A Novel Restart Strategy for Solving Complex Multi-modal Optimization Problems Using Real-Coded Genetic Algorithm

Amit Kumar Das and Dilip Kumar Pratihar(✉)

Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India
amit.besus@gmail.com, dkpra@mech.iitkgp.ernet.in

Abstract. Genetic algorithm (GA) is one of the most popular and robust stochastic optimization tools used in various fields of research and industrial applications. It had been applied for solving many global optimization problems for the last few decades. However, it has a poor theoretical assurance to reach the globally optimal solutions, while solving the complex multi-modal problems. Restart strategy plays an important role in overcoming this limitation of a GA to a certain extent. Although there are a few restart methods available in the literature, these are not adequate. In this paper, a novel restart strategy is proposed for solving complex multi-modal optimization problems using a real-coded genetic algorithm (RCGA). To show the superiority of the proposed scheme, ten complex multi-modal test functions have been selected from the CEC 2005 benchmark functions and its results are compared with that of the other strategies.

Keywords: Restart strategy · Real-coded genetic algorithm
CEC 2005 benchmark test functions · Multi-modal optimization problems

1 Introduction

Optimization is a method of searching the best feasible solution out of several possibilities. The optimization problems can be classified broadly into two categories. The first case is that, where the user is satisfied with the obtained locally optimal solution. However, in the second case, the search algorithm will try to locate the globally optimal point out of the several locally optimal solutions. Now, the global optimization technique uses mainly two types of approaches, namely deterministic method and stochastic method [1]. Deterministic approaches have the assurance to obtain globally optimal solutions to the given accuracy for certain types of problems. However, they may not be suitable for tackling the non-smoothed or ill-conditioned problems. Stochastic approaches can be applied to all kinds of problems, but they have probabilistic promise to reach the globally optimal solution [2]. It is a well-accepted fact that whenever the size of variable search space is expanded exponentially and other exact approaches fail to produce the optimal solution, the stochastic method can be used effectively to search for the optimal or near-optimal solution.

A. Abraham et al. (Eds.): ISDA 2017, AISC 736, pp. 32–41, 2018.
https://doi.org/10.1007/978-3-319-76348-4_4

Among several stochastic approaches, some popular algorithms like genetic algorithm [3], particle swarm optimization [4], differential evolution [5], ant colony optimization [6], cuckoo search [7], bat algorithm [8] etc. are in use. Genetic algorithm (GA), which was introduced by J. Holland [3], is one of the most popular global optimization techniques. It has been used successfully for solving different optimization problems in various fields of engineering, computational sciences, bio-medicine, etc. [9–11]. However, as like others, GA has also a probabilistic assurance to find the globally optimal solution. This limitation of a GA is more visible in case of solving complex multi-modal optimization problems. Although there does not exist any such algorithm, which can solve all types of global optimization problems with certainty in finite time [12], it is always desirable for a GA to have a high probabilistic guarantee in search of the globally best result. Now, to enhance the global search capability of a GA, one way might be to increase its population size. However, a GA's search process would be computationally expensive due to a very large size of population. Another method could be to multi-start the GA. This is nothing but to restart the algorithm under some user-defined conditions and consider the best result, achieved among all the generations, as the globally optimal solution. There are a few restart strategies for GA available in the literature. However, there is a chance of further improvement of these strategies. In this study, a novel restart strategy is proposed for a real-coded genetic algorithm (RCGA) to solve complex multi-modal optimization problems. The performance of the RCGA with the proposed restart strategy is evaluated through ten multi-modal problems, selected from the CEC 2005 benchmark test functions [13], and its results are compared with that of the other strategies.

