on internal measurements such as parameter convergence, parameter variance and parameter 'positioning' within a possible range of parameter values. The dynamic parameter encoding (DPE) approach [4] adjusts the accuracy of encoded parameters dynamically. In the beginning of a GA simulation, a string encodes only most significant bits of each parameter (say 4bits or so), and when GA begins to converge, most significant bits are recorded and dropped from the encoding. New bits are introduced for additional precision. In the delta coding approach [5], after every run, the population is reinitialized with the substring coding for each parameter representing distance or Δ value away from the corresponding parameter in best solution of the previous run, thereby forming a new hypercube around the best solution. The size of hypercube is controlled by adjusting number of bits used for encoding.

All the above were proposed in late eighties and early nineties, and have not been followed up adequately. Non-uniform mapping in BGA can produce faster convergence if some information about a good region in the search space is identified. Here, we suggest a simple procedure that uses the population statistics to create a non-uniform mapping in BGAs.

3 Proposed Non-Uniformly Mapped BGA (NBGA)

In NBGA, we create more solutions near the best-so-far solution ($\mathbf{x}^{b,t}$) at any generation t. We suggest a polynomial mapping function for this purpose with a user-defined parameter η. Let us consider Figure 1 in which the lower bound is a, upper bound is b of variable x_i and i-th variable value of best-so-far solution is $x_i^{b,t}$ at the current generation t. Let us also assume that the substring $\mathbf{s}^i$ decodes

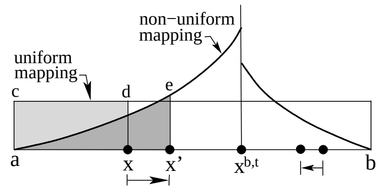


Fig. 1. Non-uniform mapping scheme is shown. Area acdxa is equal to area aex′a.

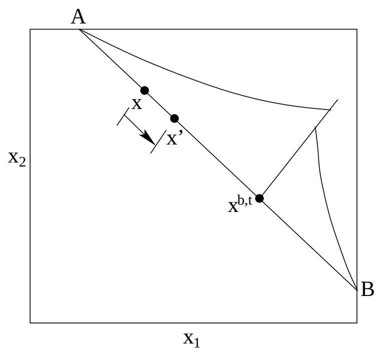


Fig. 2. Vector-wise mapping scheme is shown.

to the variable value x_i using Equation 1 with $a = x_i^{(L)}$ and $b = x_i^{(U)}$. The quantity in fraction varies between zero and one, as the decoded value $DV(\mathbf{s}^i)$ varies between zero and $(2^{\ell_i} - 1)$. Assuming the fraction $\zeta = DV(\mathbf{s}^i)/(2^{\ell_i} - 1)$, which lies in $[0, 1]$, we observe a linear behavior between x_i and ζ for uniform mapping used in BGA, $x_i = a + \zeta(b - a)$, or $\zeta = (x_i - a)/(b - a)$.

For NBGA, we re-map value x_i to obtain x'_i ($m : x_i \to x'_i$) so that there is more concentration of points near $x_i^{b,t}$, as follows (with $\eta \geq 0$) and as shown in Figure 1:

$$m(\zeta) = k\zeta^{\eta}, \qquad (2)$$

where k is a constant, which is chosen in such a manner that the point $x_i = x_i^{b,t}$ remains at its own place, that is, $\int_0^{(x_i^{b,t}-a)/(b-a)} m(\zeta)d\zeta = (x_i^{b,t} - a)/(b - a)$. This condition gives us: $k = (\eta+1)(b-a)^{\eta-1}/(x_i^{b,t}-a)^{\eta}$. Substituting k to Equation 2 and equating area under the mapped curve ($\int_0^{(x'_i-a)/(b-a)} m(\zeta)d\zeta$) with area under uniform mapping ($(x_i - a)/(b - a)$), we have the following mapping:

