Binary or Real? Real-coded Binary!

Jiri Kubalik
Czech Technical University in Prague
Technicka 2, 166 27 Prague, Czech Republic
e-mail: kubalik@labe.felk.cvut.cz

Abstracts: *This paper presents a novel approach to improve the performance of genetic algorithms - called genetic algorithms with real-coded binary representation (GARB). The proposed algorithm is capable of maintaining the population diversity during the whole run which protects it from premature convergence. This is achieved by using a special encoding scheme with a high redundancy, which is supported by so-called gene-strength adaptation mechanism. The proposed approach was evaluated on various test problems, all of them considered to be hard for standard genetic algorithms. The results show the algorithm significantly outperforms standard genetic algorithm and achieves results competitive with other techniques on non-stationary problems.*

Keywords: genetic algorithms, representation, redundancy, premature convergence.

1 Introduction

Genetic algorithms (GAs) are probabilistic population-based search techniques. A weak point of conventional GAs is that they often suffer from so-called *premature convergence*, which is caused by an early homogenization of genetic material in the population.

There are many factors that affect the convergence of a GA. Up to now research have been devoted to estimation of an adequate population size that would provide the GA with sufficient amount of genetic material to evolve the optimal chromosome [4]. Various selection schemes have been analyzed and convergence models of GAs under those selection schemes were developed [3]. Research in the area of choosing the proper recombination operators and the probabilities of their application that would provide the best balance between the exploitation and exploration of the GA has been done in [9] and [10]. An interesting approach for preserving population diversity has been introduced in [5]. The population is explicitly prevented from becoming homogenous by specifying the maximum difference between frequency of ones and zeros at any position of the chromosome in the population.

The mechanisms for maintaining population diversity become necessary when GAs are used for optimization in non-stationary environments. Thus GAs are extended to be capable of continuously adapting to changes in the environment when searching for optimal solutions. The simplest way to react to changes in the environment is to restart the genetic algorithm [1] or increase the rate of mutation to reach diversity in population [7] whenever the change of the system is detected. The drawback of such approaches is their poor ability to reuse the information gained in the past. Other approaches to maintaining the diversity in population are based on using redundant representation. A frequently used approach to redundant representations is the diploidy and dominance with multiple alleles for each gene position [2]. For non-stationary environments the diploidy can be enhanced by a dominance switching mechanism [8]. However, results achieved with multiploid representations published so far indicate that the approach is

suitable for non-stationary environments with only a few periodically changing states.

The next section introduces the proposed real-coded binary representation. Section 3 describes test problems selected for experimental evaluation of the approach. Experiments and comments on achieved results are presented in section 4. Last section concludes the paper.

2 Real-coded Binary Representation

The motivation for the proposed real-coded representation is to introduce a redundancy into the genetic code that would make the genetic algorithm resistant against premature convergence. This is achieved so that the chromosomes are not formed of direct zeros and ones as it is in the conventional genetic algorithms with binary representation. Instead, each gene is expressed by a real number r from the interval $\langle 0.0, 1.0\rangle$, which is interpreted as binary gene 0 or 1 as follows:

$$\begin{aligned} interpretation(r) &= 1, \text{ when } r > 0.5 \\ &= 0, \text{ when } r < 0.5. \end{aligned}$$

An incidental gene value 0.5 is replaced by random value resulting in unbiased interpretations of zero and one. The real-valued genes expressions represent a *gene strength*. The value $r = 0.0$ represents the strongest binary zero, and $r = 1.0$ represents the strongest one. Then as the value r changes from 1.0 to 0.5 the strength of binary one decreases and vice versa. Similarly this applies to binary zero as the value r changes on the interval 0.0 to 0.5. Such a pseudo-binary representation provides many possibilities for expressing any arbitrary binary chromosome. For example two chromosomes $Ch_1 = [0.9,\ 0.07,\ 0.23,\ 0.62]$ and $Ch_2 = [0.65,\ 0.19,\ 0.4,\ 0.86]$ are interpreted as the same binary chromosome [1, 0, 0, 1].

Standard crossover operators are used with this representation. First, the offspring is composed of parts taken from the parental chromosomes and then the gene strength of each gene is changed by the so-called *gene-strength adjustment mechanism*. It playes a crutial role in maintaining of the population diversity. The extent to which genes will be adjusted depends (i) on their interpretation and (ii) on the relative frequency of ones at their position in the chromosome. For this purpose a vector $P[]$ with relative frequency of ones at each position of the representation calculated over the whole population of interpreted chromosomes is maintained.

