An adaptive Parameters Binary-Real Coded Genetic Algorithm for Real Parameter Optimization:

Performance Analysis and Estimation of Optimal Control Parameters

Omar Abdul-Rahman[1], Masaharu Munetomo[2] and Kiyoshi Akama[3]

**[1] Graduate School of Information Science & Technology, Hokkaido University
Sapporo, Hokkaido, Japan**

**[2] Information Initiative Center, Hokkaido University
Sapporo, Hokkaido, Japan**

**[3] Information Initiative Center, Hokkaido University
Sapporo, Hokkaido, Japan**

Abstract

Genetic algorithms (GAs) are vital members within the family biologically inspired algorithms. It has been proven that the performance of GAs is largely affected by the type of encoding schemes used to encode optimization problems. Binary and real encoding schemes are the most popular ones. However, it is still controversial to decide the superiority of one of them for GAs performance. Therefore, we have recently proposed binary-real coded GA (BRGA) that has the ability to use both encoding schemes at the same time. BRGA relies on a parameterized hybrid scheme to share the computational power and coordinate the cooperation between binary coded GA (BGA) and real coded GA (RGA). In this article, we use CEC'2005 benchmark suite of 25 functions to analyze quality and time performance of BRGA and in comparison with original binary and real coded component GAs. To demonstrate the performance of BRGA, we compare it with the performance of some other EAs from the literature. In addition, we implement a robust parameter tuning procedure that relies on techniques from statistical testing, design of experiments and Response Surface Methodology (RSM) to estimate the optimal values for control parameters that can secure a good performance for BRGA against specific problems at hand.

Keywords: *Binary coded GA(BGA), Real coded GA(RGA), Hybrid Scheme, Design of Experiments.*

1. Introduction

Genetic algorithms (GAs) are efficient search metaheuristics that mimic natural evolution and play an increasingly important role in a variety of fields and applications like bioinformatics, computational science, engineering, economics, chemistry, manufacturing and other fields. In order to employ GAs effectively, the optimization problem should be encoded by an appropriate encoding scheme. A variety of encoding schemes are available within the literature; however, binary and real (or floating point) encoding schemes are still the most popular among researchers and widely employed in a variety of applications.

Theoretical and empirical investigations confirmed that that the performance of GAs is greatly affected by the type of the encoding scheme. However, the superiority of either binary or real encoding schemes for the performance of GAs is that kind of open question that for many years has and still divides the GA community. For theoreticians, BGA is the answer. The theoretical finding of schemata theory supports that enhanced schemata processing is obtained by using the alphabet 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 possible to argue that BGAs 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, RGAs are preferred by many practitioners. They are increasing in usage since the floating point representation is conceptually closest to the real design space, and moreover, the string length is reduced to the number of design variables. RGAs are robust, accurate, and efficient. However, it is possible to argue that RGAs 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 the theory of RGAs is still far from providing plausible understanding of internal RGAs mechanisms, which is a true hindrance to further development of advanced techniques in this field.

In order to tackle this problem effectively, we have recently proposed a hybrid binary-real coded genetic

algorithm (BRGA) [1] [2]. The main idea in BRGA is to give the algorithm the ability to process the optimization problems in both encoding schemes, binary and real, at the same time. By combing the usage of both encoding schemes, we aim to maximize the advantages and minimize the disadvantages resulted from employing each encoding scheme separately. BRGA employs a hybrid scheme that organizes the interactions and divides the computation power between the two cooperatives versions of GAs (BGA and RGA). The evolutionary search is primarily guided by BGA 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 conducting an extensive search through these regions. The interaction between the two versions of BRGA is regulated by adaptive parameters which have small values at the beginning of the search to allow the exploration of the search space by the BGA part, while, their values increase 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. The related literature is briefly reviewed in Section 2. The algorithmic details of BRGA are described in Section 3. The implemented numerical evaluation is explained and the obtained results are discussed and analyzed in Section 4. On the other hand, the implemented parameter tuning procedure is described and the obtained optimal configurations are discussed in Section 5. Finally, in Section 6, we conclude the paper and highlight possible directions of future research.

2. Literature Review

From a conceptual standpoint, the design issue in BRGA adopts an idea that is closely related to dynamic coding, which is a sophisticated approach to altering the coarseness of search spaces. An example of such an approach is the stochastic genetic algorithm presented by Krishnakumar et al [3]. 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, BRGA employs an RGA instead of ES to adapt the regions represented by the BGA. In addition, stochastic GA treats each parameter within the chromosome as a separate entity, while in BRGA the sampling happens from the region bounded by the promising regions collectively. This can approach can be more efficient in terms of computational power. Moreover, BRGA relies on sources other than random sampling (like best members from the old population) in guiding the optimization process.

Another example of dynamic coding is the adaptive range genetic algorithm (ARGA) as presented by Arakawa and Hagiwara [4]. In ARGA, the mapping rules from binary to real strings are updated during the optimization process according to the population statistics in order to adapt the population toward promising design regions. However, BRGA works differently when it employs an RGA that takes random samples from the promising regions found by the BGA in order to adapt the population toward promising regions in the search space.

Recently, the authors in [5] proposed an adaptive resolution micro-GA with Tabu search to solve a group of extremely difficult non-convex mix integer non-linear programming (MINLP) benchmark functions. The problem domain is divided by into discrete part to be handled by Binary GA operators and real part to be handled by Real coded GA operators. The algorithm employs an adaptive resolution which is a recursive approach that divides the search space into sub-solution spaces. The size of theses sub-solutions are controlled by information entropy. Under such scheme, the areas under the better solution have a greater chance to find even better solutions. The algorithm utilizes local search operator to increase the efficiency of the algorithm and a Tabu search like operator to eliminate redundancy in revisiting already visited local minima. So, in contrast to BRGA, the hybrid scheme in [5] used real coded algorithms to change the coarseness of the search. By using a combination of different encoding schemes, BRGA is in a better position to reap the best from the combinations.

From hybrid scheme standpoint, BRGA employs a scheme that can be described as sequential hybrid scheme [6]. Since, the cooperating algorithms in BRGA are exchanging output several times, operating in a time-interleaving manner. However, hybrid binary-real coded GAs in the literature generally follows a parallel hybridization scheme by separating problem domain into a binary part to be handled by BGA and a real part to be handled by RGA. An example is the hybrid binary real coded GA proposed by Barrios et al. [7]. to be used for designing and training feed-forward artificial neural networks. Their algorithm employs two interconnected GAs that work in parallel to design and train better neural networks to solve the problem at hand. In addition, the authors in [8] used RBLGA, which is a combination of real coded and binary like coded genetic algorithm to automatically generate Fuzz Knowledge Bases (FKBs) from a set of numerically data. The generated FKBs satisfied a contradictory paradigm in terms of KFBs high precision and simplicity. The manual construction of such FKBs is a long and tedious process, while the successful generation of such FKBs is a prerequisite for successful operation of Fuzzy Decision Support System (FDSS). In RBLGA, the problem domain is divided into real part, which is the fuzzy set repartition, and binary part, which is the fuzzy rule base. By adopting sequential hybridization scheme for BRGA, we believe that synergetic effects can be more productive in guiding optimization toward better performance.

3. The BRGA algorithm

A schematic diagram that clarifies the operation of BRGA is shown in Fig. 1. BRGA employs a hybrid scheme that organizes the interactions and divides the computational power among two cooperatives version of GAs, namely

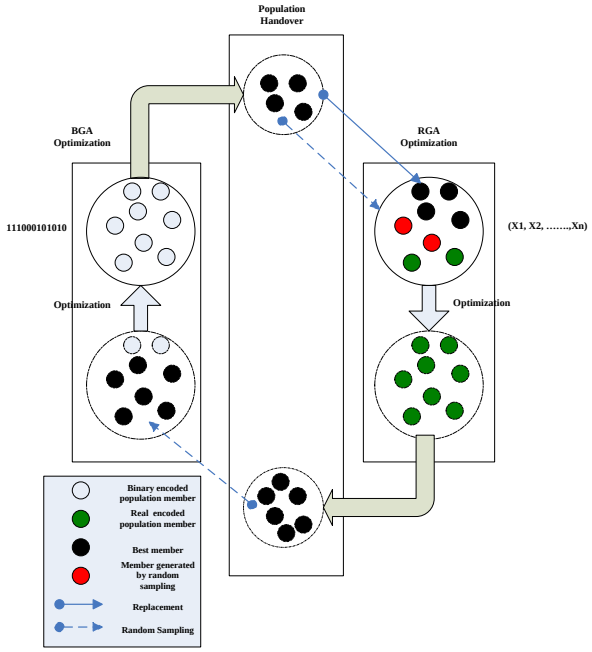


Fig. 1 Typical optimization cycle in BRGA.

Binary coded GA part (BGA part) and Real coded GA part (RGA part). The evolutionary search is primarily guided by BGA part, which is used to identify promising regions within search space. According to the schema theory [9], using of low cardinality alphabet is behind the success of the BGAs in finding good solutions. So, under our proposed hybrid scheme, the BGA part plays a vital role in scanning the search space effectively and partitioning it potentially into smaller promising parts that can be exploited later by the RGA part. Whenever the BGA part completes its shares of genetic search, the computational power switches from the BGA part to the RGA part by a process that can be best described as population handover. During population handover, the BGA part injects some of its best population's members into the RGA part's population of solutions. Another sample of injected members is randomly generated from the region bounded by those best members. So, population handover is an opportunity for the RGA part to focus its search by receiving valuable feedback from the BGA part about the promising regions within the search space. The RGA part, which has the ability to process problem variables directly, can effectively compensate the poor ability of BGA in finding accurate solutions resulted from the limitations of mapping from the real problem space into the binary representation space and vice versa. Starting from a population doped by promising members, the RGA part extensively exploits this region and adapts population members toward it in order to identify solutions of higher quality. Whenever the RGA part completes its share of genetic search, the computational power switches back to the BGA part by a reverse process of population handover. Here, the RGA part injects a portion of its best population members into the BGA part population of solutions. So, the handover process is an opportunity for BGA part to correct its behavior in scanning and partitioning search space by receiving valuable feedback from the RGA part about the optimality of regions that suggested as promising in the previous cycle. So, the evolutionary search continues in the same manner up to satisfaction of the termination criteria. In order to regulate the share of each part from the total computational power, the frequency of handover process, amount and type of information exchange, BRGA employs a hybrid scheme which relies on selection criteria and a group of static, dynamic and adaptive parameters. In this section, we aim to describe the implementation details and discuss the operation of BRGA. The main components of BRGA can be described as follows.

3.1 Chromosome representation

Both binary and floating point encoding schemes are used to represent chromosomes of two equivalent populations of candidate solutions. These populations are frequently updated by mapping from one version to another during the evolutionary cycle.