$$x'_i = a + \Big((x_i - a)(x_i^{b,t} - a)^{\eta}\Big)^{1/(1+\eta)}. \tag{3}$$

Figure 1 shows how the points in between $a \leq x_i \leq x_i^{b,t}$ are re-mapped to move towards $x_i^{b,t}$ for $\eta > 0$. For a point in the range $x_i^{b,t} \leq x_i \leq b$, a mapping similar to the above will cause the solution to move towards $x_i^{b,t}$ as well, as shown in the figure. It is clear that depending on the location of $x_i^{b,t}$ at a generation, points towards left and right of it will get re-mapped towards the best-so-far point – thereby providing a non-uniform mapping of the binary substrings in the range $[a, b]$.

The above non-uniform mapping scheme can be implemented in two ways: (i) variable-wise and (ii) vector-wise. We describe them in the following subsections.

3.1 Variable-wise Mapping

In variable-wise mapping, we perform the above re-mapping scheme for every variable one at a time, taking the corresponding variable value of the best-so-far solution ($x_i^{b,t}$) and computing corresponding re-mapped value x'_i. This remapping operation is performed before an individual is evaluated. For an individual string $\mathbf{s}$, first the substring $\mathbf{s}^i$ is identified for i-th variable, and then x'_i is calculated using Equation 3. The variable value of the best-so-far solution of the previous generation is used. Thus, the initial generation does not use the non-uniform mapping, but all population members from generation 1 onwards are mapped non-uniformly as described. It is important to note that the original string $\mathbf{s}$ is unaltered. The individual is then evaluated using variable vector $\mathbf{x}'$. The selection, recombination and mutation operators are performed on the string $\mathbf{s}$ as usual.

3.2 Vector-wise Mapping

In the vector-wise mapping, we coordinate the mapping of all variables in a systematic manner. The mapping operation is explained through Figure 2 for a two-variable case. For any individual $\mathbf{s}$, its location ($\mathbf{x}$) in the n-dimensional variable space can be identified using the original uniform mapping. Also, the location of the best-so-far ($\mathbf{x}^{b,t}$) solution in the n-dimensional space is located next. Thereafter, these two points are joined by a straight line and the line is extended to find the extreme points ($\mathbf{A}$ and $\mathbf{B}$) of the bounded search space. Parameterizing, $\mathbf{x}' = \mathbf{A} + d(\mathbf{x} - \mathbf{A})$, and substituting $a = 0$, $x_i = 1$ and $x_i^{b,t} = \|\mathbf{x}^{b,t} - \mathbf{A}\|/\|\mathbf{x} - \mathbf{A}\|$ in Equation 3, we compute the re-mapped point x'_i and save it as d'. Thereafter, the re-mapped variable vector can be computed as follows:

$$\mathbf{x}' = \mathbf{A} + d'(\mathbf{x} - \mathbf{A}). \tag{4}$$

The re-mapped vector will be created along the line shown in the figure and will be placed in between $\mathbf{x}$ and $\mathbf{x}^{b,t}$. The vector-wise re-mapping is done for every population member using the best-so-far solution and new solutions are created. They are then evaluated and used in the selection operator. Recombination and mutation operators are used as usual. Binary strings are unaltered. The vector-wise re-mapping is likely to work well on problems having a linkage among variables. We present the results obtained from both methods in the following section.

4 Results

We use the binary tournament selection, a multi-variable crossover, in which a single-point crossover is used for each substring representing a variable, and the usual bit-wise mutation operator.

We consider six standard unconstrained objective functions, which are given 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 are considered for $n = 20$ variables. Following parameter values are fixed for all runs:

- Population size = 100,
- String length of each variable = 15,
- Lower Bound = −8, Upper Bound = 10,
- Selection type: Tournament Selection,
- Crossover: Variable wise crossover with crossover probability = 0.9,
- Mutation: Bitwise mutation with mutation probability = 0.0033.