The gene at the i-th position is weakened if it interprets as the binary value that is more frequently sampled at the i-th position in the current population. And vice versa, the gene at the i-th position is strengthened if it represents the binary value that is less frequently sampled at the i-th position. The genes are weakened and strengthened proportionally to $P[i]$ as shown below. Thus if the real-valued gene represents binary value that prevails in the population at the given position then the gene is weakened – the value is moved towards 0.5. In opposite case the gene is strengthened – zero-value is moved towards 0.0 and one-value is moved towards 1.0, respectively. Parameter c denotes the *maximal adjustment step* defining the maximal value each gene can be changed by in one adjustment step. Parameter c can take on values from the

gene weakening:	$gene' = gene + c * (1.0 - P[i])$, if $(gene < 0.5)$ and $(P[i] < 0.5)$
	OR
	$gene' = gene - c * P[i]$, if $(gene > 0.5)$ and $(P[i] > 0.5)$
gene strengthening:	$gene' = gene - c * (P[i])$, if $(gene < 0.5)$ and $(P[i] > 0.5)$
	OR
	$gene' = gene + c * (1.0 - P[i])$, if $(gene > 0.5)$ and $(P[i] < 0.5)$

interval $(0.0, 0.5)$. However useful values are rather small ranging between 0.0 and 0.2.

The idea behind the introduced manipulation with the strength of genes is as follows. In standard evolutionary algorithms the above-average building blocks are used more often for creating the new population that ends up in undesirable homogenization of the population genotype. Here, the genes that are likely to prevail the population are weakened each time they are used in the newly created individual. Such genes are gradually approaching the value 0.5 and at some point of this weakening process they are adjusted so that they turn over the critical value 0.5 and change their interpretation from one to zero and vice versa, respectively. Even when some binary gene completely overwhelm the population at some position it is just a matter of time when the opposite value will emerge and reproduce again. In opposite case, binary genes that are sampled in the current population with below-average frequency are strengthened in the newly created individuals in order to make them more likely to survive and reproduce.

Important thing is that the weakening and strengthening of genes is driven by the demand of the population for needed genes at given positions. Moreover there is no need for any explicit mutation operator, since *all genes are* passed into the new population already *modified by the gene-strength adjustment mechanism*, which can be considered as an *implicit mutation*.

The gene-strength adjustment offers a possibility to distinguish between "promising" and "ordinary" individuals. Let us consider the newly generated individual that is better than its both parents as a promising one. As such its genotype should be retained in the population with an unchanged interpretation for some time. In order not to change its interpretation by the adjustment too early all its genes are rescaled to be strong before it is inserted into the new population (Note that the generational evolutionary model is used in this work). This means that the genes interpreted as ones are set by random to be close to 1.0 and genes interpreted as zeros are set to be close to 0.0. For example the individual $o = (0.71, 0.45, 0.18, 0.57)$ would be rescaled to $o' = (0.97, 0.03, 0.02, 0.99)$. Obviously the latter one is more likely to survive unchanged for some generations than the original one. This rescaling mechanism boosts up the exploitation ability of the algorithm.

3 Test Problems and Experimental Setup

The test problems used in our experiments were selected to be representatives of non-linear function, deceptive, hierarchical and non-stationary optimization problems - all of them considered to be hard for genetic algorithms.

First test problem is based on function F101(x, y) taken from [12]. It is non-linear non-separable and highly multimodal function of two variables. Our problem consists of 10 copies of the function. The total length of the problem is 200 bits and the global minimum value is -955.96.

The second problem is a representative of deceptive problems. As the base function the 4-bit fully deceptive function DF3 taken from [13] was used. The problem is composed of 50 DF3 functions resulting in a 200-bit long chromosome with the global optimum of value 1500 in the string of all ones and local optima about the value 1400.

A hierarchical-if-and-only-if function (H-IFF) was proposed in [11]. The function has two global optima - one consists of all 1's and the other all 0's. The 256-bit function with global optima of value 2304 was used here.

As a representative of non-stationary problems an oscillating version of the single knapsack problem was used [6]. The goal is to fill a knapsack using a subset of objects from an available set of size n, such that the sum of object weights is as close as possible to the target weight t. The problem uses 14 objects, each of them of the weight $w_i = 2^i$, where i ranges from 0 to 13.