In BRGA, the initial population is usually generated as real numbers and quantization method is used to map the chromosomes from the floating point format to the binary format and vice versa. Quantization begins by sampling a function (i.e. an optimization problem at hand) and placing the samples into equal quantization levels, which are nonoverlapping subranges. Then, the samples will be assigned unique discrete values equal to the mid values of the quantization levels. The difference between actual value and quantized value is known as quantization error. It also possible to assign the samples values equal to the highest and lowest values of the quantization level. However, using the mid value of the quantization level is better in terms of generated quantization errors. [10]

Supposing that we have a population of initial solutions, where P is the P_{th} chromosome from that population and P_n is the *n*th variable (or gene) within P. Then, the mathematical formulas for binary encoding and decoding of P_n can be described as the following:

For encoding;

$$P_{norm} = \frac{P_n - P_{lo}}{P_{hi} - P_{lo}} \quad (1)$$

$$gene[\mathrm{m}] = \mathbf{round}\left\{P_{norm} - 2^{-m} - \sum_{p=1}^{m-1} gene[p]2^{-p}\right\} \quad (2)$$

For decoding ;

$$P_{quant} = \sum_{m=1}^{M} gene[m]2^{-m} + 2^{-(M+1)} \quad (3)$$

$$q_n = P_{quant}(P_{hi} - P_{lo}) + P_{lo} \quad (4)$$

In each case, P_{norm} is the normalized variable, $0 \leq P_{norm} \leq 1$. P_{lo} is the minimum value within the variable P_n range. P_{hi}, is the maximum value within the variable P_n range. gene[*m*], is the binary version of P_n. round{.} is round to the nearest integer equal to or greater than argument. M is

the number of assigned bits to represent the variable P_n. P_{quant} is the quantized version of P_{norm}. Finally, q_n is the quantized version of P_n.

It is clear from Eq. (3) that increasing the number of bits will decrease quantization error. Actually, such direct relationship between quantization error and number of bits is one of the obstacles that limit the performance of BGAs whenever applied to real parameters optimization problems.

3.2 Binary coded GA (BGA)

In this study, we use a standard BGA as described by Haupt and Haupt [10] to implement the BGA part. A simple implementation adopted for the BGA to keep the complexity of BRGA as minimum as possible. The main components of the BGA are the mating operators (single point crossover operator and one point mutation operator) and the rank weighting selection scheme with elitism.

Selection is survival-of-the-fittest mechanism used by the GAs to allocate more offspring to the better individuals, and the rest thrown away from the evolutionary process. There are a variety of schemes that simulate this mechanism; however, roulette wheel selection is the most common among them. With this approach the probability of selection is proportional to an individual's fitness. Individuals with better fitness have greatest probabilities to be selected to the next generations, while those with weakest fitness have a very low probability of propagation to the next generations. In BGA part, a version of this mechanism, known as roulette wheel weighting, was implemented. This version is problem independent and finds a probability from the rank, *r*, of the individual as follows.

$$P_r = \frac{N_{keep} - n + 1}{\sum_{n=1}^{N_{keep}} n} \quad (5)$$

$$\mathrm{N_{keep}} = \mathbf{round}\{S.N_{pop}\} \quad (6)$$

Where, *n* is the individual rank according to its fitness, S is a user define parameters that takes a value from the range 0 < S < 1, **round{.}** is round to the nearest integer equal to or less than argument and N_{pop} is a user defined parameters refers to the population size. Therefore, each individual receives a cumulative probability according to its rank defined as the following.

$$\mathrm{P_j} = \sum_{r=1}^{j} \mathrm{P_r} \quad (7)$$

So, the selection scheme proceeds by firstly generates a random number between zero and one. Then, by starting from the top of the list, the first chromosome with a cumulative probability that is greater than or equal to the random number is selected for the mating pool. This process is repeated several times until N_{keep} is satisfied. And the population members within the mating pool are arranged in pairs to be processed later one by the mating operators.

On the other hand, The BGA part employs a single point crossover and one point mutation as mating operators. BGAs usually use crossover operator as a method for exploration of new parts of the search space by combining information from different solutions. In the single point crossover, pairs of individuals are selected from mating pool to produce offspring. The single point crossover proceeds by cutting the pair of selected strings at a randomly selected locus, or crossover point, and swapping the tails to create child strings. The probability of applying crossover operator can be estimated from the following:

$$\mathrm{P_{cross}} = \frac{\mathrm{N_{pop}} - \mathrm{N_{keep}}}{\mathrm{N_{pop}}} \quad (8)$$

On the other hand, mutation is another operator used by BGAs to explore fitness landscape. It acts as a method to maintain diversity by introducing new traits in the original population and keeps the GA from converging too fast before sampling the entire cost surface. A single point mutation simply changes a 1 to a 0, and vice versa. Mutation points are randomly selected from the total number of bits in the population matrix. The total number of these points can be deduced from the following formula.

$$No.of\ Mutations = P_m * (N_{pop} - 1) * N_{bits} \quad (9)$$

Where P_m is mutation probability and N_{bits} is number of bits per gene (or variable). It is clear from Eq. (9) that the best chromosome is excluded from mutation process due to the elitism. Actually, Keeping the best, or the elite, individual between generations is another technique used by the BGA part to avoid converging too fast to and trapping to the local minima.

3.3 Real-coded GA (RGA):

RGAs do not suffer from the problem of quantization limitation that faces the BGAs. Therefore, RGAs have a better ability to find solutions of better precision ability. Due to the importance of this part, we use Unimodal Normal Distribution (UNDX) as an advanced real crossover operator [11] with the minimal generation gap (MGG) [12] as a generation alteration model to implement the RGA part. For RGAs, It has been accepted by the researchers that mutation operators are the most important operators and they can act as main search operators, while crossover operators can perform only assistant roles. However, UNDX along with BLX-α[13], Fuzzy recombination (FR) [14], and Simulated Binary crossover (SBX) [15] are a group of advanced crossover operators that have been developed for RGAs which show relatively good performance and ability to operate independently from mutation operators. When compared with other RGAs` crossover operators, UNDX generates new population lie on some ponds or along some valleys in order to focus the search on promising areas from a viewpoint of search efficiency. So, UNDX can efficiently optimize functions especially those have epistasis among parameters. Moreover, one dimensional UNDX with MGG has shown good performance on several benchmark problems [11].

UNDX procedure can be briefly explained as follows.

(1). Select 3 individuals as parents' g_{p1}, g_{p2} and g_{p3} from the population of solutions. Where g_{p1}, g_{p2} are the main parents and g_{p3} is a subparent.
(2). Define the middle point of the main parents g_m, where $g_m = \frac{(g_{p1}+g_{p2})}{2}$
(3). Let the direction from gp1 to gp2 be primary search line d, where $d = g_{p2} - g_{p1}$
(4). Define D as the distance from g_{p3} to primary search line direction (d). Where $D^2 = |g_{p3} - g_{p1}|^2 \left(1 - \left(\frac{(g_{p3}-g_{p1})^T(g_{p2}-g_{p1})}{|g_{p3}-g_{p1}||g_{p2}-g_{p1}|}\right)^2\right)$
(5). Define e_i be the orthogonal basis vectors spanning the subspace perpendicular to the primary search direction (d).
(6). Two children g_{c1} and g_{c2} are now generated as follows;

$$g_{c1} = g_m + \xi d + D\sum_{i=1}^{n} \eta_i e_i \quad (10)$$

$$g_{c1} = g_m - \xi d - D\sum_{i=1}^{n} \eta_i e_i \quad (11)$$

$$\xi \sim N(0,\sigma_\xi^2),\ \eta_i \sim N(0,\sigma_\eta^2) \quad (12)$$

Where n is the dimension of search space, *N (a,v)* represents a normal distribution whose average is *a* and whose variance is *v*, σ_ξ and σ_η are constant parameters that are recommended to be set as (1/2) and ($0.35/\sqrt{n}$) respectively.
It has been found that minimal generation gap, or MGG, is appropriate for keeping the diversity of the population. The MGG model consists of the following five steps.

(1). Generate an initial population.
(2). Select a pair of individuals randomly from the population as parents.
(3). Generate $2N_{cross}$ offspring by carrying out N_{cross} time's crossover. Where N_{cross} is a user defined parameter.
(4). Select two individuals from the family containing the two main parents and their $2N_{cross}$ offspring. The first individual is the best one within the family, while the second individual is one selected by the roulette wheel selection (similar to that in BRGA). Replace the two main parents in step (2) with the two individuals.
(5). Repeat the procedure from step (2) to step (4) until a stopping condition is satisfied.