An algorithm terminates when any of the following two scenarios take place: (i) a maximum generation of $t_{\max} = 3,000$ is reached, or (ii) the difference between the best objective function value and the corresponding optimal function value of 0.01 or less is achieved. For each function, 50 runs are performed and the number of function evaluations for termination of each run is recorded. Thereafter, minimum, median and maximum number of function evaluations (FE) are reported. Note that a run is called success (S), if a solution with a difference in objective function value from the optimal function value of 0.01 is achieved, otherwise the run is called a failure (F). In the case of failure runs, the obtained objective function values (f) are reported.

4.1 Linear Increase in η:

Figure 1 suggests that a large value of η will bring solutions close to the current best solution. During initial generations, the use of a large η may not be a good strategy. Thus, first, we use an η-update scheme in which η is increased from zero linearly with generations to a user-specified maximum value $\eta_{\max}$ at the maximum generation $t_{\max}$:

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

We use different values of $\eta_{\max}$ = 10, 100, 1000, 5000, 10000, and 15000, for variable-wise mapping case and compare results with the usual uniformly-mapped binary coded GA. The results are tabulated for sphere function in Table 1. The

Table 1. Performance of linearly increasing η schemes are shown for the sphere function.

Method	FE or f	S or F	min	median	max
BGA	FE	S = 17	9,701	13,201	23,501
	f	F = 33	0.0517	0.0557	0.157
NBGA ($\eta_{\max} = 10$)	FE	S = 50	10,501	12,901	176,701
NBGA ($\eta_{\max} = 100$)	FE	S = 50	7,401	10,101	87,901
NBGA ($\eta_{\max} = 1,000$)	FE	S = 50	3,901	5,901	19,501
NBGA ($\eta_{\max} = 5,000$)	FE	S = 50	2,301	2,901	3,501
NBGA ($\eta_{\max} = 10,000$)	FE	S = 50	**1,801**	**2,301**	**2,801**
NBGA ($\eta_{\max} = 15,000$)	FE	S= 50	1,901	2,301	3,101

first aspect to note is that the original BGA (with uniform mapping) is not able to solve the sphere problem in 33 out of 50 runs in 3,000 generations, whereas all NBGA runs with $\eta_{\max} \geq 10$ are able to find the desired solution in all 50 runs. This is a remarkable performance depicted by the proposed NBGA.

The best NBGA performance is observed with $\eta_{\max} = 10,000$. Note that although such a high $\eta_{\max}$ value is set, the median run with 2,301 function evaluations required 23 generations only. At this generation, η value set was $\eta = (23/3000) \times 10000$ or 76.67. The important matter in setting up $\eta_{\max} = 10,000$ is that it sets up an appropriate rate of increase of η with generations (about 3.33 per generation) for the overall algorithm to work the best. An increase of η from 1.67 per generation to 3.33 per generation seems to work well for the Sphere function, however, an increase of η by 5 per generation (equivalent to $\eta_{\max}$ =

15, 000) is found to be not so good for this problem. Since the sphere function is a unimodal problem, a large effective η maps the solution closer to the best-so-far solution. But a too close a solution to the best-so-far solution (achieved with a large η) may cause a premature convergence, hence a deterioration in performance is observed.

Figure 3 shows the variation in function value of the population-best solution with generation for the sphere function of the best run. It is clear that NBGA is much faster compared to BGA.

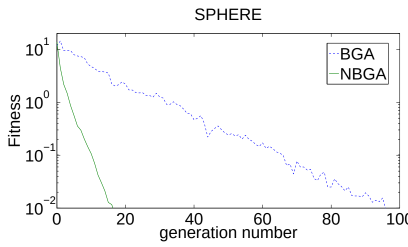


Fig. 3. Population-best objective function value with generation for the sphere function.