The optimum fitness is 1.0 and the fitness decreases towards 0.0 as the Hamming distance to the optimum solution string increases. The target weight oscillates between values 12643 and 2837, where the change occurs every 1500 generations as used in [6].

50 independent runs were carried out for each experiment and the results were averaged. Performance of GARB has been evaluated and compared to the standard genetic algorithm (SGA) and algorithms used in [6]. The comparisons are based on the quality of the solutions achieved after the same number of fitness function evaluations, which is a commonly used condition when comparing different evolutionary algorithms. Parameters of GARB and SGA:

- population size: 500 (F101, DF3, H-IFF), 150 (Knapsack)
- tournament selection with $N = 3$ for all problems
- evolutionary model: generational
- elitism: one copy of the best individual is copied to the new population
- crossover: 2-point (F101, DF3, H-IFF), uniform (Knapsack)
- mutation: simple bit-swapping mutation operator is used in SGA
- #fitness eval.: 500000 (F101, DF3, H-IFF), 30000 (Knapsack)

4 Experiments and Results

In Table 1 we see that GARB with well chosen maximal adjustment step c significantly outperforms SGA on all static test problems. We observe an optimal behavior of GARB for rather moderate values of c; 0.075 in case of F101 and DF3 problem and 0.125 in case of H-IFF problem. Results achieved with $c = 0.005$ as well as with $c = 0.175$ are poor. This is in agreement with our expectations that the exploration power of the algorithm is low when too small c is used. On the other hand, too big values of c cause the exploitation of the useful genetic information drops down since the changes of the gene interpretation are too fast.

Table 2 compares GARB with results obtained for Haploid-Recover, Extended-Additive and Ng-Wong approaches presented in [6]. There the steady-state evolutionary model with uniform crossover was running for 1500 generations between two subsequent target weight changes. This equals to 1500 or 3000 fitness evaluations depending on whether the crossover operator generated

Table 1: Comparison of GARB and SGA on F101, DF3, and H-IFF problems. The results are in the form *average best-of-run / stdev.*

Algorithm	F101	DF3	H-IFF
GARB $c = 0.005$	-903 / 20	1479 / 7	1544 / 155
GARB $c = 0.025$	-935 / 7	1500 / 0	2112 / 138
GARB $c = 0.075$	**-941 / 4**	**1500 / 0**	2253 / 100
GARB $c = 0.125$	-931 / 8	1494 / 3	**2304 / 0**
GARB $c = 0.175$	-893 / 10	1390 / 15	1808 / 45
SGA $Mutation_{rate} = 0.5$	-896 / 14	1451 / 6	1267 / 80
SGA $Mutation_{rate} = \mathbf{1}$	**-905 / 17**	**1461 / 9**	**1323 / 76**
SGA $Mutation_{rate} = 2$	-900 / 10	1408 / 9	1289 / 35
SGA $Mutation_{rate} = 3$	-883 / 13	1362 / 17	1267 / 82

Table 2: Comparison of GARB and other algorithms on Oscillating knapsack problem. A number of runs in which the optimum was found in each period is shown.

		Oscillation period									
	Algorithm	1	2	3	4	5	6	7	8	9	10
	GARB $C = 0.025$	46	0	7	0	17	6	0	0	11	9
Oscillation	GARB $C = 0.075$	45	19	5	11	9	16	7	10	12	10
period	**GARB** $C = 0.125$	**41**	**35**	**31**	**37**	**38**	**32**	**36**	**35**	**41**	**32**
10 generations	GARB $C = 0.175$	32	28	25	28	29	33	35	21	28	31
	GARB $C = 0.025$	47	1	0	1	0	6	3	2	1	4
Oscillation	GARB $C = 0.075$	50	46	34	42	28	43	29	40	25	43
period	**GARB** $C = 0.125$	**49**	**49**	**49**	**47**	**49**	**49**	**50**	**50**	**50**	**48**
20 generations	GARB $C = 0.175$	46	42	44	44	48	39	46	44	43	42
	Haploid-Recover	45	44	33	45	33	44	29	43	37	47
	Extended-Additive	43	29	44	42	39	40	45	37	39	40
	Ng-Wong	32	21	41	25	34	27	32	26	32	27

1 or 2 offspring (this is not specified in [6]). Our implementation of GARB uses generational model so we have carried out experiments with two different oscillation periods – 10 and 20 generations – in order to change the target after 1500 and 3000 fitness evaluations, respectively. Results achieved with GARB for oscillation period 10 generations and $c = 0.125$ are better than Ng-Wong but slightly worse than Haploid-Recover and Extended-Additive. When the oscillation period 20 generations was used GARB outperformed all other algorithms. Again the best results were achieved with c set to 0.125.

