The Sixteenth International Symposium on Artificial Life and Robotics 2011 (AROB 16th '11),
B-Con Plaza, Beppu,Oita, Japan, January 27-29, 2011

An Adaptive Resolution Hybrid Binary-Real Coded Genetic Algorithm

Omar Abdul-Rahman[1], Masaharu Munetomo[2] and Kiyoshi Akama[2]

[1]Graduate School of Information Science & Technology, Hokkaido University,
[2]Information Initiative Center, Hokkaido University, Japan
(Tel : 81-11-706-7890; Fax : 81-11-706-7890)
(omar@ist.hokudai.ac.jp, munetomo@iic.hokudai.ac.jp, akama@iic.hokudai.ac.jp)

***Abstract*: Which is better to be used in Genetic Algorithms (GAs) binary or floating point coding schemes? In this paper, we try to answer this controversial question by proposing a novel algorithm that shares the computational power between two cooperated versions of GAs, a binary coded GA (bGA) and a real coded GA (rGA). The evolutionary search is primarily led by bGA, which is used to identify promising regions in the search space. While, the rGA is used to increase the quality of the obtained solutions by conduct a detailed search through these regions. The interactions between two versions are organized by a resolution factor that it is increasingly adapted during the evolutionary search. Comparison experiments were conducted over a typica l function to proof the feasibility of the algorithm under critical scenarios of increasing problem dimensi ons and decreasing precision power.**

Keywords: Binary coded GA, Real coded GA and Hybriding.

I. INTRODUCTION

The intersection between Genetic Algorithms (GAs) and artificial intelligence has a long history. From one side, GAs play a central 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 tools for solving practical problems and as a scientific models of evolutionary processes. On the other hand, GAs have been successfully applied to solve difficult real world problems related to robotics like path planning, navigation controller optimization and line balancing problem.

In this paper, we try to approach again this long and controversial debate over the real coded and binary coded GAs. For theoretician, binary coded GA is an attractive solution. The theoretical finding of schemata theory supports that enhanced schemata processing is obtained by using alphabet of low cardinality. Binary coded GAs are efficient and the latest developments in the field of GAs research adds much to the robustness, speed and accuracy of such algorithms. However, it is possible to argue that binary coded GAs suffer from several disadvantages when applied to real-world problems involving a large number of real design variables. The direct relationship between the desired precision and the increased binary string length, and the discrepancy between the binary representation space and the actual problem space are good examples of such disadvantages.

On other hand, real coded GAs 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. Real coded GAs are robust, accurate, and efficient. However, it is possible to argue that real coded GAs are still susceptible to premature convergence especially for complex real world problems with a large number of design variables. It is also possible to say that theory of real coded GAs is still far from providing plausible understanding of internal real coded GAs mechanisms, which is a true hinder before the development of advanced techniques in this field.

Therefore, in this paper we propose the adaptive resolution binary-real coded GA (brGA) that combines the advantages of both versions. 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 binary coded GA part, which is used to identify promising regions in the search space. While, the real coded GA part is used to increase the quality of the obtained solutions by conduct a detailed search through these regions. The interactions between two parts are organized 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 binary coded GA part, while, its value increases gradually as the search progresses to allow the exploitation of the search space by real coded GA part.

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

II. LITERATURE REVIEW

Dynamic coding 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 binary coded GA is adapted during the optimization process using Evolutionary Strategies (ES). In our work, we adopted a similar concept. However, in contrast to stochastic GAs we employed a real coded GA instead of ES to adapt the region represented by the binary coded GA.

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 our work we depend on a real coded GA that sample from the regions determined by a binary coded GA to adapt the population toward promising region in the search space.

Finally, a different 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 parallelly to design and train better neural networks that solves the problem. In our work, we also employed two cooperative GAs, but they interact with each others in an interleaving manner.

III. THE ALGORITHM

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

1. **Coding schemes:** floating point and binary coding schemes is used to represent two equivalent populations of solutions. These populations are kept updated by continuously mapping from one version to another during the evolutionary cycle.
2. **Binary coded GA:** the evolutionary cycle is usually led by the binary coded GA. This part is implemented using a binary GA as described by Haupt and Haupt [4] . The main parts of this algorithm are single point crossover operator, one point crossover operator, rank weighting selection scheme.
3. **Population handover**: the population of binary chromosomes that processed by the binary GA is transmitted to the real coded GA and vice versa at specific points of the evolutionary cycle. The process of population handover is controlled by the resolution factor that has an adaptive value that allows the real coded GA to increase the coarseness of the search space. During the process of handover, the best half from the current binary population is directly transmitted to the real coded population, while the other half comes by random sampling within the region determined by the binary population.
4. **Real coded GA**; the role of this version is to adapt the population toward the promising regions of the search space. 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.

IV. EXPERIMENTAL RESULTS

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

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

This function has complex structure of numerous local extremum. It has a value of 0 at global optimum.