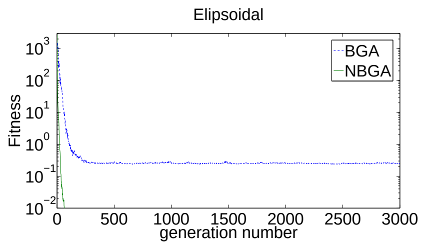


Fig. 4. Population-best objective function value with generation for the ellipsoidal function.

Next, we consider the ellipsoidal function and results are shown in Table 2. For this problem, $\eta_{\max} = 15,000$ turns out to be the best option. This corre-

Table 2. Performance of linearly increasing η schemes are shown for the ellipsoidal function.

Method	FE or f	S or F	min	median	max
BGA	f	F = 50	0.2374	4.5797	29.6476
NBGA ($\eta_{\max} = 10$)	FE	S = 11	40,001	236,501	284,201
	f	F = 39	0.01	0.0564	3.3013
NBGA ($\eta_{\max} = 100$)	FE	S = 46	24,501	78,001	198,101
	f	F = 04	0.0137	0.01986	0.0246
NBGA ($\eta_{\max} = 1,000$)	FE	S = 50	12,001	31,401	125,201
NBGA ($\eta_{\max} = 5,000$)	FE	S = 50	7,301	21,601	48,601
NBGA ($\eta_{\max} = 10,000$)	FE	S = 50	6,401	8,501	29,801
NBGA ($\eta_{\max} = 15,000$)	FE	S = 50	**5,701**	**7,701**	**21,501**

sponds to an increase of η of 5 per generation. At the final generation of the best run (having minimum function evaluations), an effective $\eta = (57/3000) \times 15000$ or 285 was used. This problem is also unimodal; hence a larger η produced a faster convergence. Notice here that the original BGA (with uniform mapping of variables) is not able to find the desired solution (close to the true optimum) in all 50 runs in a maximum of 3,000 generations, whereas all runs with $\eta_{\max} \geq 1,000$ are able to solve the problem in all 50 runs with a small number of function evaluations. Figure 4 shows the variation in function value of the population-best solution with generation for the ellipsoidal function of the best run. Again, NBGA is much faster compared to BGA.

The performance of BGA and NBGA approaches on Ackley's function are shown in Table 3. The best performance is observed with $\eta_{\max} = 10,000$ with

Table 3. Performance of linearly increasing η schemes are shown for the Ackley's function.

Method	FE or f	S or F	min	median	max
BGA	f	F = 50	0.249	1.02	1.805
NBGA ($\eta_{\max} = 10$)	f	F = 50	0.0208	0.1113	0.703
NBGA ($\eta_{\max} = 100$)	FE	S = 33	74,901	122,801	232,101
	f	F = 17	0.01073	0.0203	0.08
NBGA ($\eta_{\max} = 1,000$)	FE	S = 50	20,601	32,401	203,701
NBGA ($\eta_{\max} = 5,000$)	FE	S = 50	11,001	40,101	110,701
NBGA ($\eta_{\max} = 10,000$)	FE	**S = 50**	**10,301**	**31,601**	**66,701**
NBGA ($\eta_{\max} = 15,000$)	FE	S = 36	9,201	25,701	51,301
	f	F = 14	1.1551	1.1551	2.0133

100% successful runs. Here, $\eta \approx 1,054$ is reached at the final generation of a median NBGA's run. Importantly, a rate of increase of 3.33 per generation seems to be the best choice for Ackley's function. The BGA is found to fail in all 50 runs. Figure 5 shows the objective function value of the population-best solution with generation for Ackley's function. The results from the median run is shown. The proposed NBGA approach is found to be much faster compared to BGA.

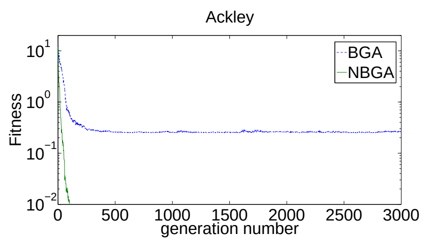


Fig. 5. Population-best objective function value with generation for Ackley's function.