3.4. Population handover:

The main contribution of our work is in the population handover mechanism, which is the core of the hybrid scheme of the BRGA. Due to the using of low alphabet cardinality, BGA has a better possibility to identify promising regions within search space. So, by receiving feedback from BGA through population handover, RGA can approach optimal solution regions rapidly by exploiting these promising regions. On the other hand, RGA has a better ability to obtain better quality solutions. So, BGA can adapt it behavior in scanning the search space based on feedback received from RGA part through population handover. So, during a typical optimization cycle, the populations of real and binary solutions are kept updated by frequently mapping from one version to another and vice versa. Population handover is the responsible on controlling the mapping process in itself and the frequency of its occurrence. The pseudocode of handover procedure is shown in Fig. 2.
As it is clear from Line 0 in Fig. 2, the handover process happens at moments controlled by the by K. K is user defined dynamic parameter that takes an initial value of K_i and an incremental step size of K_s as shown in the equation of Line 43.Whenever handover process starts, the computational power switches from BGA part to RGA part and BRGA starts by updating handover process counter, T, and stores the final population which generated by BGA, POP_{BGA}^{T}, to be used in the subsequent handover steps. BRGA uses two user defined parameters, which are Φ to evaluate the quality of improvement achieved in POP_{BGA}^{T} in comparison with POP_{RGA}^{T-1} and Θ to control the influx of information exchange between the BGA and RGA parts. It should be noted that POP_{RGA}^{T-1} is the final population generated by RGA part from the last handover cycle. In case of initial handover cycle it equals the initial randomly generated solutions. Both POP_{BGA}^{T} and POP_{RGA}^{T-1} are sorted according to their penalized fitness values. Taking into consideration that the framework in Fig. 2 has been developed for the minimization problems, BRGA starts a binary-to-real population handover considering the following selection criteria:
(1). The mean performance of BGA part is *better* than the mean performance of RGA part, i.e. logic in Line 3 and Line 4 are true. This scenario can be especially true at the initial stages of the evolutionary search where BRGA usually moves within regions far away from the optimality region. Here, the BGA part has a good opportunity in frequently identifying new promising search regions and adapts population towards them. If this happens, then BRGA wants from the RGA part to extensively exploit these regions so it increases the influx of data migration between the two parts by setting the value of Θ to its maximum (Lines 6 and 11) and directs RGA part to replace the solution members of POP_{RGA}^{T-1} by new members from POP_1 and POP_2 (Lines 7-9, 12-14) where POP_1 is a fraction of POP_{BGA}^{T} best members, while POP_2 is the random samples taken by BRGA from the region defined by POP_1. By taking samples from this region, BRGA hopes that RGA part can locate rapidly higher quality solutions. Finally, the logic in Line 5 is introduced to protect the elite member within POP_{RGA}^{T-1}, if existed, from being destructed by the above mentioned replacement process.

```
0. IF K = iga THEN
1. T = T + 1;
2. POP^T_BGA ← POP^K_BGA;
3.      IF Mean(f(POP^T_BGA)) < Mean(f(POP^{T-1}_RGA)) THEN
4.          IF Mean(f(POP^{T-1}_RGA)) - Mean(f(POP^T_BGA)) >= Φ THEN
5.              IF Min(f(POP^{T-1}_RGA)) < Min(f(POP^T_BGA)) THEN
6.                  Θ = 1/2 popsize;
7.                  POP_1 ← POP^T_BGA(2:Θ, 1:n);
8.                  POP_2 ← Random Sampling (Θ, POP_1);
9.                  POP^{T-1}_RGA(2:m, 1:n) ← Replacemnet (POP_1, POP_2);
10.             ELSE
11.                 Θ = 1/2 popsize;
12.                 POP_1 ← POP^T_BGA(1:Θ, 1:n);
13.                 POP_2 ← Random Sampling (Θ, POP_1);
14.                 POP^{T-1}_RGA(1:m, 1:n) ← Replacemnet (POP_1, POP_2);
15.             END;
16.         ELSE
17.             Θ = 1/4 popsize;
18.             POP_1 ← POP^T_BGA(1:Θ, 1:n);
19.             POP_2 ← Random Sampling (Θ, POP_1);
20.             POP^{T-1}_RGA(1/2 m + 1: m, 1:n) ← Replacemnet (POP_1, POP_2);
21.     ELSE
22.         IF Mean(f(POP^T_BGA)) - Mean(f(POP^{T-1}_RGA)) <= Φ THEN
23.             Θ = 1/4 popsize;
24.             POP_1 ← POP^T_BGA(1:Θ, 1:n);
25.             POP^{T-1}_RGA(3/4 m + 1: m, 1:n) ← Replacemnet (POP_1);
26.         END;
27.     END;
28.     FOR i = 1: Γ
29.         Apply RGA(POP^{T-1}_RGA);
30.     END;
31.     POP^T_RGA ← POP^Γ_RGA;
32.     IF Mean(f(POP^T_RGA)) <= Mean(f(POP^T_BGA)) THEN
33.         Θ = popsize;
34.         POP_1 ← POP^T_RGA(1:Θ, 1:n);
35.         POP^T_BGA(1:m, 1:n) ← Replacemnet (POP_1);
36.     ELSE
37.         IF (Mean(f(POP^T_RGA)) - Mean(f(POP^T_BGA))) <= Φ THEN
38.         Θ = 1/4 popsize;
39.         POP_1 ← POP^T_RGA(1:Θ, 1:n);
40.         POP^T_BGA(3/4 m + 1: m, 1:n) ← Replacemnet (POP_1);
41.         END;
42.     END;
43.     K = K_i + K_s;
44.     Γ = Γ_i + Γ_s;
45. END;
```

Fig. 2 Population handover pseudocode.

(2). The mean performance of BGA part is *slightly better* than the mean performance of RGA part, i.e. the logic in Line 3 is true while the logic in Line 4 is false. As the evolutionary search progresses, the probability of identifying new promising regions decreases. So, the BGA part gradually losses its ability in significantly pushing the mean performance of the population of member solutions. However, since the BGA part still exhibits signs of improvement, BRGA still hopes that RGA can find higher quality solutions by accepting new best members of POP_1 (Line 18) and POP_2 or the random samples generated from the region defined by POP_1 (Line 19). However here, BRGA decreases the influx of data migration from BGA part to RGA part (Line 17) and it directs RGA part to replace only the worst half of its population members (POP^{T-1}_{RGA}) by POP_1 and POP_2 (Line 20).

(3). The mean performance of BGA part is *slightly lags* behind the mean performance of RGA part, i.e. the logic in Line 3 is false while the logic in Line 22 is true. This situation indicates that the performance of both parts of BRGA becomes comparable to each other and the RGA starts gradually to dominate the performance of BRGA. Here, BRGA becomes less interested in accepting new members from BGA part. So, it keeps the low influx of data migration from the BGA part toward the RGA part (Line 23) and it directs the RGA part to replace only the worst quarter of its population member (Line 25) by POP_1 (Line 24). By increasing the percentage of good members within POP_{RGA}^{T-1}, BRGA hopes to fasten the convergence profile towards optimality region.

(4). The mean performance of BGA part is *lags* behind the mean performance of RGA part, i.e. the logic in Lines 3 and 22 are false. This scenario can be true especially at final stages of the evolutionary search, where the BRGA comes within the vicinity of the optimality region. Here, there is a very low probability for the BGA part to find even better promising regions by visiting new parts of the search space. Whenever this scenario happens, BRGA stops data migration and it directs the RGA part to use its un-updated POP_{RGA}^{T-1} in the subsequent search process.

On the other hand, the pseudocode in Fig. 3 describes the random sampling process. The random sampling process takes POP_1 and Θ as an input from the handover process (Lines 8, 13 and 19, Fig. 3). The process starts by creating X_{lo} and X_{hi} vectors which contain the minimum and maximum population members for each search dimension within POP_1 respectively. These vectors draw the boundary of the regions from which random samples to be taken by BRGA. The random samples are generated according to the equation in Line 7. Where, N is pseudorandom value drawn from the standard normal distribution. The elements within X_{lo} and X_{hi} should be distinct from each for sampling process to be implemented effectively. However, as the evolutionary search progresses the diversity within the population members decreases gradually. Actually, it is highly possible that populations of best members become replicas of the same solution at some moments during the evolutionary search. To guard against such situation, logic in Line 3 is introduced. It shifts an X_{lo} element by an amount determined by Ω from a correspondent element within X_{lo}. Ω is a real valued user defined parameter that takes values greater than zero.

So, whenever the process of binary-to- real handover process completes, the RGA part starts from POP_{RGA}^{T-1} a fresh search with iteration budget controls by Γ. Γ is a dynamic user defined parameter that takes an initial value of Γ_i and an incremental step size of Γ_s as shown in the equation in Line 44. As discussed above, Γ gives the RGA part small fractions from total computational budget of BRGA at the initial stages of the evolutionary search to allow an extensive exploration of the search space by the BGA part. On the other hand, it increases gradually to allow the RGA part to concentrate more on the exploitation of the promising search space regions by the final stages of the evolutionary search. Therefore, K and Γ are the deceive factors in dividing the total computational budget available to BRGA between the BGA and the RGA parts.

So, when the RGA part completes its iteration budget, BRGA stores the final population generated by RGA as POP_{RGA}^{T}(Line 31). It starts the reverse population handover process, or real-to-binary handover, since the computational power switches back to the BGA part. By comparing POP_{BGA}^{T} to POP_{RGA}^{T}, the handover framework has been developed to process the following selection criteria:

(1). The mean performance of RGA part is *better than* or *equal to* the mean performance of BGA part, i.e. logic in Line 32 is true. This scenario can be true when true when the RGA was successful in identifying an even better quality solutions from the regions suggested as "promising" by the BRGA part. So, the BRGA wants from the BGA to continue the search towards these regions. If this scenario happens, BRGA maximize the influx of reverse migration of data from the RGA part toward the BGA part (Line 33) and it directs the BGA part to replace all of its member solution in POP_{BGA}^{T} (Line 35) by POP_1 from the RGA part (Line 34).

(2). The mean performance of RGA part is *slightly lags* behind the mean performance of BGA part, i.e. the logic in Line 32 is false while the logic in Line 37 is true. This scenario can be true when the BRGA approaches the optimality regions. Here, the performance of both parts becomes comparable from each others. However, here, the RGA parts still fails in finding significantly better solutions from the regions suggested as "promising" by the BGA part and it shows signs of lagging. So, the BRGA wants from the BGA parts to retain some of its old population members and continue the exploration of the search space. If this scenario happens, BRGA decreases the influx of reverse migration from RGA part to the BGA part (Line 38) and directs the BGA part to replace only the worst quarter of its member's solutions (Line 40) by POP_1 from the RGA part (Line 39). By increasing the percentage of good members within POP_{BGA}^{T}, BRGA hopes that can enhance the ability of the BGA part to find even better promising regions within the search space.

(3). The mean performance of RGA part is *lags* behind

```
0. X_lo(1,1:n) = Min(POP_1);
1. X_hi(1,1:n) = Max(POP_1);
2. FOR i = 1: n
3.         IF X_lo(1,i) = X_hi(1,i) THEN
4.                 X_lo(1,i) = X_hi(1,i) - |ΩX_hi(1,i)|;
5.         END;
6.         FOR j = 1: Θ
7.                 X_1(j,i) = N(X_hi(1,i)-X_lo(1,i)) + X_lo(1,i);
8.         END;
9. END;
```

Fig. 3 Random sampling pseudocode.

the mean performance of BGA part, i.e. the logic in Lines 32 and 37 are false. This scenario can be frequently happened at the initial stages of the evolutionary search. Since, the BRGA is far from the optimality region there is a very low chance for RGA part to find good solutions within the regions suggested as "promising" by the BGA part. So, BRGA becomes less interested in initiating data exchange between the two parts since this process can degrade the quality of solutions within POP^{T}_{BGA}. If this situation happens, the BRGA stops the reveres migration from the RGA part toward the BGA part. By starting from the same un-updated POP^{T}_{BGA}, BRGA hopes to preserve the quality of solutions within POP^{T}_{BGA} from the influx of low quality members within POP^{T}_{RGA}.

4. Numerical Evaluation

In order to systematically evaluate the performance of BRGA, we have followed an experimental procedure that can be explained as follows.

4.1 Experimentation setup

The experimentation in EC suffers from the lacking of standardized benchmark problems. The available ones within the literature are usually based on small subset of standard test problems. The empirical results are generally confusing and limited such that the same algorithm working well for a set of functions may not working well for some other set of functions. Therefore, algorithms should be evaluated more systematically by determining a common termination criterion, size of problems, initialization scheme and running time. So, the special session on real-parameter optimization in CEC'2005 proposed a test suite of 25 benchmark functions, which represent a step toward achieving this objective [16]. These functions and the proposed guidelines were employed in our experiments. In addition, All the experiments were programmed in MATLAB 7 and ran in the same computer, Intel Celeron(R) 575 2GHZ, 3 GB RAM, Windows Vista (SP2). On the other hand, we have used Minitab 16 software was used for applying the required statistical techniques.