In the first set of experiments, the performance of the algorithm is testified against increasing problem dimensions. The performance is compared against an ordinary binary GA[4] and UNDX real coded GA[5].

The value of resolution factor was chosen as 2 increases in step of 2. The settings for real and binary coded GA were chosen as recommended in [4] and [5].

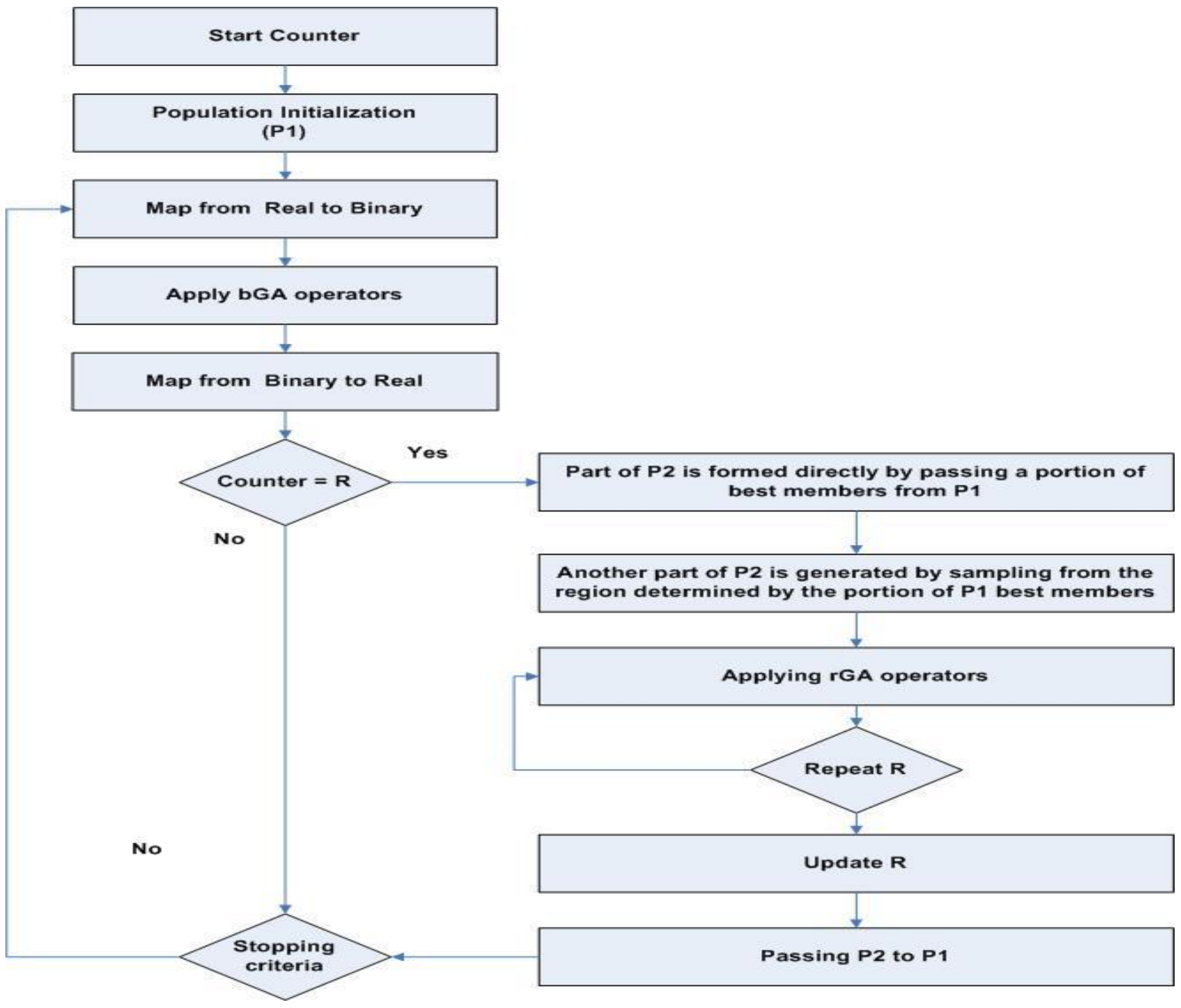
Start Counter
Population Initialization (P1)
Map from Real to Binary
Apply bGA operators
Map from Binary to Real
Counter = R
Yes
No
Part of P2 is formed directly by passing a portion of best members from P1
Another part of P2 is generated by sampling from the region determined by the portion of P1 best members
Applying rGA operators
Repeat R
Update R
No
Stopping criteria
Passing P2 to P1

Fig.1. The proposed algorithm flowchart

The results were averaged over 50 experiments. Each run consisted of 990 iterations and a population of 6 members was used. The function is minimized over the range of $5 \leq x \leq -5$. Each parameter is represented by 8 bits.