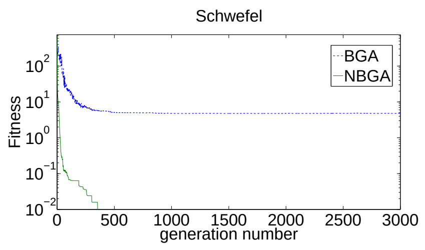


Fig. 6. Population-best objective function value with generation for Schwefel's function.

Next, we consider Schwefel's function. Table 4 tabulates the results. Here,

Table 4. Performance of linearly increasing η schemes are shown for the Schwefel's function.

Method	FE or f	S or F	min	median	max
BGA	f	F = 50	4.7183	17.417	37.6447
NBGA($\eta_{\max} = 10$)	f	F = 50	2.129	6.2925	16.093
NBGA ($\eta_{\max} = 100$)	f	F = 50	1.1814	2.3364	3.8168
NBGA ($\eta_{\max} = 1,000$)	FE	S = 19	73,501	251,001	298,501
	f	F = 31	0.0107	0.0436	0.1954
NBGA ($\eta_{\max} = 5,000$)	FE	S = 50	27,601	144,401	292,801
NBGA ($\eta_{\max} = 10,000$)	FE	S = 50	35,501	104,901	167,301
NBGA ($\eta_{\max} = 15,000$)	FE	S = 50	**25,701**	**74,401**	**124,301**

$\eta_{\max} = 15,000$ performs the best and the original BGA fails to make any of its 50 runs successful. Figure 6 shows the objective function value of the population-best solution with generation for Schwefel's function. The proposed NBGA approach is found to be much faster compared to BGA.

Next, we consider Rosenbrock's function. Table 5 shows the results. This

Table 5. Performance of linearly increasing η schemes are shown for the Rosenbrock's function.

Method	FE or f	S or F	min	median	max
BGA	f	F = 50	0.3686	42.78	281.7013
NBGA ($\eta_{\max} = 10$)	f	F = 50	0.1328	0.4836	90.1493
NBGA ($\eta_{\max} = 100$)	f	F = 50	0.0865	0.1918	17.0241
NBGA ($\eta_{\max} = 1,000$)	f	F = 50	0.0505	13.2236	76.1873
NBGA ($\eta_{\max} = 5,000$)	f	F = 50	0.0112	10.4476	19.3426
NBGA ($\eta_{\max} = 10,000$)	FE	**S = 27**	200,301	247,001	288,101
	f	F = 23	0.012	0.3443	19.0727
NBGA($\eta_{\max} = 15,000$)	FE	S = 24	109,001	183,701	213,801
	f	F = 26	0.0112	3.9888	69.2418

function is considered to be a difficult function to solve for the optimum. BGA could not find the desired optimum in all 50 runs. Also, NBGAs with small $\eta_{\max}$ values could not solve the problem. But, NBGA with $\eta_{\max} = 10,000$ is able to find the desired solution in 27 out of 50 runs.

It is clear from the above section that (i) The proposed non-uniform mapping method (NBGA) performs much better than the usual BGA, and (ii) In general, a high value of $\eta_{\max}$ produces a better performance.

5 Adaptive NBGA

An increasing η with a rate of about 3 to 5 per generation seems to work well, but the optimal performance depends on the problem. In this section, we propose an adaptive approach in which, instead of a predefined increase in η with generation, the algorithm will decide whether to increase of decrease η after every few generations.

