An adaptive resolution hybrid binary-real coded genetic algorithm

Omar Arif Abdul-Rahman*, Masaharu Munetomo++, and Kiyoshi Akama++

*Graduate School of Information Science and Technology, Hokkaido University,

Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido, 060-0814, Japan.

++Information Initiative Center, Hokkaido University,

Kita 11, Nishi 5, Kita-ku, Sapporo, Hokkaido, 060-0811, Japan

++Tel: + 81-11-706-7890, ++Fax: + 81-11-706-7890, *E-mail:omar@ist.hokudai.ac.jp

++Information Initiative Center, Hokkaido University, Sapporo, 060-0811, Japan

Abstract-Which is better be used in Genetic Algorithms (GAs) binary encoding schemes or floating point encoding schemes? In this paper, we try to tackle this controversial question by proposing a novel algorithm that divides the computational power between two cooperative versions of GAs. They are a binary-coded GA (bGA) and a real-coded GA (rGA). The evolutionary search is primarily led by bGA, which identifies promising regions in the search space; while, the rGA increases the quality of the obtained solutions by conduct an exhaustive search through these regions. The resolution factor (R), which has a value that is increasingly adapted during the search, controls the interactions between two versions. We conducted Comparison experiments employing a typical benchmark function to proof the feasibility of the algorithm under critical scenarios of increasing problem dimensions and decreasing precision power.

Key Words: Binary-coded GA, Real-coded GA, Hybrid Scheme

1. INTRODUCTION

The intersection between Genetic Algorithms (GAs) as an emerging paradigm within artificial intelligence and robotics as a prominent application of it has a long history. From one side, GAs play a vital role in many artificial life models. They are currently the most prominent and widely used models of evolution in artificial life systems. In artificial intelligence, GAs have been used both as a tool for solving practical problems and as a scientific model of evolutionary processes. On the other hand, GAs have been successfully applied to solve complex problems related to robotics like path planning, navigation, controller optimization and line balancing problem.

In this paper, again we try to approach this long and controversial debate over the binary encoding schemes and real (floating point) encoding schemes for GAs. For theoretician, bGA is an attractive solution. The findings of schemata theory support that enhanced schemata processing can be obtained by using alphabets of low cardinality. bGAs are efficient, and the latest developments in the field of GAs research add much to the robustness, speed and accuracy of such algorithms. However, it is also possible to argue that bGAs still suffer from several disadvantages when applied to real-world problems involving a large number of design variables. An example of such disadvantage is the direct relationship between the desired precision and the increased binary string length. Another example is the discrepancy between the binary representation space and the actual problem space.

On other hands, rGAs are preferred by many practitioners. They are on rising usage since the floating point representation is conceptually closest to the real design space, and moreover. The

string length reduces to the number of design variables. rGAs are robust, accurate, and efficient. However, it is also possible to argue that rGAs are still vulnerable to premature convergence, especially for complex real-world problems with a large number of design variables. Furthermore, theory of rGAs, even so, is far from providing plausible understanding of internal mechanisms of rGAs. That is a true hinder before the development of advanced techniques in this field.

Therefore, in this paper we propose an adaptive resolution binary-real coded GA (brGA) with the purpose of combining the advantages of both versions and minimizing the disadvantages of them. Under the proposed algorithm, two versions of the same population of solutions are kept, binary-coded and real-coded populations. They are updated by continuously mapping from one version to another during the evolutionary search. The search is primarily guided by bGA part, which identifies promising regions in the search space; while, the rGA part increases the quality of the obtained solutions by conduct a detailed search through these regions. The interactions between two parts are controlled by a resolution factor that it is increasingly adapted during the evolutionary search. It has a small value at the begin of the search to allow the exploration of the search space by the bGA part, while, its value increases gradually as the search progresses to allow the exploitation of the search space by rGA part.

The remaining part of this paper is organized as follows. Related literature is reviewed in Section 2. The proposed algorithm framework and implementation are explained in Section 3. Then, experimental results are reported and discussed in Section 4. Finally, we conclude the paper in Section 5.

2. LITERATURE REVIEW

From a conceptual standpoint, we adopted in this work an idea that it is closely related to the dynamic coding, which is a sophisticated approach to alter the coarseness of search spaces. An example of such approach is the stochastic genetic algorithm as presented by Krishnakumar et al.[1]. In stochastic GAs, the region represented by each point of a bGA is adapted during the optimization process using Evolutionary Strategies (ES). In contrast to stochastic GAs, we employed an rGA instead of ES to adapt the regions represented by the bGA.