4.2 Parameter tuning

The aim of the experiments in this phase is to pick up the interactions between \algorithm performances against a particular benchmark function within the suite by identifying parameter configurations that maximize the performance of the algorithm for that particular function. In this study, we have adopted "experimental analysis of search heuristics", which is proposed by Bartz–Beielstein [17]. It is a useful approach which utilizes design of experiments and statistical testing techniques in investigating how sensitive performance of an algorithm is to parameter changes. In addition, it is more flexible and requires fewer experimental runs when compared with other approaches. Like Schaffer et. al. who proposed the use of a complete factorial design experiment to study control parameters of GA [18]. Or, Myers and Hancock who proposed different experimental framework based on factorial designs for the empirical modeling of GA [19]. In section 5, we describe the details of the parameter tuning procedure and discuss the obtained optimal parameter configurations.

4.3 Investigating the quality

In order to judge the quality of improvement achieved by our hybrid algorithm, comparison experiments were conducted against the original BGA and RGA (UNDX) over the CEC'2005 benchmark functions. The aim of the experiments is to analyze the effectiveness and efficiency performance of the BRGA. The conducted experiments can be explained as follows.

4.3.1 Effectiveness experiments

To evaluate our algorithm Effectivity, or the ability of the algorithm to achieve good performance over a wide range of test problems, the experiments in this phase were run for a fixed Number of Fitness Evaluation (MAX_FES = 1E+5) or up to achieve a fixed tolerance. Table 1 shows the descriptive statistics for the obtained errors from the experiments. The error is computed as (Error = f(x) –f(x*)), where f(x) is the best found solution by the algorithm, while the f(x*) is the already known global optimum for a particular benchmark function [16]. So, performance of the algorithm can be analyzed from the following stand points:

***a) Best solution found*:** It is obvious from the Table 1 that BRGA was able to locate solutions within the specified accuracy levels defined by CEC'2005 in 7 cases (F1-2, F4, F6-7, F12 and F15). The best found solutions are in the vicinity of required accuracy level in 6 cases (F5, F9-11, and F13-14). While the best found solutions for the remaining functions are located far from the global optimal solution.

b) Problem characteristics: The experimental results show that the performance of BRGA interacts differently with the underlying problem properties. Some of the properties are greatly affect the performance. For example, the benchmark contains functions with different condition numbers (F1-3). The using of high condition number, as the case of F3, greatly deteriorates algorithm performance by increasing its convergence time to the global optimum. The performance of the algorithm also deteriorates on the function which uses high condition number matrix (F22) compared with the function that uses orthogonal matrix (F21). Moreover, BRGA performs better with continuous function (F21) compared with non-continuous function (F23). On the other hand, BRGA shows a little sensitivity towards other properties. For example, the rotation of the benchmark function has a very small affect on the algorithm performance, since the mean performance is comparable for functions without rotation (F9 and F15) against functions with rotation (F10 and F16). In addition, locating the global optimum on bounds, like the case of F20 in comparison with F18, does not affect the

Table 1: Error values achieved for problems 1-25 (10D)

F	BRGA			UNDX			bGA			Tolearnce
	Best	Mean	Std	Best	Mean	Std	Best	Mean	Std	
1	5.73E-07	8.28E-07	1.39E-07	3.32E-07	7.92E-07	1.80E-07	1.15E+04	2.19E+04	4.59E+03	1.00E-06
2	7.57E-07	9.39E-07	6.67E-08	5.91E-07	8.75E-07	1.16E-07	1.09E+04	2.99E+04	1.06E+04	1.00E-06
3	6.24E+02	2.43E+04	2.12E+04	6.35E+03	5.33E+04	5.01E+04	1.40E+08	4.41E+08	2.67E+08	1.00E-06
4	4.14E-07	8.34E-07	1.42E-07	6.62E-07	8.26E-06	2.37E-05	1.58E+04	3.90E+04	1.21E+04	1.00E-06
5	3.41E-01	1.69E+00	7.76E-01	5.88E-01	2.62E+00	2.51E+00	1.46E+04	2.04E+04	3.27E+03	1.00E-06
6	7.51E-03	3.32E+01	7.22E+01	9.87E+00	1.00E+02	1.11E+02	4.54E+08	7.88E+09	3.53E+09	0.01
7	7.62E-03	6.13E-02	4.74E-02	3.63E-01	7.34E-01	1.87E-01	4.23E+02	8.65E+02	2.93E+02	0.01
8	2.00E+01	2.01E+01	1.41E-01	2.02E+01	2.04E+01	7.72E-02	2.03E+01	2.04E+01	5.46E-02	0.01
9	9.95E-01	6.09E+00	2.79E+00	1.17E+01	2.04E+01	3.60E+00	8.89E+01	1.10E+02	1.16E+01	0.01
10	9.95E-01	3.85E+00	1.66E+00	1.54E+01	2.35E+01	3.62E+00	1.28E+02	1.78E+02	1.85E+01	0.01
11	3.86E-02	6.67E-01	5.15E-01	6.13E+00	7.54E+00	6.93E-01	8.08E+00	9.12E+00	5.61E-01	0.01
12	4.94E-03	3.28E+02	6.51E+02	7.96E+03	2.38E+04	6.70E+03	3.24E+03	1.05E+04	4.34E+03	0.01
13	3.50E-01	8.21E-01	3.45E-01	1.47E+00	2.08E+00	2.47E-01	2.28E+01	1.19E+02	9.31E+01	0.01
14	8.83E-01	2.47E+00	6.07E-01	2.81E+00	3.34E+00	1.59E-01	4.34E+00	4.74E+00	1.24E-01	0.01
15	9.53E-03	2.56E+02	1.39E+02	2.48E+02	3.40E+02	4.09E+01	3.70E+02	6.05E+02	1.26E+02	0.01
16	7.22E+01	9.52E+01	6.29E+00	1.08E+02	1.32E+02	8.80E+00	3.03E+02	3.53E+02	2.21E+01	0.01
17	4.15E+01	9.59E+01	1.42E+01	1.26E+02	1.47E+02	9.43E+00	3.45E+02	3.89E+02	2.81E+01	0.1
18	3.00E+02	4.20E+02	2.18E+02	3.03E+02	3.74E+02	1.35E+02	1.05E+03	1.07E+03	9.05E+00	0.1
19	3.00E+02	4.00E+02	2.04E+02	3.03E+02	3.95E+02	1.08E+02	1.06E+03	1.07E+03	9.47E+00	0.1
20	3.00E+02	3.60E+02	1.66E+02	3.05E+02	4.30E+02	1.66E+02	1.06E+03	1.07E+03	5.23E+00	0.1
21	2.00E+02	5.32E+02	1.57E+02	5.00E+02	5.18E+02	6.21E+01	1.47E+03	1.54E+03	2.45E+01	0.1
22	3.00E+02	6.53E+02	2.02E+02	3.03E+02	5.49E+02	2.22E+02	1.08E+03	1.13E+03	3.32E+01	0.1
23	5.59E+02	6.09E+02	1.36E+02	5.59E+02	5.61E+02	5.37E+00	1.43E+03	1.54E+03	3.45E+01	0.1
24	2.00E+02	2.00E+02	0.00E+00	2.00E+02	2.00E+02	1.45E-02	1.40E+03	1.43E+03	2.74E+01	0.1
25	2.00E+02	2.00E+02	0.00E+00	2.00E+02	2.00E+02	2.03E-02	1.29E+03	1.43E+03	4.10E+01	0.1

performance of the algorithms. The same is true for the profile of the global optimum and initialization range. Locating the global optimum in a narrow basin (F19) compared with wide basin (F18) or locating the global optimum within the initialization range (F24) compared with locating it outside the initialization range (F25) has no affect on the algorithm performance. Finally, the same is true for the noise since the performance of BRGA for functions without noise (F2 and F16) is similar to the performance of BRGA with noisy functions (F4 and F17).

c) Comparison with BGA and RGA: To evaluate the effectiveness of the proposed hybrid scheme in improving the performance of its component algorithms, the original BGA and RGA were run on the same benchmark functions under similar experimental conditions with the recommended parameter configurations. The obtained results are reported in Table 1. When compared with BGA, it is clear that BRGA outperforms BGA in obtaining better quality solutions for all the functions. When compared with RGA, it is obvious that BRGA found better solutions for 19 functions (F3 and F5-F22). Both BRGA and RGA found the same best solutions for 6 functions (F1-2, F4 and F23-25). For these functions, both RGA and BRGA were able to identify solutions within the required accuracy levels for F1-2, while the performance is comparable for F23-F25. However, RGA has a better mean performance for F23, while BRGA has better robust performance for F24-25. So, the results show that that both the proposed hybrid scheme and the adopted parameter tuning procedure were successful in improving the performance against its component algorithms for CEC'2005 benchmark functions.

d) Comparison with other EAs: To evaluate the position of BRGA within the literature of EAs, we compare the performance of BRGA with some state-of-the-art EAs. Table 2 compares the obtained results from the BRGA experiments with some of EAs which has been testified against CEC'2005 benchmark suite under similar experimental conditions.

According to [20], steady-state Real parameter GA called SPC-PNX has been successfully applied to several nonlinear parameter estimation problems arising in Earth Sciences. When compared with BRGA, BRGA outperformed in 12 cases by identifying better best solutions for F3, F6-8, F10, F12, F14-17, F21 and F25. Both algorithms exhibit similar behavior in 10 cases when they achieved same level of accuracy for F1-2 and F4, while the performance is comparable in F9, F18-20 and F22-24. SPC-PNX was able to identify better best solutions only in 3 cases (F5, F11 and F13).