2 Literature Survey

For global optimization, a few restart strategies for GA are available in the literature. Ghannadian et al. [14] proposed a concept of the random restart to GA. In their work, the mutation operator is substituted by a random restart scheme. Whenever the search of a GA is seen to be stagnated, random restart introduces a set of new initial population and the next evolution starts with these new solutions. Beligiannis et al. [15] introduced a technique for restarting of the classical GA, where an insertion operator is used when the restarting criteria are satisfied. This insertion operator is nothing but the act of incorporating a certain constant percentage of the genomes from the last generation to the new initial population of the GA during restarting of the algorithm. For example, if the size of the population is 50 and the constant percent is taken as 20%, then 40 new solutions are created randomly and ten randomly chosen solutions from the last generation are merged together and the next evolution begins with this mixed population. The algorithm is restarted after every predefined number of generations. Hughes et al. [16] introduced a recentering-restarting evolutionary algorithm for the epidemic network to evolve and their method could be applied for TSP implementation, ordered gene problems, etc. Dao et al. [17] suggested an improved structure of a GA, where the algorithm restarts, if the best solution obtained so far is not improved till a certain number of consecutive generations. During restart, the new population consists of a number of elite

chromosomes (which is defined by the user), and the rest are randomly created solutions. Suksut et al. [18] introduced a support vector machine with restarting GA for classifying imbalanced data. In their study, a GA restarts when the fitness value of the new generation becomes less than that of the old population.

These restart strategies were not sufficient for solving complex multi-modal optimization problems using an RCGA. So, in this paper, a novel restart strategy is proposed and the performance of the same is demonstrated through ten multi-modal optimization case studies. Section 3 gives a detailed description of the developed restart strategy. Section 4 deals with the results and discussion, and the conclusions are presented in Sect. 5.

3 Developed Restart Strategy

To describe the developed restart strategy, we take the help of two terms, namely locally best solution (*l_best*) and globally best solution (*g_best*). The developed strategy consists of four different restart conditions. If any one of the conditions is found to be satisfied, the algorithm gets restarted. During restart, the next evolution begins with the *N* number of randomly created new solutions, where *N* is the population size of the GA. It is to be noted that no elite solution transfer takes place at the time of restart. The said conditions are as follows:

1st condition: At first, we assume an accuracy level of the problem, let say, 1e−14. For a particular generation, if the change in the fitness value of *l_best* is seen to be less than the stated accuracy level, then it is considered that there is no improvement of the *l_best* solution in that generation. Now, if the total number of such consecutive generations, say *m*, exceeds a predefined threshold generation value (say *th_gen*), then the GA gets restarted. The initial value of *m* is set equal to zero at the starting of an evolution and whenever there is an improvement in the *l_best*, *m* again takes the value of zero. So, the GA restarts, *if* $m > th\_gen$.

The task of deciding the value of *th_gen* is difficult, as it depends on the nature of the objective function's space. If the said parameter is set at a very low value, then it might happen that the algorithm gets restarted very frequently and the optimal points may not be reached. On the contrary, a too large value of *th_gen* can make the global search of a GA less efficient and time consuming. The thumb rule is as follows: if the complexity of the optimization problem is on the higher side, then the value of *th_gen* should be set at a lower value and vice-versa. A user can determine the value of *th_gen* using Eq. (1):

$$th\_gen = d_1 \times max\_gen, \tag{1}$$

where *max_gen* represents the maximum number of generations of an RCGA and d_1 is set equal to 0.03 through some trial experiments with the test functions studied in this paper.

2nd condition: In an evolution, the algorithm is supposed to count the number of generations (say, *c_gen*), where the value of m is found to be greater than zero. This

means that if the value of *m* is seen to be greater than zero in a certain generation, then the parameter (*c_gen*) is going to be incremented by one. When *c_gen* exceeds the value of threshold count parameter (say, *th_count*) and the value of *m* is also found to be greater than zero at that moment, then the RCGA prepares for a new start. This condition of restart can be written as follows:

If ($c\_gen > th\_count \,\&\&\, m > 0$), there is a restart of the algorithm.