Another example of dynamic coding is the Adaptive Range Genetic Algorithm (ARGAs) as presented by Arakawa and Hagiwara.[2] In ARGAs, mapping rules from binary to real strings is updated during the optimization process according to the population statistics to adapt the population toward promising design regions. However, in this work we employed a rGA that samples from the regions determined by a bGA to adapt the population toward promising regions in the search space.

From the hybrid scheme standpoint, we employed two cooperative versions of binary-real GAs in the development of this algorithm. Similar approach is adopted by Barrios et al.[3] in their cooperative binary-real coded GA for designing and training feed-forward artificial neural networks. They employed two interconnected GAs that work in parallel to design and train better neural networks that solve a problem at hand. In contrast to this paralleled interacting cooperative algorithm, the cooperative versions in this algorithm are interacting in an interleaving manner.

3. THE ALGORITHM

The proposed algorithm (brGA) flowchart is shown in Fig.1, which illustrates the typical optimization cycle. The main components of the algorithm can be described as follows.

1) **Encoding schemes:** Floating point and binary encoding schemes are used to represent two populations of solutions. These populations are kept updating by continuously mapping from one version to another during the evolutionary cycle.

2) **Binary-coded GA (bGA):** The evolutionary cycle is usually led by the bGA, which has the responsibility of exploring the search space and identifying promising regions within it. In this algorithm, bGA part is implemented using a standard bGA as described by Haupt and Haupt.4 The main parts of this algorithm are the single point crossover operator, one point mutation operator and rank weighting selection scheme with elitism.

3) **Population handover:** During a typical optimization cycle, populations of solutions are frequently mapped from one version to another and vice versa by a process that can be best described as population handover. The process of population handover is decided and controlled by a resolution factor (R) that has an adaptively increasing value which allows the rGA to increase the coarseness of the search space. During the process of population handover from the bGA to rGA, the best half from the current binary-coded population is directly transmitted to the real-coded population, while the other half is created by random sampling from the region determined by the binary-coded population. On other

hands, the process of population handover from rGA to bGA is done by totally replacing the latest binary-coded population with the current real-coded population.

4) **Real-coded GA (rGA):** The role of this part is to adapt the population toward the promising regions of the search space by exploiting the regions which identified by bGA. In this paper, we have implemented this part using Unimodal Normal Distribution (UNDX) as an advanced real crossover operator [5]. Minimal Generation Gap (MGG) is used as a generation-alteration model.

4. EXPERIMENTAL RESULTS

The proposed algorithm was implemented using MATLAB 7. Griewangk's function was used to testify the performance of the algorithm. The function is mathematically defined as follows.

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad (1)$$

Griewangk's function as a common benchmark function has been chosen due to its complex structure of numerous local extremum. It has a global optimum at zero.

In order to testify the feasibility and effectiveness of the brGA, two sets of experiments were performed. In the first set of experiments, the performance of the algorithm is testified against increasing problem dimensions. Here, the performance of brGA is compared against the original bGA [4] and UNDX rGA [5], when they applied separately to the benchmark function. Experimental setup conditions are shown in Table 1, while algorithms primary parameters'

settings are shown in Table 2. The value of the resolution factor from Table 2 represents the initial value. When updated, the resolution factor is incremented by a value equal to step size. The values of both resolution factor and step size have been selected after an extensive tuning process. On the other hand, primary parameters' settings of bGA [4] and UNDX rGA [5] are selected as recommended by the authors.

Experimental results are shown in Table 3. The best solution and the mean of best solutions from 50 runs are used to measure quality, while, efficiency rates shown in the Table 3 are used to measure the quality-computational effort relationship. It is defined according to Hillstrom[6] as follows.

$$Efficiency = ln\left(\left|f^{(0)} - f^{*}\right| / \left|\acute{f} - f^{*}\right|\right)/T \qquad (2)$$

Where $f^{(0)}$, f^{*} and $\acute{f}$ are the best initial, best known and best final fitness, while, T is the execution time in seconds. Efficiency rates shown in Table 3 were averaged over 50 test runs.

By comparing brGA to bGA, it is clear from Table 3 that brGA outperforms in terms of quality performance in all the cases. The difference in performance is clearer at higher problem dimensions. In terms of efficiency rates, bGA outperforms only in case of 10 dimensions in spite of its lower solution quality, since the execution time of bGA is smaller than that of brGA. However, its efficiency rates degrade drastically to zero in other cases, because bGA loses the ability to progress towards the promising regions at higher problem dimensions, which is one of the binary coding disadvantages. In contrast, brGA successfully maintains reasonable efficiency

rates. So, the experimental results clearly indicate that the adopted hybrid scheme enhanced the time and quality performance of its bGA part.