On the other hand, Estimation of Distribution Algorithms (EDAs) refer to a class of EAs which based on probabilistic modeling instead of classical genetic operators such as crossover or mutation. In [21] the authors used EDA_{mvg}, which employs a multivariate Gaussian distribution and is therefore able to represent correlation between variables in the selected individuals

Table 2: Error values achieved for problems 1-25 (10D)

F	BRGA		SPC-PNX[20]		EDA_{mvg}[21]		coEVO[22]		RCMA [23]	
	Best	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*	*Best*	*Mean*
1	5.73E-07	8.28E-07	6.49E-09	8.90E-09	0.00E+00	0.00E+00	4.60E-09	8.83E-09	9.87E-09	9.87E-09
2	7.57E-07	9.39E-07	8.74E-09	9.63E-09	0.00E+00	0.00E+00	5.00E-09	8.60E-09	9.94E-09	9.94E-09
3	6.24E+02	2.43E+04	7.75E+02	1.08E+05	0.00E+00	0.00E+00	5.67E-09	8.49E-09	7.21E+03	4.77E+04
4	4.14E-07	8.34E-07	7.69E-09	9.38E-09	0.00E+00	0.00E+00	6.13E-09	8.55E-09	8.77E-09	2.00E-08
5	3.41E-01	1.69E+00	7.94E-09	9.15E-09	1.97E-06	2.34E+02	9.17E-02	2.13E+00	2.96E-08	2.12E-02
6	7.51E-03	3.32E+01	1.86E-02	1.89E+01	0.00E+00	0.00E+00	1.27E+00	1.25E+01	4.06E-01	1.49E+00
7	7.62E-03	6.13E-02	9.86E-03	8.26E-02	3.94E-01	5.35E-01	1.02E-02	3.71E-02	2.22E-02	1.97E-01
8	2.00E+01	2.01E+01	2.08E+01	2.10E+01	2.02E+01	2.03E+01	2.01E+01	2.03E+01	2.01E+01	2.02E+01
9	9.95E-01	6.09E+00	9.95E-01	4.02E+00	2.34E+01	3.23E+01	8.99E+00	1.92E+01	8.10E-09	4.38E-01
10	9.95E-01	3.85E+00	1.99E+00	7.30E+00	2.47E+01	3.19E+01	1.50E+01	2.68E+01	3.04E+00	5.64E+00
11	3.86E-02	6.67E-01	3.00E-04	1.91E+00	3.75E+00	8.27E+00	6.78E+00	9.03E+00	1.14E-03	4.56E+00
12	4.94E-03	3.28E+02	3.91E+00	2.60E+02	0.00E+00	5.37E+01	2.66E+01	6.05E+02	1.18E-03	7.43E+01
13	3.50E-01	8.21E-01	3.49E-01	8.38E-01	1.59E+00	2.61E+00	4.69E-01	1.14E+00	3.81E-01	7.74E-01
14	8.83E-01	2.47E+00	1.39E+00	3.05E+00	2.92E+00	3.61E+00	3.26E+00	3.71E+00	6.84E-01	2.03E+00
15	9.53E-03	2.56E+02	6.32E+01	2.54E+02	3.34E+02	5.11E+02	1.37E+02	2.94E+02	5.69E+01	3.11E+02
16	7.22E+01	9.52E+01	9.11E+01	1.10E+02	1.32E+02	1.64E+02	1.23E+02	1.77E+02	8.96E+01	1.02E+02
17	4.15E+01	9.59E+01	9.89E+01	1.19E+02	1.50E+02	1.83E+02	1.45E+02	2.12E+02	1.04E+02	1.27E+02
18	3.00E+02	4.20E+02	3.00E+02	4.40E+02	3.00E+02	4.20E+02	8.00E+02	9.02E+02	8.00E+02	8.03E+02
19	3.00E+02	4.00E+02	3.00E+02	3.80E+02	3.00E+02	4.00E+02	5.00E+02	8.45E+02	3.00E+02	7.63E+02
20	3.00E+02	3.60E+02	3.00E+02	4.40E+02	3.00E+02	3.80E+02	5.00E+02	8.63E+02	8.00E+02	8.00E+02
21	2.00E+02	5.32E+02	3.00E+02	6.80E+02	5.00E+02	5.00E+02	2.00E+02	6.35E+02	5.00E+02	7.22E+02
22	3.00E+02	6.53E+02	3.00E+02	7.49E+02	7.66E+02	7.73E+02	3.00E+02	7.79E+02	3.00E+02	6.71E+02
23	5.59E+02	6.09E+02	5.59E+02	5.76E+02	5.59E+02	5.59E+02	4.25E+02	8.35E+02	5.59E+02	9.27E+02
24	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	2.00E+02	3.14E+02	2.00E+02	2.24E+02
25	2.00E+02	2.00E+02	4.06E+02	4.06E+02	3.64E+02	3.76E+02	2.00E+02	2.57E+02	2.02E+02	7.92E+00

via the full covariance matrix of the model. When compared with BRGA, BRGA outperformed in 13 cases by identifying better best solutions for F7-11, F13-17, F21-22 and F25. Both algorithms exhibit similar behavior in 8 cases when they achieved same level of accuracy for F1-2 and F4, while the performance is comparable in F18-20 and F23-24. However, EDA_{mvg} was able to identify better best solutions only in 4 cases (F3, F5-6 and F12).

Moreover, An evolutionary strategy with covariance matrix adaptation (CMA-ES) [22] becomes popular as an efficient tool for real parameter optimization. However, in [22] the Mutation Step Co-evolution algorithm (coEVO) does not use any covariance matrix, rather it co-evolves a population of successful mutation steps used in previous generations which substitute the probabilistic model (the variances or the covariance matrix) used in ES. The new population is then created by mutating the whole old population, thus allowing the algorithm to perform the search in several areas of the search space, and not only in one area covered by the Gaussian cloud (as in the case of CMA-ES). When compared with BRGA, BRGA outperformed in 15 cases by identifying better best solutions for F6-20 and F23. Both algorithms exhibit similar behavior in 7 cases when they achieved same level of accuracy for F1-2 and F4, while the performance is comparable in F21-22 and F24-25. However, coEVO was able to identify better best solutions only in 3 cases (F3, F5 and F23).

Finally, Memetic Algorithms (MAs) [23] refers to a class of EAs that apply a separate Local Search (LS) process to refine new individuals. In MAs, GAs are hybridized with LS with aim of addressing the trade-off between the exploration abilities of the GA, and the exploitation abilities of the LS used. One commonly used formulation of real coding MAs applies LS to members of the population after recombination and mutation, with the aim of exploiting the best search regions gathered during the global sampling done by the RGA. In [23], the authors used adaptive LS with a Steady-State Genetic Algorithm (SSGA) in their Real Coded Memetic Algorithm (RCMA). When compared with BRGA, BRGA outperformed in 13 cases by identifying better best solutions for F3, F6-8, F10, F13, F15-18, F20-21 and F25. Both algorithms exhibit similar behavior in 7 cases when they achieved same level of accuracy for F1-2 and F4, while the performance is comparable in F19 and F22-24. However, RCMA was able to identify better best solutions only in 5 cases (F5, F9, F11-12 and F14).

4.3.2 Efficiency experiments

The aim of the experiments in this phase is to evaluate algorithm efficiency, or how competent our algorithm is in using the available computational resources to come closer to the specified solutions targets. Therefore, the experiments in this phase were run until they reached a predefined error tolerance or a maximum number of fitness evaluation (1E+5). Since the fixed accuracy levels defined

by CEC'2005 are too strict, we defined a more flexible error tolerance levels derived from the best found solutions shown in Table 1. The fixed error tolerances in addition to the empirical results of this phase are shown in Table 3. The competition in this phase is limited to RGA, since it is the only competing algorithm which showed competitive results from the previous phase. Efficiency rates in addition to descriptive statistics of FES to reach the fixed tolerance are used as metrics to evaluate the efficiency of the competing algorithms. Efficiency rate is a quality-effort metric defined by Hillstrom [24]. It is used to measure how efficient an algorithm is in progress from the starting point using the available computational resources. It can be described as follows:

$$\text{Eff. rate} = \frac{\ln\left(|\bar{f}(0) - f^*| / |\bar{f}(x) - f^*|\right)}{\bar{T}} \quad (13)$$

Here, $\bar{f}(0)$ and $\bar{f}(x)$ are the mean of best initial and final solutions found by algorithm, f* is the known global optimum and $\bar{T}$ is mean time elapsed by the algorithm to reach the final solution. Smaller efficiency rates indicate higher efficiency in using the computational resources to reach high quality solutions.

It is clear from Table 3 that BRGA outperforms in term of efficiency rates in all the cases. It should be noted that BRGA gains better efficiency rates even for cases when it has a comparable quality performance to the UNDX, as the cases of F1-2, F4 and F23-25. This is an indication that the good performance for BRGA in term of efficiency rates is resulted not only form it ability in finding a better solutions but also from that fact that it has a better time complexity when compared with RGA.

5. Parameter Tuning Procedure

In [17], Bartz–Beielstein proposed a general heuristic procedure for parameter tuning that relies on techniques from statistical testing, design of experiments and Response Surface Methodology (RSM). The proposed framework consists from three steps of screening, modeling and optimization. The aim in the screening phase is to reduce the initial list of parameters by dropping out statistically insignificant parameters from the tuning procedure. Screening experiments were implemented using 2-levels Plackett-Burman design (P-B design) experiments [25], which are efficient in estimating the main effects of all factors at the same precision. Regression analysis in addition to graphical methods like Normal probability plot and the Pareto plot of the standardized effects were used to separate out the significant factors which needs further investigation in the subsequent steps. Then, a suitable local model (first-order or second-order) is usually built to approximate the shape of the response surface within the region of interest in the modeling step. This model is essential in guiding the search for optimal parameter values in the subsequent optimization step. In the case of first-order model, the direction of improvement was determined by implementing line search in the direction of the "path of the steepest descent", i.e. it is a minimization problem. If no further improvement along the path of the steepest descent was possible, we explored the area by

Table 3: The number of fitness evaluations to achieve the requested accuracy levels