5 Conclusions and Future Work

This paper introduces a novel approach to prevent a premature convergence of a genetic algorithm. It is based on a redundant pseudo-binary representation where the binary genes are represented by real numbers where all values greater than 0.5 are interpreted as binary ones and values less than 0.5 are interpreted as binary zeros, respectively. The introduced *gene-strength adjustment mechanism* is used to control the diversity of the evolved population so that real-valued genes corresponding to the binary value that prevails at the given position in the population are weakened and pushed towards the opposite binary interpretation. The real-valued genes interpreted as the minor binary value are strengthened making it more likely to retain in the population and reproduce.

The proposed algorithm was empirically evaluated on various test problems proving that it represents a competitive alternative to the techniques designed to prevent a premature convergence. The algorithm exhibits a strong capability of self-controlling of the population diversity that extends its explorative power and makes it capable of recovering even from completely homogeneous population.

Analysis and deep understanding of the algorithm's behavior will be the primary goal of the future work. Especially applications to optimization in non-stationary environment will be investigated as the experiments revealed the capabilities of tracking a dynamically changing optimum. Original version of the proposed algorithm uses a generational model where the convergence characteristics of the population is updated after the whole population has been generated. Steady-state model could provide faster response to the state of the population since the statistics could have been updated after inserting each single individual. Implementation

and analysis of the steady-state model are subject to the future research as well.

Acknowledgments

This research work was supported by the Grant Agency of the Czech Republic within the project No. 102/02/0132.

References

[1] Fukunaga, A. (1997). Restart scheduling for genetic algorithms. In Thomas Back, editor, *Proceedings of ICGA'97*, 1997.

[2] Goldberg D. E., Smith, R. E. (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J.J. Grefenstette, ed., *Proceedings of ICGA'87*, pp. 59-68. Lawrence Erlbaum Associates, 1987.

[3] Goldberg, D. E., Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G. J. E. (Ed.), *Foundations of Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, pp. 69-93, 1991.

[4] Harik, G. R., Cant-Paz, E., Goldberg, D. E., Miller, B. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In Bck, T. (Ed.), *Proceedings of ICEC'97*, pp. 7-12, New York: IEEE Press, 1997.

[5] Kubalik, J., Rothkrantz, L.J.M., Lazansky, J. (2002). Genetic Algorithm with Limited Convergence. In *Proceedings of the 4th International Workshop on Frontiers in Evolutionary Algorithms* in conjunction with the 6th JCIS, ISBN 0-9707890-1-7, pp. 610-613, 2002.

[6] Lewis, J., Hart, E., Ritchie,G. (1998). A comparison of dominance mechanisms and simple mutation on non-stationary problems. In Eiben, A.E. et al. (Eds.). *Proceedings of PPSN'98*, LNCS 1498, pages 139-148. Springer, 1998.

[7] Morrison, R. W., De Jong, K. A. (2000). Triggered hypermutation revisited. In *Proceedings of CEC'2000*, pp. 1025-1032, 2000.

[8] Ng, K. P., Wong, K. C. (1995). A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *Proceedings of ICGA'95*, pp. 159-166. Morgan Kaufmann, 1995.

[9] Spears, W. M. (1997). Recombination Parameters. In *Handbook of Evolutionary Computation*, T. Back, D. B. Fogel, and Z. Michalewicz, eds., New York: Oxford Univ. Press and Institute of Physics, pp E1:3:1-11, 1997.

[10] Srinivas, M., Patnaik, L. M. (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms. In *IEEE Transaction of System, Man and Cybernetics*, vol. 24, pp. 656-667, 1994.

[11] Watson, R. A., Hornby, G. S. and Pollack, J. B. (1998). Modeling Building-Block Interdependency. In *Proceedings of PPSN'98*, Springer, pp.97-106., 1998.

[12] Whitley D., Mathias, K., Rana, S., Dzubera, J. (1996). Evaluating Evolutionary Algorithms, *Artificial Intelligence*, Volume 85, pp. 245-261, 1996.

[13] Whitley D. (1991). Fundamental Principles of Deception in Genetic Search. In: *Foundations of Genetic Algorithms*, G. Rawlins ed., Morgan Kaufmann, pp. 221-241, 1991.