On the other hand, by comparing brGA to rGA it is obvious that rGA slightly outperforms at lower dimensions (10,50). However, But the difference in performance increases at the higher dimensions (100, 120) for the benefit of brGA. The same is true for efficiency rates. So, that indicate that the adopted hybrid scheme is effectively enhanced the quality and time performance of its rGA part at difficult situations of high-dimensions problems.

In the second set of experiments, the sensitivity of the brGA performance is testified against increasing chromosome length. The same settings as shown in Table 1 and Table 2 were used. However, here the problem dimensions are fixed to 2 and the performance of brGA is compared against that of bGA.

The experimental results are illustrated in Fig. 2. As expected, performance of bGA is largely depending on the chromosome length. By increasing the number of bits, the coding precision is increased. This translated into an enhanced bGA performance. In contrast to bGA, it is evident from Fig. 2 that brGA shows a little sensitivity toward chromosome length. So, the adopted hybrid scheme is useful in minimizing the dependency of its bGA part on chromosome length to achieve a better quality performance. This will be helpful in decreasing the complexity of the algorithms in case of difficult real-world problems with a lot of variables that require long binary chromosome representations to ensure solutions with a high precision.

5. CONCLUSION AND FUTURE WORK

An adaptive resolution binary-real coded GA (brGA) is proposed in this paper. It is a hybrid scheme that organizes the cooperation between rGA and bGA through an adaptive resolution factor to combine the advantages of both versions. brGA flowchart is presented and explained. Then, experimental results over Griewangk's benchmark problem are reported and discussed. It is clear from the obtained results that brGA significantly enhances the quality and time performance of its bGA part against increasing problem dimensions and reduces the sensitivity of its bGA part against increasing/decreasing chromosome length. On the other hand, the results also showed that the brGA enhances the quality and time performance of its rGA part in complex cases of high-dimensions problems.

Possible direction for future work includes extending the algorithm to the constrained optimization problems by incorporating appropriate constraints handling techniques and applies it to complex real-world problems like the resource allocation problem in cloud platforms, which is a part of this research agenda.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to the Editor-In-Chief and the anonymous reviewers for their efforts and comments, which help considerably in improving the quality of this paper. Moreover, this work is supported by Grant-in-Aid for Scientific Research (C) by MEXT, Japan.

REFERENCES

[1] Krishnakumar K, Swaminathan R, Garg S, Narayanaswamy S (1995) Solving large parameter optimization problems using genetic algorithms. Proceedings of the Guidance, Navigation, and Control Conference, 1995, pp 449-460.

[2] Arakawa M, Hagiwara I (1998) Development of Adaptive Real Range (ARRange) genetic algorithms. JSME Intl. J., Series C, Vol. 41, No. 4, 1998, pp. 969-977.

[3] Barrios D, Carrascal A, Manrique D, Ríos J (2003) Cooperative binary real coded genetic algorithms for generating and adapting artificial neural networks. Journal of Neural Computing and Applications, Vol. 12, No. 3-4, Dec. 2003, pp. 49-60.

[4] Haupt RL, Haupt SE (1998) Practical genetic algorithms. Wiley-Interscience, New York.

[5] Ono I, Kobayashi S (1997) A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. Proceedings of 7th ICGA, 1997, pp. 246-253.

[6] Hillstrom KE (1977) A simulation test approach to the evaluation of nonlinear optimization algorithms. ACM Transactions on Mathematical Software, Vol. 3, Issue 4, Dec. 1977, pp. 305–315.