F	relaxed tolerance	BRGA				UNDX			
		Min	*Mean*	*Std*	*Eff. Rate*	*Min*	*Mean*	*Std*	*Eff. Rate*
1	1.00E-02	4692	5958.48	724.08017	11.653693	16178	17547.92	819.13867	4.2658297
2	1.00E-02	8100	11491.28	1822.9161	4.8413387	32090	43700.24	3968.0541	1.6133544
3	6.24E+02	80013	99216.68	4000.7667	0.6080307	100010	100010	0	0.3828527
4	1.00E-02	29056	41490.88	6759.214	2.2796215	40202	53703.44	7536.5416	1.2605565
5	4.00E-01	85013	99415.88	3000.6	0.6693871	100010	100010	0	0.3736455
6	1.00E-02	75222	96508.44	7483.2521	1.1341429	100020	100020	0	0.783671
7	1.00E-02	69589	98830.6	6092	0.5052535	100020	100020	0	0.3562993
8	20	100002	100003.44	0.8205689	0.0034568	100020	100020	0	0.001073
9	1	87026	99510.8	2601	0.2111119	100020	100020	0	0.1061642
10	1	99556	100032.16	99.2	0.2098578	100020	100020	0	0.1101683
11	0.05	92286	99678.36	1548.8857	0.0667475	100020	100020	0	0.015342
12	0.01	56070	96523.68	11415.636	0.2988972	100020	100020	0	0.0781164
13	0.5	35452	93427.2	17126.969	0.2773496	100020	100020	0	0.1580582
14	1	90053	99654.28	2000.2676	0.0368567	100020	100020	0	0.0135691
15	0.02	95022	99823.92	1000.4	0.0147541	100020	100020	0	0.0041089
16	73	94013	99773.96	1200.2	0.0119735	100020	100020	0	0.006876
17	41.51	97009	99889.96	600.2	0.0204165	100020	100020	0	0.0068924
18	300.01	52553	78340.92	21319.181	0.0123079	100020	100020	0	0.0060959
19	300.01	38400	74839.36	27042.861	0.0139932	100020	100020	0	0.0053907
20	300.01	56432	75880.08	15268.06	0.0144898	100020	100020	0	0.0042801
21	200.01	90030	99633.84	2000.8	0.0080166	100020	100020	0	0.0038307
22	300.01	68194	95077.2	10537.892	0.0069542	100020	100020	0	0.001187
23	559.01	93050	99773.84	1400.8	0.0052568	97019	99899.96	600.2	0.0040857
24	200.01	22516	26127.04	2788.9475	0.0762959	94140	99391.2	1459.3204	0.0069285
25	200.01	22958	26773.68	1477.9093	0.0732316	97524	99920.16	499.2	0.0107656

fitting another local first-order model and obtain a new direction for the steepest descent. This step was repeated until the expected optimum area is found (if the response surface is unimodal). Here, the linear model is inadequate and shows significant lack-of-fit. Therefore, 2- levels central composite designs experiments were conducted to fit second-order regression models [26]. From this model, the optimal values for the parameters were deduced using graphical methods like response surface plots and contour plots. On the other hand, it is possible for the tuning procedure to be terminated by the first-order model when the rate of improvement becomes small or insignificant. Finally, it should be noted that the underlying assumptions of the obtained regression models in the above mentioned stages were verified using the appropriate graphical and formal statistical approaches. On the other hand, we have used Box-Cox power transformation to correct the obtained dataset for normality assumption [26].

We have identified an initial list of 12 parameters as shown in Table 4. We assume that these can influence the performance of BRGA. These parameters can be explained as follows.

1) ***BGA part parameters;*** BGA part has three parameters that can influence the performance of BRGA, which are selection pressure (S), mutation probability (P_m) and number of assigned bits per gene or variable (N_{bits}). Selection pressure (S) is inversely affects crossover rate (P_{cross}) as shown in Eq. (8). The crossover operator is considered by many researchers to be at least very important for GA. Different values have been found to be optimal parameter settings for P_{cross} as shown in Table 4. On the other hand, it has been suggested by many researchers [27] that a good performance for GA can be guaranteed by using mutation operator with low mutation probability. Table 4 shows different values recommended for mutation probability. Finally, the value of N_{bits} can be adjusted according to the problem domain of the specific problem at hand. However, identifying an optimal parameter value is important since it affects the quality-effort balance of the algorithm. By increasing N_{bits}, we can enhance the ability of the BGA part to find better resolution solutions (according to the Eq. (3)), however such process can also increase time overhead of the algorithm itself.

2) ***RGA part parameters;*** from the RGA part parameters, we selected σ_ξ and N_{cross} to be included in the parameter tuning procedure. According to [11], σ_ξ plays a role in preserving the statistical characteristics of the population distribution. The authors showed though theoretical analysis and numerical experiments that the optimal value for this parameter is 0.5 for UNDX algorithm. However, we would like to investigate how this value can be influenced by the synergetic effects between the RGA and BGA parts within BRGA. On the other hand, N_{cross} is that kind of parameters that affects both quality and time complexity of the algorithm. Increasing N_{cross} can enhance the ability of UNDX in finding better solution; however it also increases the time overhead of the algorithm. Different values have been used for N_{cross} within the literature as shown in Table 4.

3) ***Population size:*** it is one of the most important parameters that can affect the performance and time complexity of the GAs in general. According to analytical and experimental investigation done by Goldberg [28], selecting the suitable population sizing is essential to guarantee the success of genetic search in BGAs. However, different population size has been proposed as good settings for BGA as shown in Table 4. For UNDX, on the other hand, the author in [11] used different population sizing of 50 for unimodal functions and 300 for multimodal functions.

4) ***Hybrid scheme parameters:*** the hybrid scheme has 6 parameters that can affect the ability of the hybrid scheme in regulating the synergetic effects between the cooperative algorithms. Since these parameters are novel ones, it is necessary to include them into parameter tuning procedure to gain some insight about optimal parameters range and the significance of these parameters to BRGA performance. In the experiments of parameter tuning, we usually located the initial range for these parameters around the values shown in the Table 4.

Table 4: Initial list of the parameters

Parameter	Abbreviation	Type	Recommended values
BGA part parameters	S	Real	0.5 [10], (P_{cross} = 0.95) [27]
	P_m	Real	0.15[10], 0.01[27]
	N_{bits}	Integer	Depends on specific problems at hand.
RGA part parameters	Sigma (σ_ξ)	real	0.5 [11]
	N_{cross}	Integer	5 [11]
Population size	POP_{size}	Discrete	(300,50) [11], 30[10]
Hybrid scheme parameter	Lamda_i (Γ_i)	Integer	initial range located around 50
	Lamda_s (Γ_s)	Integer	initial range located around
	K_i	Integer	initial range located around
	K_s	Integer	initial range located around
	Omega (Φ)	Real	initial range located around 0.05
	Phi (Ω)	Real	initial range located around 0.3

So, Table 5 reports the obtained optimal parameter configurations by the tuning procedure for the benchmark functions. On the other hand, Table 6 summarizes the main outcomes the tuning procedure. It reports the cost of tuning in terms of FES and the shape of the response surface at the termination of tuning procedure. The most significant parameters refer to those which persist up to the termination of the tuning procedure. In case of termination at first-order model, only two of the most influential and statistically significant parameters were reported. Finally, the shape of response surface at optimal configurations in case of second order model is illustrated in Fig. 4 (A&B).

So, it is possible to argue that the performance of the BRGA is affected by the underlying problem characteristics. However, the parameter tuning procedure plays a vital role in picking up the interaction between the performances of BRGA against the specific problems at hand. By identifying the optimal values for control parameter, it was possible to secure a good performance for BRGA against a large spectrum of problems within the benchmark suite.

Moreover, the parameter tuning procedure gives us valuable insights into the behaviour of the BRGA. Taken Tables 5 and 6 into consideration, it can be briefly summarized as the following.

1) It is well known that conventional GAs performs poorly with small population sizes due to insufficient information processing. However, the results of parameter tuning procedure indicate that better performance for BRGA can be obtained at smaller population size. So, BRGA behavior from this point is similar to that of Micro-genetic algorithm.

2) Due to the synergetic affects, the RGA parts exhibits different behavior when with normal UNDX algorithm. In UNDX, the value of σ_ε plays a vital role in preserving the statistical characteristics of the new generations. So, it has been set to 0.5 [11]. However, the tuning procedure reveals that the RGA part acts as a population widening for some functions (the value of σ_ε greater than 0.5).

3) The results of the parameter tuning procedure confirm that the mutation-crossover scheme with relatively small mutation rate and moderate crossover rate is beneficial for the BGA part. However, there exceptions. The value of mutation rate is extremely small for F12, while the value of crossover rate is relatively large (or relatively small S values) for F1-2. On the other hand, the effect of resolution (or N_{bits}) was critical for the performance of BRGA especially for F3,F5-7,F9-11, F13-15 and F17 as it is clear from Table 6.

4) The results of the parameter tuning procedure are important in clarifying the region of interest for the hybrid scheme parameters, which are newly proposed ones. It is oblivious from Table 6 that the parameters which responsible on regulating the frequency of population handover (Ki and Ks) has been critical for the performance of BRGA in many cases. On the other hand, the effects of parameters which regulate the budget of RGA part (Γi and Γs) have been evident especially for F2-3, F7. In addition, the relatively high value for Φ associated with low value of Ω suggests that the RGA part is less stringent to receive members from the BGA part at initial stages of the search, while it becomes more stringent at the final stages of the

Table 5: Optimal parameters configurations

F	Hybrid Scheme Parameters						POP_{size}	BGA parameters			RGA Parameters	
	Γ_i	Γ_s	K_i	K_s	Φ	Ω		P_m	S	N_{bits}	N_{cross}	σ_ξ
1	40	44	46	30	0.06924	0.3	8	0.15	0.388394	13	15	0.9
2	50	44	43	49	0.05	0.3	4	0.15	0.308664	16	9	1.5
3	45	47	85	48	0.13869	0.195744	4	0.145472	0.432845	17	16	1.05
4	30	30	30	30	0.05	0.3	20	0.15	0.5	10	30	0.5
5	30	32	34	30	0.05	0.3	20	0.15	0.5	10	33	0.6
6	30	37	43	35	0.068227	0.222166	20	0.15	0.5	16	12	0.866162
7	30	22	61	74	0.05	0.137252	20	0.31498	0.5	19	30	0.689394
8	30	30	36	38	0.05	0.3	4	0.126245	0.443884	10	23	0.5
9	30	30	40	59	0.000347	0.3	20	0.15	0.5	5	30	0.5
10	30	30	30	16	0.05	0.3	44	0.15	0.404154	4	29	0.491662
11	30	30	12	30	0.05	0.3	36	0.15	0.5	4	22	0.5
12	27	30	30	84	0.043126	0.3	4	0.023659	0.426096	10	30	0.932241
13	30	30	84	30	0.05	0.3	20	0.15	0.5	12	30	0.430303
14	30	51	50	13	0.082524	0.3	16	0.15	0.5	14	30	0.5
15	30	35	30	34	0.05	0.3	52	0.130237	0.5	9	30	0.5
16	30	30	26	30	0.05	0.3	44	0.15	0.5	3	18	0.5
17	30	30	22	30	0.5	0.3	32	0.15	0.5	5	33	0.468097
18	30	30	55	30	0.05	0.3	40	0.07401	0.5	8	30	1
19	30	30	60	16	0.05	0.3	24	0.15	0.5	6	44	1
20	30	30	30	30	0.05	0.3	36	0.15	0.5	6	32	1.15985
21	30	30	30	26	0.05	0.3	52	0.15	0.5	6	18	0.774659
22	30	58	70	30	0.05	0.3	52	0.15	0.5	8	30	0.723715
23	30	30	30	30	0.105266	0.3	56	0.15	0.5	6	30	0.790051
24	33	30	34	30	0.05	0.3	32	0.15	0.5	6	30	0.54146
25	30	32	34	32	0.05	0.3	32	0.15	0.5	6	30	0.571519