Similar to *m*, the value of *c_gen* is initialized to zero in every evolution. For setting the value of *th_count*, the same thumb rule, as in case of *th_gen*, is applicable. Nevertheless, the value of *th_count* can be estimated using Eq. (2), as given below.

$$th\_count = d_2 \times max\_gen, \tag{2}$$

where d_2 is set equal to 0.1 through some trial experiments with the test functions.

3rd condition: Another situation of the restart for RCGA occurs, when the value of *c_gen* exceeds a pre-fixed parameter (say, *th_count_1*) and the objective function value of *l_best* is seen to be inferior to the same of *g_best*. This condition of restart for a minimization problem can be expressed as follows:

If ($c\_gen > th\_count\_1 \,\&\&\, l\_best > g\_best$), there is a restart of the algorithm.

The probable value of the parameter *th_count_1*, can be evaluated using Eq. (3):

$$th\_count\_1 = d_3 \times th\_count, \tag{3}$$

where d_3 has been assigned a value of 0.3 through some trial experiments with the test functions considered in this study.

4th condition: This condition is designed to avoid revisiting either the local or global optimum basin. It is obvious that if an RCGA can detect and keep away from the already visited local basins, then the probability of reaching the globally optimal solution will be more, as the search process will be extended over more number of unvisited regions in the variable space. For the purpose of detecting, whenever the algorithm restarts after satisfying any of the above-mentioned first three conditions, the obtained locally best solution (*l_best*) is memorized. In the next evolution, for every generation, the Euclidean distances are calculated between the presently available *l_best* solution and the previously memorized locally best points. If any one of the Euclidean distances is found to be less than a pre-allocated threshold value (say, *th_val*) and the *l_best* is seen to be inferior to the *g_best* in terms of the objective function value, then the next restart occurs. Here, one important point is to be noted that if the algorithm gets restarted under this 4th condition, then the locally best solution is not going to be captured for that evolution, as it is a revisited one. Also, another significant point to be noted is that, this condition is not going to be satisfied for the first evolution of the RCGA.

The selection of the best value for the parameter *th_val* is very difficult. However, a method for calculating this value from the problem information can be stated as follows:

1st Step: At first, the ranges of all the variables are calculated. The range of a variable is nothing but the absolute difference between the given boundaries for that variable. Next, the maximum value among all the ranges is found out and it is denoted as *max_range*.

2nd Step: Another parameter, say *s_range*, is calculated. This is done after multiplying all the ranges by a factor of d_4 and then, by taking the sum of these values.

$$s\_range = \sum_{i=1}^{k} \left(d_4 \times range_i\right), \tag{4}$$

where k is the total number of variables of the optimization problem and $range_i$ represents the range for ith variable. Here, a suitable value of d_4 is obtained through some preliminary experiments with the test functions studied in this paper and it is found to be equal to 0.04.

3rd Step: In this step, the minimum value in between the *s_range* and $\left(max\_range \times d_5\right)$ is determined. This obtained value is considered as the threshold value (*th_val*) of the problem, which is calculated as follows:

$$th\_val = minimum\left(max\_range \times d_5, s\_range\right), \tag{5}$$

where d_5 is seen to be equal to 0.4 through some trial experiments.

From the above description of the restart strategy, it can be comprehended easily that the last two conditions (i.e., 3rd and 4th) are never going to be satisfied during the first evolution of an RCGA, as *l_best* and *g_best* solutions are found to be same for that period. The novelty of the proposed strategy lies with the developed restart conditions, specially from 2nd to 4th. These are totally new types of conditions used to restart an RCGA and simultaneously, they are very efficient, robust and easy to implement. Nevertheless, it is important to note that the values of d_1 through d_5 of the Eqs. (1) through (5), respectively, are dependent on the nature of test functions.

4 Results and Discussion

For the purpose of evaluating the performance of the proposed restart strategy for the RCGA, ten complex multi-modal test functions have been chosen from the CEC 2005 benchmark test functions [13]. These functions were designed from some classical optimization functions by rotating, shifting or hybridizing. These operations increased the complexity of these functions. Table 1 shows the list of these selected test functions with their variable boundaries and global minimum objective function values $\left(f_{min}\right)$.