Experimental results are shown in Table 1. Efficiency rates shown in the table is used to measure quality- computational effort relationship.
It is defined according to Hillstrom [6] as follows.

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

Where$f^{(0)}$,f^{*} and $\acute{f}$ are the best initial, best known a nd best final fitness, while, T is the elapsed time i n seconds.

By comparing brGA to bGA, it is evident from the table that brGA outperforms in term of performance in all the cases. The difference in performance is clear at higher problem dimensions. In term of efficiency rate, bGA outperforms only in case of 10 dimensions in spite of lower solution quality. This is due to the fact the implementation of bGA is faster than brGA. However, its efficiency rates degrades drastically to 0 in other cases, while, brGA keeps reasonable efficiency rates. This is due to the fact that the bGA loses the ability to progress the population at higher problem dimensions, which is one of binary coding disadvantages.

On the other hand, by comparing brGA to rGA it is evident that the performance is comparable at lower dimensions (10,50). But, the difference in performance increases at the higher dimensions (100, 120) for the benefit of brGA. So, the hybriding enhance the performance of rGA at complex situation of high problem dimension. In term of efficiency rates, the results are comparable.

In the second set of experiments, the performance of the algorithm is testified against increasing chromosome length. The same settings as above were used. However, here the dimension is fixed to 2. In addition, only bGA and brGA were testified.

The experimental results are shown in Fig. 2. As expected the performance of bGA is largely depend on chromosome length. By increasing the number of bits, coding precision increases. Which, translated into

Table 1. Performance against increasing problem dimension

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

an enhanced performance. It is clear from Fig. 2 that brGA shows an independent performance against chromosome length. So, the hybriding is useful in decreasing the dependency of its binary part on chromosome length to achieve a better performance. That it is helpful in case of high precision that requires long chromosome representation.

V. CONCLUSION

An adaptive resolution binary-real coded GA is proposed in this paper. It is a framework that integrates the cooperation between rGA and bGA through an adaptive resolution factor to combine the advantages of both versions. Algorithm flowchart is presented, and explained. Then, experimental results over a typical benchmark problem is reported and discussed. It is evident from the results that the proposed algorithm greatly enhanced the performance of the binary GA part against increasing problem dimensions and decreasing chromosome length. On the other hand, the algorithm enhances the performance of rGA in case of higher problem dimension.

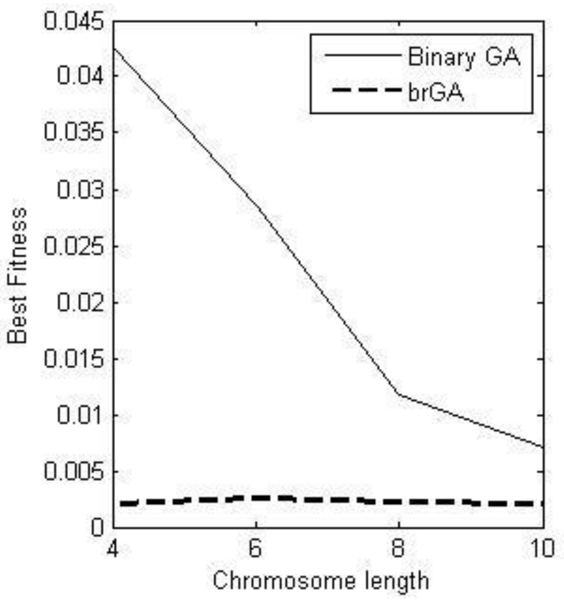

Fig.2. Solution quality versus chromosome length

ACKNOWLEDGMENT

This work is supported by Grant-in-Aid for Scientific Research (C) by MEXT, Japan.

REFERENCES

[1] Krishnakumar, K., Swaminathan, R., Garg, S. and N arayanaswamy, S., Solving Large Parameter Optimizati on Problems Using Genetic Algorithms, Proc. of the Gu idance, Navigation, and Control Conference, (1995), 44 9-460.
[2] Arakawa, M. and Hagiwara, I., Development of Adaptive Real Range (ARRange) Genetic Algorithms, JSME Intl. J., Series C, Vol. 41, No. 4 (1998), 969-977.
[3] Barrios, D., Carrascal, A., Manrique, D., Ríos, J., Cooperative binary real coded genetic algorithms for generating and adapting artificial neural networks, Neural Computing and Applications 12 (2003), 49-60.
[4] Haupt,R.L. and Haupt,S.E. Practical Genetic Algorithms. (1998) Wiley, New York.
[5] Ono, I. and Kobayashi, S. ,A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover, Proc. 7th ICGA, 246-253 (1997).
[6] Hillstrom, K. E.,A simulation test approach to the evaluation of nonlinearoptimization algorithms. ACM Transactions on Mathematical Software, 3(4), 305– 315(1977)..