Table 6: Outcomes of parameter tuning procedure

F	Most significant parameters	Response surface at Termination	Cost (FES)
1	N_{bits}, σ_ξ	first-order	4.00E+06
2	σ_ξ , N_{cross}, Γi,Ki	second-order	1.27E+07
3	σ_ξ , N_{cross}, Γs,Φ, N_{bits}	second-order	1.27E+07
4	P_m, POP_{size}, σ_ξ, Ki	second-order	6.98E+06
5	σ_ξ , N_{cross}, POP_{size}, N_{bits},Ki	second-order	7.10E+06
6	$\sigma\xi$,Ω, N_{cross}, N_{bits}	second-order	6.05E+06
7	σ_ξ ,N_{bits}, P_m, Γs, Ks	second-order	2.00E+07
8	POP_{size}, Ks	first-order	4.10E+06
9	N_{bits}, Ks, Ki, Φ	second-order	1.43E+07
10	POP_{size}, N_{bits}, Ks	second-order	2.55E+07
11	POP_{size}, N_{cross}, N_{bits}, Ki	second-order	1.43E+07
12	σ_ξ , P_m, Ks	second-order	1.11E+07
13	σ_ξ , N_{bits}, Ki	second-order	1.20E+07
14	POP_{size}, N_{bits}, Γs	first-order	1.26E+07
15	POP_{size}, N_{bits}	first-order	2.25E+07
16	POP_{size}, N_{cross}, Ki	second-order	1.46E+07
17	σ_ξ , POP_{size}, N_{cross}, N_{bits}	second-order	1.74E+07
18	σ_ξ ,Ki	first-order	1.48E+07
19	σ_ξ ,N_{cross}	second-order	1.39E+07
20	σ_ξ ,N_{cross}, POP_{size}	second-order	1.60E+07
21	σ_ξ ,POP_{size}	second-order	4.23E+08
22	σ_ξ ,POP_{size}	second-order	1.29E+07
23	σ_ξ ,POP_{size}	first-order	1.03E+07
24	σ_ξ ,POP_{size}	first-order	1.18E+07
25	σ_ξ ,POP_{size}	first-order	1.18E+07

evolutionary search. This relation has been critical for the performance of BRGA especially for F3, F6 and F9.

Finally, it is evident that the parameter tuning procedure is computationally an expensive one. However, the payoff of this cost is the optimal control values and the insight into the behavior of the algorithm. BRGA is a stochastic approach and hybrid in nature. So, it is extremely difficult to use the formal methods to analyze the behavior of the BRGA or to predict the optimal values for the control parameters. Taking the limitation of the formal methods into consideration, it is even more difficult to predict the changes in the behavior of algorithm in response to the changes in the underlying problem characteristics.

6. Conclusion and Future Work

BRGA is a recent hybrid approach that relies on a parameterized hybrid scheme to share the computational power and control the interactions between two cooperative versions of GAs (BGA and RGA). In this article, we conducted experimental investigation to evaluate the performance of BRGA using CEC'2005 benchmark function. The experimental results showed that BRGA succeeded in locating the global optimum or approaching the vicinity of it for 13 problems. The most challenging problems for performance of BRGA are those which uses high condition number, high condition number matrix and non-continues problems. On the other hand, BRGA shows a little sensitivity toward characteristics like noise, rotation, locating global optimum on bounds or outside initialization regions or locating it in challenging profiles. When compared with BGA, BRGA outperforms in all the cases. When compared with RGA, BRGA showed better quality and time performance for majority of the cases. The results are an indication that the adopted hybrid scheme was successful improving the performance of BGA and RGA components against the benchmark functions. Moreover, the experimental results showed that BRGA has a superior performance when compared with other state-of-the-art EAs that had been testified on the same benchmark suite under similar experimental conditions. In addition, the outcomes of parameter tuning procedure confirmed that BRGA is efficient in using memory since it uses small population sizes. Finally, the implemented parameter tuning procedure was computationally expensive. However, it was effective in identifying optimal parameter settings which secures a good performance for BRGA over a broad spectrum of the benchmark functions. Most importantly, it sheds some lights on the behavior and the internal mechanisms of BRGA.

Important points within or current research agenda includes extending the algorithm to the constrained optimization problems by incorporating appropriate constraints handling techniques and applying it to complex real-world problems like the resource allocation problem in cloud platforms, which is a part of our research agenda [29].Another possible directions for future work includes investigating the effectiveness of the hybrid scheme when the BGA part is implemented by more advanced BGAs versions and reducing the number of hybrid scheme parameters by fixing some of them or by introducing more innovative adaptive schemes.

Acknowledgments

We would like to express my sincere gratitude to the anonymous reviewers for their efforts and comments, which help considerably in improving the quality of this paper. Moreover, this work is supported by KAKENHI (No.22500196), Japan Society for the Promotion of Science.

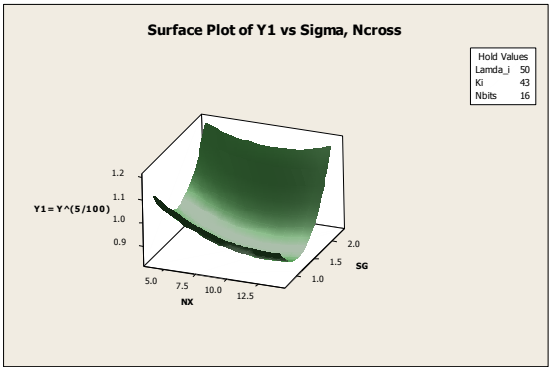
References

[1] O. Abdul-Rahman, M. Munetomo and K. Akama, "An Adaptive Resolution Hybrid Binary-Real Coded Genetic Algorithm", Proc.of the 16th International Symposium on Artificial Life and Robotics (AROB 16th ' 11), Jan. 2011,pp. 359-362.Japan.

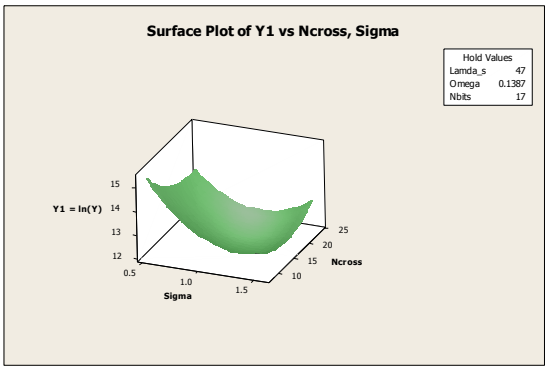
[2] O. Abdul-Rahman, M. Munetomo and K. Akama, "An adaptive resolution hybrid binary-real coded genetic algorithm", Journal of Artificial Life and Robotics, Springer, Vol. 16(1), 2011, pp. 121-124.

[3] K. Krishnakumar, R. Swaminathan, S. Garg and S. Narayanaswamy, "Solving large parameter optimization on problems using genetic algorithms", Proc. of the Guidance, Navigation, and Control Conference, 1995, pp.449-460.

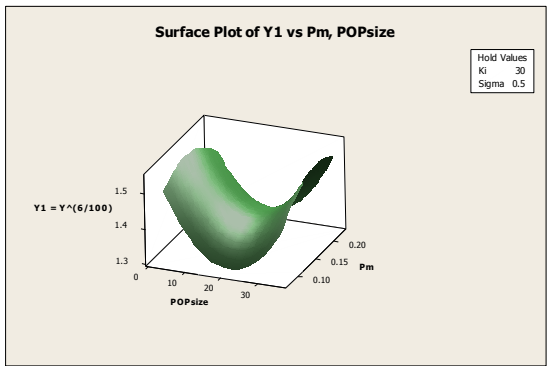
[4] M. Arakawa and I. Hagiwara, "Development of adaptive real range (ARRange) genetic algorithms", JSME Intl. J., Series C, Vol. 41(4), 1998, pp. 969-977.



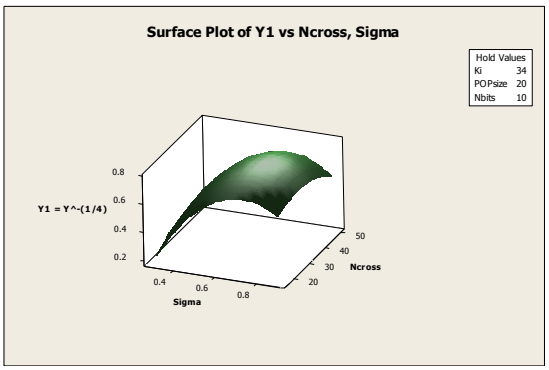
F2



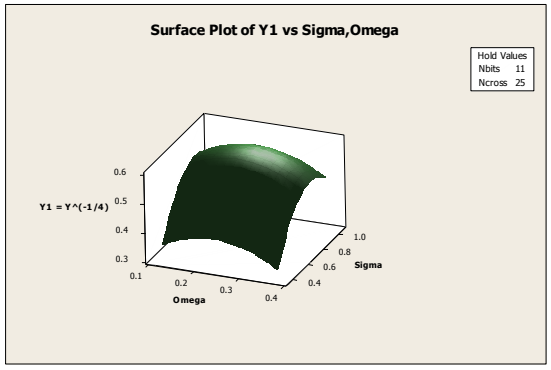
F3



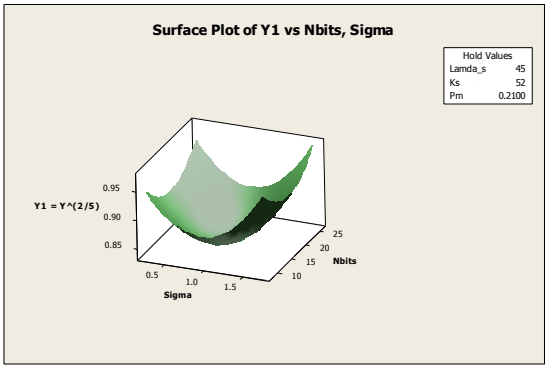
F4



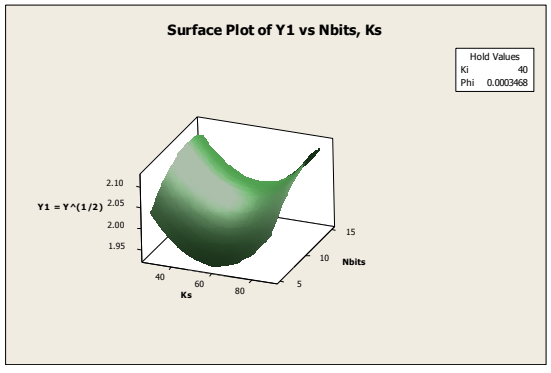
F5



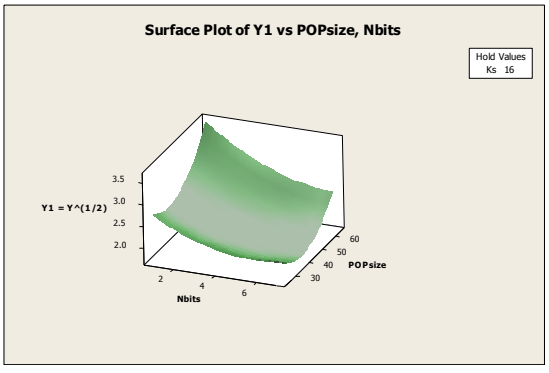
F6



F7



F9



F10

Fig. 4-A Second-order performance empirical models at optimal parameters configurations.