The experiment is carried out with a standard elitism-based RCGA, equipped with tournament selection [19], simulated binary crossover operator [20] and polynomial mutation operator [21]. The RCGA has been run with four different strategies, such as

- Proposed Restart Strategy of this paper (say, St-1)
- Restart strategy suggested by Beligiannis et al. [15] (say, St-2)
- Restart strategy introduced by Dao et al. [17] (say, St-3)
- Without Restart (say, St-4)

Table 1. A list of ten complex multi-modal problems from CEC'05 benchmark test functions [13].

Functions	Hybrid Composition function	Bound	f_{min}
F01	Hybrid Composition function 1	[−5, 5]	120
F02	Rotated Hybrid Composition function 1	[−5, 5]	120
F03	Rotated Hybrid Composition function 1 with Noise in fitness	[−5, 5]	120
F04	Rotated Hybrid Composition function 2	[−5, 5]	10
F05	Rotated Hybrid Composition function 2 with a Narrow basin for the Global optimum	[−5, 5]	10
F06	Rotated Hybrid Composition function 2 with the Global optimum on the bounds	[−5, 5]	10
F07	Rotated Hybrid Composition function 3	[−5, 5]	360
F08	Rotated Hybrid Composition function 3 with High Conditioned Number Matrix	[−5, 5]	360
F09	Non-Continuous Rotated Hybrid Composition function 3	[−5, 5]	360
F10	Rotated Hybrid Composition function 4	[−5, 5]	260

4.1 Parameters' Settings

For the purpose of fair comparison, all the common parameters of RCGA, such as crossover probability $(p_c = 1.0)$, mutation probability $(p_m = 0.02)$, user index parameter for crossover $(\eta_c = 2.0)$, user index parameter for mutation $(\eta_m = 10)$, population size $(N = 50)$, number of variables $(k = 10)$, maximum number of fitness evaluation (taken as 1,00,000) and maximum number of generations ($max\_gen = 2000$) are kept fixed. The stopping criterion is set as the maximum number of fitness evaluation. Special parameters for St-1 like, *th_gen*, *th_count*, *th_count*_1 and *th_val*, are calculated using the Eqs. (1), (2), (3) and (5), respectively, and accuracy level of the problems is considered as 1e−14. Table 2 displays the values of these parameters.

Table 2. Special parameters' values for St-1

Special parameters	Values
th_gen	60
th_count	200
*th_count*_1	60
th_val	4

In St-2, after every 60 generations, the algorithm gets restarted and during a new evolution, randomly chosen 20% of the population of the last generation is inserted in the randomly created $(0.8 \times N)$ number of new solutions. In case of St-3, if there is no change in the best solution up to a consecutive number of 60 generations, the RCGA will go for a restart. During the restart, the best found solution (*g_best*) is going to be transferred into the next evolution and the rest (N−1) solutions are generated at random.

Each problem has been evaluated for 50 times with the same initial population. We have used the same seed in each run to serve this purpose. After each run, the gained globally best objective function value is captured and its absolute deviation from the

actual globally best objective function value is calculated. Next, the average of the obtained 50 deviations is found out. This type of experiment is carried out for all the ten problems using four different strategies, as mentioned earlier. Table 3 presents a comparison of the average deviations for these selected test functions (best results are shown in bold).

Table 3. Comparison of the average deviations obtained using RCGA with four different strategies

Function	St-1	St-2	St-3	St-4
F01	**1.308E−02**	8.564E+00	1.682E+01	1.702E+01
F02	**1.233E+02**	1.298E+02	1.320E+02	1.334E+02
F03	1.298E+02	1.301E+02	1.354E+02	**1.297E+02**
F04	**7.639E+02**	9.551E+02	8.787E+02	9.595E+02
F05	**7.610E+02**	9.493E+02	9.393E+02	9.652E+02
F06	**7.608E+02**	9.589E+02	9.351E+02	9.713E+02
F07	**4.682E+02**	9.735E+02	9.657E+02	9.895E+02
F08	**7.619E+02**	8.278E+02	7.901E+02	8.303E+02
F09	**5.353E+02**	1.097E+03	8.912E+02	1.104E+03
F10	**2.000E+02**	3.887E+02	3.345E+02	4.698E+02