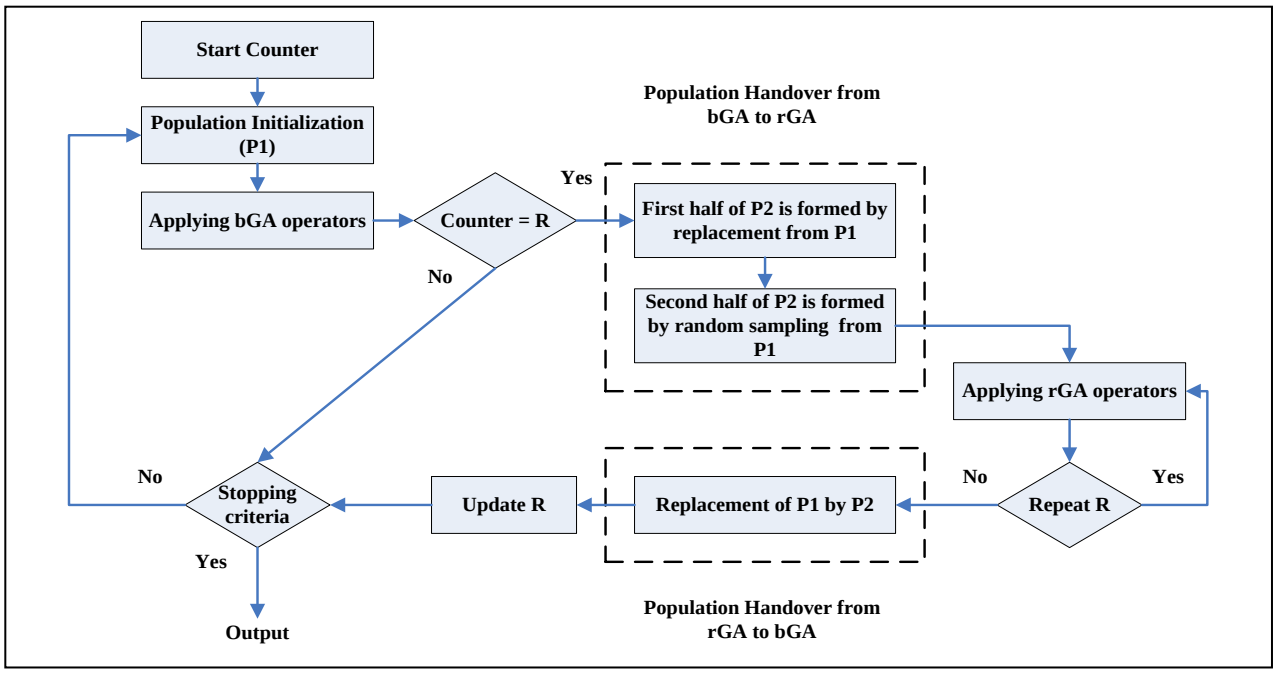

Fig.1. The proposed algorithm flowchart

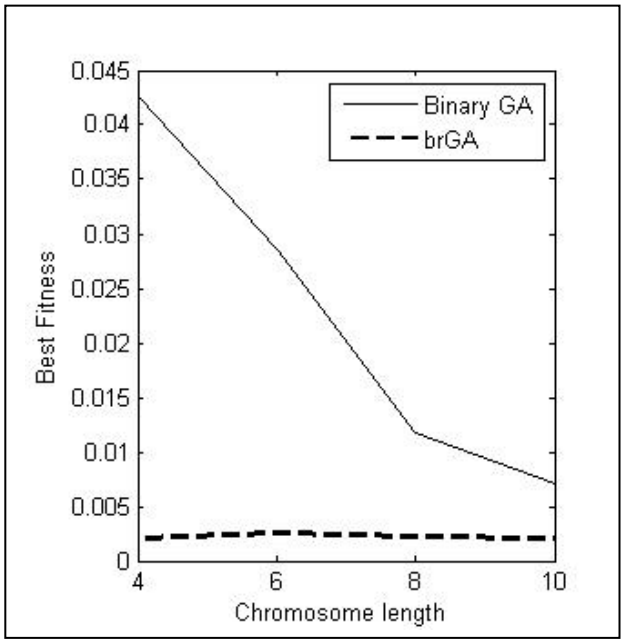

Fig.2. Solution quality versus chromosome length

Table 1. Experimental setup

Parameter name	Parameter value
Number of experiments	50
Number of iterations	990
Population size	6
Variable range	$5 \leq x \leq -5$
Initial range	$5 \leq x \leq -5$
Binary chromosome length (bits per parameter)	8

Table 2. Algorithms primary parameters

Parameter name	Parameter value		
	bGA	rGA	brGA
Resolution factor	-	-	3
Step size	-	-	2
Mutation probability	0.15	-	0.15
Selection pressure	0.5	-	0.5
Number of crossover	-	10	10
δ_z	-	0.5	0.5

Table 3. Performance against increasing problem dimensions

Problem Dimension	Binary GA			Real GA			brGA		
	Best	mean	Efficiency	Best	mean	Efficiency	Best	mean	Efficiency
10	0.1391	0.317	4.6125	0.004932	0.011739	3.8532	0.004934	0.012536	3.5739
50	1.0676	1.089	0	0.004305	0.024375	2.6723	0.004655	0.014299	2.2888
100	1.1602	1.188	0	0.61367	0.85668	0.17888	0.28473	0.45065	0.37779
120	1.1885	1.226	0	0.89541	0.98175	0.11178	0.57957	0.7713	0.16254