In the adaptive NBGA, we initialize η to be zero at the initial generation. Thereafter, we keep track of the current-best [1] objective function value ($f(\mathbf{x}^{cb,t})$) at every five generations, and re-calculate $\eta_{\max}$, as follows:

$$\eta_{\max} = \eta_{\max} - 100\frac{f(\mathbf{x}^{cb,t}) - f(\mathbf{x}^{cb,t-5})}{f(\mathbf{x}^{cb,t-5})}. \tag{11}$$

The above equation suggests that if the current-best is better than the previous current-best solution (five generations ago), $\eta_{\max}$ is increased by an amount proportional to percentage decrease in objective function value. Since an increase in $\eta_{\max}$ causes decoded values to move closer to the best-so-far solution, an improvement in current-best solution is rewarded by a further movement towards the best-so-far solution. On the other hand, if the new current-best is worse than the previous current-best solution, $\eta_{\max}$ is reduced. In effect, an effort is being made to increase diversity of the population by not moving the solutions much towards the best-so-far solution.

When the change in the current-best function value is less than 0.001 in three consecutive updates of $\eta_{\max}$, it is likely that the population diversity is depleted and in such a case we set $\eta_{\max} = 0$ and also set mutation probability to 0.5 to introduce diversity in the population. From the next generation on, the mutation probability is set back to 0.0033 as usual, but $\eta_{\max}$ is set free to get updated using Equation 11. After updating $\eta_{\max}$ after every five generations, η is varied from $\eta_{\max}/5$ to $\eta_{\max}$ for the next five generations in uniform steps.

5.1 Results with Adaptive NBGA Approach

All five problems are chosen and identical parameter settings as those used in Section 4 are used here. Table 6 compares the performance of the adaptive NBGA approach with both variable-wise and vector-wise non-uniform mappings. In both cases, identical adaptation scheme (Equation 11) is used. The table also compares the adaptive NBGA results with the original variable-wise NBGA approach discussed earlier.

Figure 7 shows variation of $\eta_{\max}$ and population-best objective value with generation on three problems. It can be observed that both variable-wise and vector-wise non-uniform mapping approaches perform well. However, the fact that adaptive approach does not require a pre-defined $\eta_{\max}$ value and is still able to perform almost similar to the pre-defined increasing η-scheme, we recommend the use of the adaptive NBGA approach.

[1] The current-best solution is different from the best-so-far solution, as the former is the population-best solution at the current generation.

Table 6. Comparison of performance between adaptive NBGA and original NBGA approaches.

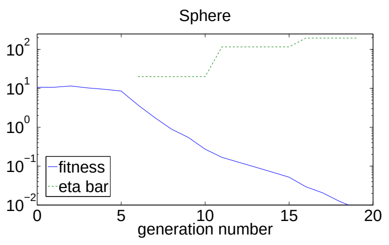
Function	Method	S or F	FE or f	min	median	max
Sphere	Variable-wise	S = 50	FE	2,001	3,401	4,501
	Vector-wise	S = 50	FE	2,001	2,601	3,501
	NBGA ($\eta_{\max} = 10,000$)	S = 50	FE	**1,801**	**2,301**	**2,801**
Ellipsoidal	Variable-wise	S = 50	FE	5,401	8,501	**14,601**
	Vector-wise	S = 50	FE	**5,101**	**7,501**	17,101
	NBGA ($\eta_{\max} = 15,000$)	S = 50	FE	5,701	7,701	21,501
Ackley	Variable-wise	S = 50	FE	**8,901**	**29,901**	69,001
	Vector-wise	S = 49	FE	9,101	25,101	241,001
		F = 01	f	1.841	1.841	1.841
	NBGA ($\eta_{\max} = 10,000$)	S = 50	FE	10,301	31,601	**66,701**
Schwefel	Variable-wise	S = 50	FE	24,501	**58,001**	**103,601**
	Vector-wise	S = 50	FE	**18,801**	75,601	113,901
	NBGA ($\eta_{\max} = 15,000$)	S = 50	FE	25,701	74,401	124,301
Rosenbrock	Variable-wise	F = 50	f	0.019	0.092	12.413
	Vector-wise	F = 50	f	0.023	0.083	12.888
	NBGA ($\eta_{\max} = 10,000$)	**S = 27**	FE	200,301	247,001	288,101
		F = 23	f	0.012	0.3443	19.0727