From the comparison of these results, it is clear that the proposed restart strategy (St-1) has outperformed the other three strategies for nine problems out of ten (that is, except in case of F03). Moreover, in case of F03, the obtained value of deviation yielded by St-1 is found to be comparable with the best value gained using St-4. Figure 1 depicts the variations in absolute deviations of the obtained *g_best* for 50 runs of the four strategies. From this figure, it is clear that the RCGA with the proposed restart strategy, is able to yield the better results for most of the times compared to the other three strategies. The superiority of the proposed strategy lies with the construction of the efficient restart conditions. This strategy is capable of detecting and avoiding the already visited locally or globally optimal points. It enhances the global search capability of an RCGA within the given maximum number of fitness evaluations. Nevertheless, as there is no elite solution transfer takes place during a restart, a new evolution is not influenced by the previously found best solutions. This makes each search stage independent of the others.

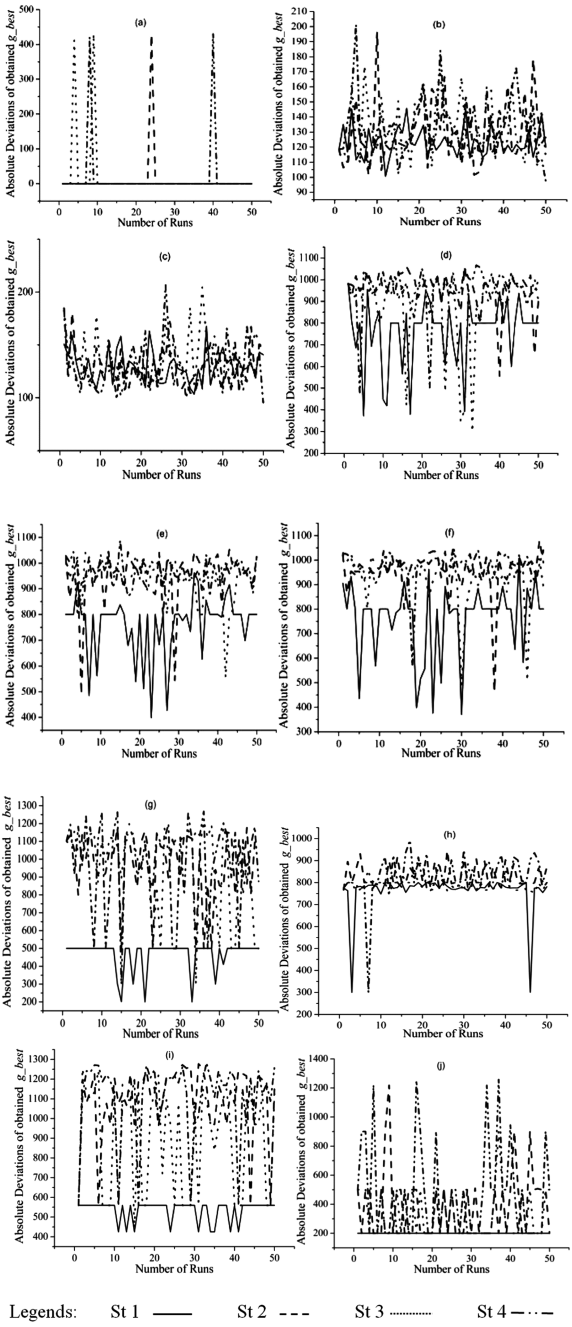

Fig. 1. Absolute deviations of obtained globally optimal function values in 50 runs: (a) F01, (b) F02, (c) F03, (d) F04, (e) F05, (f) F06, (g) F07, (h) F08, (i) F09 and (j) F10.