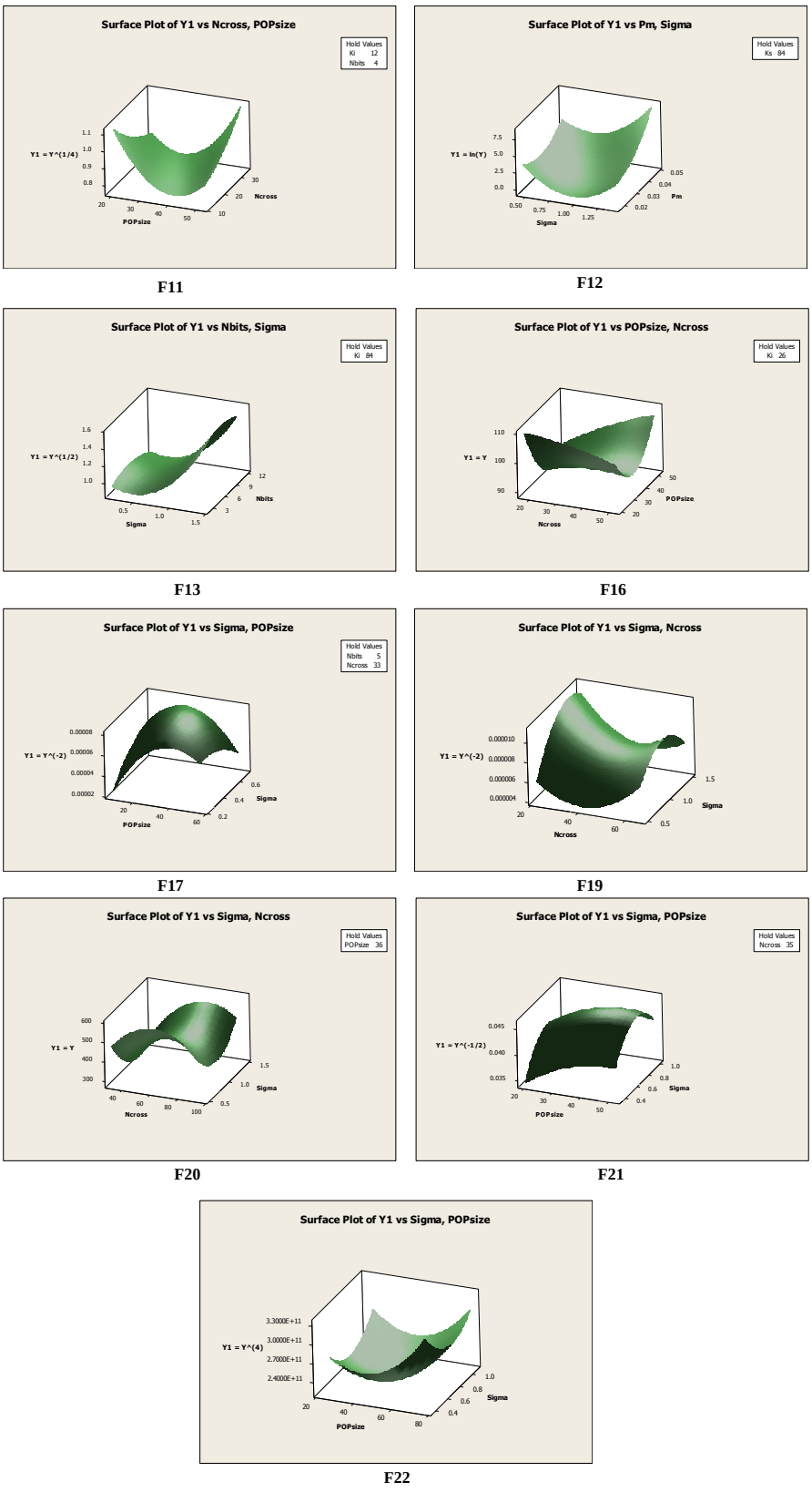


Fig. 4-B Second-order performance empirical models at optimal parameters configurations.

[5] Munawar, M.Wahib, M. Munetomo and K. Akama, "Solving extremely difficult MINLP problems using Adaptive Resolution micro-GA with Tabu Search", Proc. of the Learning and Intelligent Optimization (LION5), Lecture Notes in Computer Science 6683, Springer (2011).
[6] P Preux and E G Talbi, "Towards hybrid evolutionary algorithms", International Transactions in Operational Research, Vol. 6(6), 1999, pp. 557-570.
[7] D. Barrios, A. Carrascal, D. Manrique, J. Ríos, "Cooperative binary real coded genetic algorithms for generating and adapting artificial neural networks", Neural Computing and Applications, Vol. 12, 2003, pp. 49-60.
[8] S. Achiche, L. Baron and M. Balazinski, Real/binary-like coded genetic algorithm to automatically generate fuzzy knowledge bases, Proc. of 4th International Conference on Control and Automation, ICCA '03, 2003, pp. 799 – 803.
[9] D. E. Goldberg, The Design of Innovation: Lessons from and for Competent Genetic Algorithm, Springer: Kluwer Academic Publishers, 2002.
[10] R. Haupt, and S. Haupt, Practical Genetic Algorithms, NewYork; Wiley-Interscience, 1998.
[11] I. Ono and S. Kobayashi, "A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover", Proc. of the 7th ICGA, 1997, pp 246–253.
[12] H. Satoh, M. Yamamura, and S. Kobayashi, "Minimal generation gap model for gas considering both exploration and exploitation", Proc. of the IIZUKA: Methodologies for the Conception, Design and Application of Intelligent Systems, 1996, pp. 494–497.
[13] L.J. Eshelman and J.D. Schaffer, "Real-Coded Genetic Algorithms and Interval-Schemata, Foundations of Genetic Algorithms 2", 1993, pp. 187-202.
[14] F. Herrera, F. Herrera and E. Herrera-viedma, E. Herrera-viedma, M. Lozano, M. Lozano, J. L. Verdegay and J. L. Verdega, "Fuzzy tools to improve genetic algorithms", Proc. of the Second European Congress on Intelligent Techniques and Soft Computing, 1994, pp. 1532- 1539.
[15] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space", Complex Systems, Vol. 9, 1995, pp. 115–148.
[16] P. N. Suganthan1, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger et. al., "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 #2005005, IIT Kanpur, India, http://www.ntu.edu.sg/home/EPNSugan/,Last access on 2011.08.21.
[17] T. Bartz-Beielstein, "Experimental Research in Evolutionary Computation: The New Experimentalism", Natural Computing Series, Springer, 2006.
[18] J. D. Schaffer, R. A. Caruana, L. Eshelman, and R. Das. "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization", In J. D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, San Mateo, CA, 1989. Morgan Kaufman.
[19] R. Myers and E.R. Hancock, "Empirical Modelling of Genetic Algorithms". Evolutionary Computation, Vol. 9(4), 2001, pp.461–493.
[20] P.J. Ballester, J. Stephenson, J.N. Carter and K. Gallagher, "Real-Parameter Optimization Performance Study on the CEC-2005 benchmark with SPC-PNX", Proc. of IEEE Congress on Evolutionary Computation (CEC'2005), Sept. 2005, pp. 498 – 505, Vol.1, Edinburgh, Scotland.
[21] B. Yuan, and M. Gallagher, "Experimental Results for the Special Session on Real-Parameter Optimization at CEC 2005: A Simple, Continuous EDA", Proc. of IEEE Congress on Evolutionary Computation (CEC'2005), Sept. 2005, pp. 1792 - 1799, Vol.2, Edinburgh, Scotland.
[22] P. Posik, "Real Parameter Optimization Using Mutation Step Co-evolution", Proc. of IEEE Congress on Evolutionary Computation (CEC'2005), Sept. 2005, pp. 872 – 879, Vol.1, Edinburgh, Scotland.
[23] D. Molina, F. Herrera, and M. Lozano, "Adaptive Local Search Parameters for Real-Coded Memetic Algorithms", Proc. of IEEE Congress on Evolutionary Computation (CEC'2005), Sept. 2005, pp. 888 - 895, Vol.1, Edinburgh, Scotland.
[24] K. E. Hillstrom, "A Simulation Test Approach to The Evaluation of Nonlinearoptimization Algorithms", ACM Transactions on Mathematical Software, Vol. 3(4), 1977,pp. 305– 315.
[25] J. Antony, Design of Experiments for Engineers and Scientists, Butterworth-Heinemann:Elsevier Science & Technology Books, October 2003.
[26] D. C. Montgomery, Design and Analysis of Experiments, USA: Wiley, 7th ed., 2008.
[27] R. Salomon, "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms", Journal of BioSystems,Vol. 39, 1995, pp. 263-278.
[28] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Professional, January 1989.
[29] O. Abdul-Rahman, M. Munetomo and K. Akama, "Multi-level autonomic architecture for the management of virtualized application environments in cloud platforms", Proc. of 2011 IEEE International Conference on Cloud Computing, CLOUD 2011, Washington, USA, 2011, pp. 754-755.

Omar Abdul-Rahman received the B.Sc. and the M.Sc. from the Department of Electrical & Electronic Engineering, University of Technology, Baghdad, Iraq in 2004 and 2006 respectively. Currently, he is pursuing PhD degree in Information Technology, Hokkaido University under Japanese government scholarship (MEXT). His current research interests include genetic algorithms, evolutionary computation, multiobjective optimization and cloud computing. .

Masaharu Munetomo received the PhD in information engineering from graduate school of engineering, Hokkaido University in 1996. From 1998 to 1999, he joined Illinois Genetic Algorithms Laboratory (IlliGAL), university of Illinois at Urbana-Champaign as a visiting scholar. Since 1999, he works for Hokkaido University as an associate professor. He also engaged in designing Hokkaido university academic cloud at information initiative center of the university.

Kiyoshi Akama received the PhD in control engineering from Tokyo Institute of Technology in 1989. Now he works as a professor at Information Systems Design Laboratory, Division of Large Scale Computing Systems, Information Initiative Center, Hokkaido University, Japan. His research interests include automatic program construction, logical problem solving, equivalent transformation computation model, and artificial intelligence.