6 Conclusions

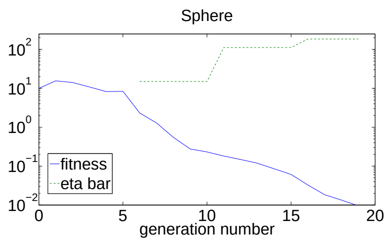
In this paper, we have suggested two non-uniform mapping schemes in binary-coded genetic algorithms (BGA). Using the best-so-far solution information at every generation, the variable-wise and vector-wise non-uniformly coded BGAs (or NBGAs) map binary substrings to a variable value that is somewhat closer to the best-so-far solution than its original decoded value. The non-uniform mapping requires a parameter $\eta_{\max}$, which takes a non-negative value. The variable-wise approach with an increasing $\eta_{\max}$-update is applied to five standard test problems and the performance of the proposed NBGA has been found to be much better than the BGA approach that uses a uniform mapping scheme.

Thereafter, the NBGA approach has been made adaptive by iteratively updating the $\eta_{\max}$ parameter associated with the NBGA approach. Starting with $\eta_{\max} = 0$ (uniform mapping) at the initial generation, the adaptive NBGA approach has been found to update the $\eta_{\max}$ parameter adaptively so as to produce similar results as in the case of increasing $\eta_{\max}$-update strategy. Both variable-wise and vector-wise non-uniform approaches have been found to perform equally well.

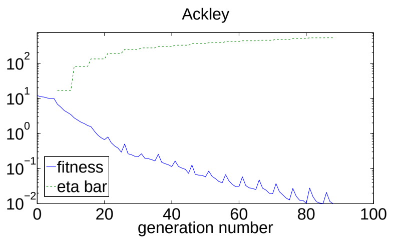
The research in evolutionary algorithms originally started with binary-coded GAs, but recently their use has been limited due to the advent of real-parameter GAs and other EC methodologies. Instead of BGA's usual choice of uniform mapping, the effective use of a non-uniformly mapped representation scheme demonstrated in this paper should cause a resurrection of BGAs in the near future.



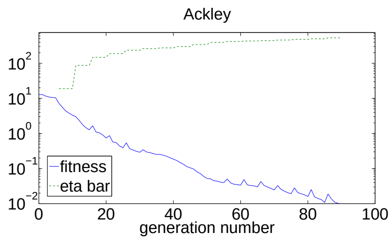
(a) Variable-wise on sphere function.



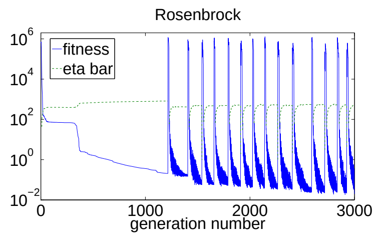
(b) Vector-wise on sphere function.



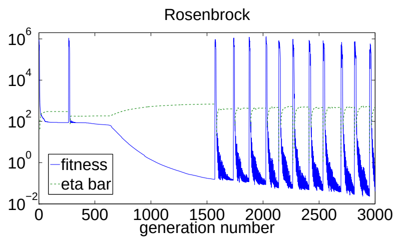
(c) Variable-wise on Ackley's function.



(d) Vector-wise on Ackley's function.



(e) Variable-wise on Rosenbrock's function.



(f) Vector-wise on Rosenbrock's function.

Fig. 7. Variations in $\eta_{\max}$ with generation number for variable-wise and vector-wise non-uniform mapping are shown.

References

1. J. H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: MIT Press, 1975.
2. D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
3. 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.
4. N. N. Schraudolph and R. K. Belew. Dynamic parameter encoding for genetic algorithms. Technical Report LAUR90-2795, Los Alamos: Los Alamos National Laboratory, 1990.
5. 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.