5 Conclusions

In this study, a novel restart strategy of an RCGA for solving complex multi-modal optimization problems has been proposed and presented. Four types of conditions for restarting have been embedded in the scheme introduced. This strategy is capable of detecting and avoiding the revisiting of any local or global optimum basin. Also, the proposed restart scheme has been designed in such a way that it is able to increase the global search capability of an RCGA and due to this fact, the probabilistic guarantee of an RCGA of finding out the globally optimal solution is enhanced by a considerable amount. From the CEC 2005 benchmark functions, ten multi-modal problems have been selected and experiments are carried out to test the performance of the developed strategy of restarting. The results of the proposed scheme are compared with that of the other three schemes.

The proposed strategy of restarting has yielded the better results compared to other three schemes for nine test functions out of ten. However, the performance of the developed strategy is decided by a number of parameters, whose numerical values are dependent on the test functions to be studied. The performance of the proposed scheme of restarting will be tested in future on other CEC 2005 benchmark test functions and some practical problems related multi-modal optimization.

References

1. Liberti, L., Kucherenko, S.: Comparison of deterministic and stochastic approaches to global optimization. Int. Trans. Oper. Res. **12**(3), 263–285 (2005)
2. Moles, C.G., Mendes, P., Banga, J.R.: Parameter estimation in biochemical pathways: a comparison of global optimization methods. Genome Res. **13**(11), 2467–2474 (2003)
3. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
4. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
5. Das, S., Suganthan, P.N.: Differential evolution: a survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)
6. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **26**(1), 29–41 (1996)
7. Yang, X.-S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
8. Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp. 65–74. Springer, Heidelberg (2010)
9. Wang, Y., Huang, J., Dong, W.S., Yan, J.C., Tian, C.H., Li, M., Mo, W.T.: Two-stage based ensemble optimization framework for large-scale global optimization. Eur. J. Oper. Res. **228**(2), 308–320 (2013)
10. Ng, C.-K., Li, D.: Test problem generator for unconstrained global optimization. Comput. Oper. Res. **51**(Suppl. C), 338–349 (2014)
11. dos Santos Coelho, L., Ayala, H.V.H., Mariani, V.C.: A self-adaptive chaotic differential evolution algorithm using gamma distribution for unconstrained global optimization. Appl. Math. Comput. **234**(Suppl. C), 452–459 (2014)

12. Boender, C.G.E., Romeijin, H.E.: Stochastic methods. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization. Kluwer Academic Publishers, Boston (1995)
13. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore, May 2005 and KanGAL Report 2005, IIT Kanpur, India (2005)
14. Ghannadian, F., Alford, C., Shonkwiler, R.: Application of random restart to genetic algorithms. Inf. Sci. **95**(1), 81–102 (1996)
15. Beligiannis, G.N., Tsirogiannis, G.A., Pintelas, P.E.: Restartings: a technique to improve classic genetic algorithms' performance. In: International Conference on Computational Intelligence 2004, pp. 404–407 (2004)
16. Hughes, J.A., Houghten, S., Ashlock, D.: Recentering and restarting a genetic algorithm using a generative representation for an ordered gene problem. Int. J. Hybrid Intell. Syst. **11**(4), 257–271 (2014)
17. Dao, S.D., Abhary, K., Marian, R.: An improved structure of genetic algorithms for global optimisation. Prog. Artif. Intell. **5**(3), 155–163 (2016)
18. Suksut, K., Kerdprasop, K., Kerdprasop, N.: Support vector machine with restarting genetic algorithm for classifying imbalanced data. Int. J. Futur. Comput. Commun. **6**(3), 92 (2017)
19. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. Found. Genet. Algorithms **1**, 69–93 (1991)
20. Agrawal, R.B., Deb, K.: Simulated binary crossover for continuous search space. Complex Syst. **9**(2), 115–148 (1995)
21. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Comput. Sci. inf. **26**, 30–45 (1996)