Comparison of Real-Coded Genetic Algorithm and Real-Coded Biogeography-Based Optimization, Global Unconstrained Optimization Algorithms

Ayodele P Boglo
Carrtelcom Nigeria Company Limited
Lagos, 100216, NIGERIA
ayodele.boglo@carrtelcom.com,

Barend W Botha
Department of Engineering Management, Postgraduate School of Engineering Management,
University of Johannesburg
Aucklandpark, South Africa
bwbotha@uj.ac.za

Babatunde A Sawyerr
Department of Computer Science, University of Lagos,
Akoka, Lagos, Nigeria
bsawyerr@unilag.edu.ng

Abstract

This research focuses on the performance comparison of real-coded Genetic Algorithms (GA) and Biogeography-based Optimization (BBO) algorithms. Specifically, it takes three modified versions of the original algorithms made suitable for continuous operations (real-coded) – the standard real-coded GA (SRCGA), the real-coded GA with mathematical projection (RCGA-P) and the real coded-biogeography-based optimization algorithm (RCBBO); then conducts a performance-based comparison of these three with respect to convergence, speed and robustness criteria. This comparison was done using 52 standard optimization benchmark problems over four, ten and twenty dimensions. Results show that overall, SRCGA outperforms the others for speed while RCGA-P was the most robust of the three. The major value-add in performance of RCBBO was mainly from speed as in two dimensions, it delivers better speed than the two real-coded GA

Keywords
Optimization, bio-inspired, algorithm, performance

Overview

1.1 Introduction

Historically, differential calculus was used to find minima or maxima of optimization problems, from the introduction of Gantt Charts in 1900, to its use in determining economic order quantity by Harris in 1915; from the queuing analysis of Erlang to its more recent uses in Network flow, knapsack, facility location and layout, logistics and transportation, production planning and scheduling problems (Sarker, Newton 2007). In all these cases, optimization problems have shown increasing levels of complexity.

More broadly, the complexity of optimization problems can be classified to include those that are solvable (where there is an effective procedure for solving the problem) or unsolvable (where finding an algorithm that exactly solves the problem non-deterministic in time is futile). (Adewumi 2016) classified methods for solving optimization problem across the various complexity classes as exact (I.e. solvable), and either Heuristics/ Metaheuristics, Artificial Intelligent and Hybrid for the unsolvable.

Bio-Inspired Computing can be said to be the use of computing inspired from the field of biology by studying social behaviour of animals, insects and other living organisms such that, with the use of the computer, nature can be modeled and the same be used to solve optimization problems (Pintea 2014). Bio-Inspired computing takes the Meuristics/ Metaheuristics method as they lend themselves to problems that are unsolvable and the focus is more exploring possibilities and exploiting the same.

(Binitha, Sathya 2012)organised bio-inspired computing into various approaches broadly based on evolution, swarm and ecology. (Xing, Gao 2014) updated the list to include 99 bio-inspired approaches in total.

1.2 Background

As with problems in optimization, representation of the various options is at the core of a solution. (Floreano, Mattiussi 2008) mentioned that representations of possibilities could be: (a) discrete – as a sequence of L values from an alphabet with cardinality k. Therefore, binary representation would mean any sequence of values based on the alphabet 0 and 1 with a cardinality of k=2; (b) real-coded or real valued representation refers to possibilities represented as a set of n numbers based on the domain of real numbers; (c)tree-based or hierarchical representations which is composed of a finite set of functions and a finite set of terminals.

Real-coded representations are the background of this research, because they offer the following advantages as mentioned by (Herrera, Lozano et al. 1998): (a) ability to use large domains for the variables, (b) the capacity to exploit how slight changes in the variables correspond to slight changes in functions (i.e. graduality), (c) improved tuning capability from this graduality and (d) more closely mimicks nature.

1.3 Problem Statement

The original algorithms of GA and BBO were based in discrete characteristics and as such may not be truly representative of the real-world environment they attempt to optimize as these problems tend to be in the real-valued domain. This study focused on the performance comparison of three real-coded versions of GA and BBO algorithms used in the optimization problems. The comparison includes tests for convergence, robustness and speed.

2. Literature Review

2.1 Optimization

Optimization concept revolves around finding the best possible solution to a given model or problem (Sarker, Newton 2007). The decision about the best solution involves iteration over a set of alternatives with the best having the highest evaluated value when compared to other alternatives. As such optimization algorithms, ‘are search methods, where the goal is to find a solution to an optimization problem, such that a given quantity is optimized, possibly subject to a set of constraints.’ (Engelbrecht 2007). The best or optimal solution is more specifically finding an alternative that provides the maximal or minimal value beyond others which the problem owner seeks.

(Engelbrecht 2007) identified the components that optimization problems possess to generally be (a) *Objective Function* - This is the quantity or measure that is to be optimized, (b) *Set of Variables:* These are the unknown that determine the value of the objective function and (c) *Set of Constraints:* These restrict the values that can be assigned to the variables or unknowns.

Mathematically, given a function f such that $f : S \rightarrow R$ where $S \subset R$ find $x^* \in S$ such that:

$f(x^*) \leq f(x) \forall x \in S$ ………..global minimization function

If the variable x is a composite variable, then that makes the optimization a multivariate problem. The process for solving an optimization problem is made up of six steps (Saker & Newton, 2008) namely (a) identifying and clarifying the problem, (b) defining the problem, (c) formulating and constructing a mathematical model, (d) obtaining a solution to the model/ picking an optimization algorithm, (e) testing the model, evaluating the solution, and carrying out sensitivity analysis and (f) Implementing and maintaining the solution. Sections 2.2 and 2.3 discuss the GA and BBO and how they are applied to the optimization process.

2.2 Genetic Algorithm

GAs are part of evolutionary computing; evolutionary Computation refers to a class of algorithms that utilize simulated evolution to some degree as a means to solve a variety of problems, from numerical optimization to symbolic logic (Floreano, Mattiussi 2008). By simulated evolution, we mean that the algorithms have the ability to evolve(i.e. change) a population of potential solutions such that weaker solutions are removed and replaced with incrementally stronger (better) solutions.

As with optimization problems the major goal is to find / explore for the minima (global or local). All living organisms consist of cells, where each cell contains a set of chromosomes (strings of DNA). These chromosomes serve as the basis for genetic algorithms, where a potential solution is defined as a chromosome, and the individual elements of the solution are the genes (Jones 2008). To reach the global minima, adaptation of the major genetic operators are performed.

In encoding, the problem is represented as chromosome with each gene represented as zero and ones (Holland John 1975). (Davis 1991) invented other encoding techniques, with (Floreano, Mattiussi 2008) broadly classifying them as discrete, real-valued and tree-based representations. Encoding is the result of the identifying and defining stages of the optimization problem.

In evaluation, the main objective is to determine the suitability of each chromosome based on the objective function being optimized. An evaluation function in GAs plays the same role the environment plays in natural evolution (Negnevitsky 2005). The objective function returns a value called fitness. The solution with the best fitness based on the termination criteria is the optimal value of the optimization problem. Obtaining the final solution involves going through the evolutionary steps of selection and recombination iteratively until the termination conditions are met.

In selection, as with natural selection, based on the fitness candidate members of the population with the optimal fitness value who would be chosen to contribute towards the next generation of solutions. (Jebari, Madiafi 2013) designed a technique that reviews the quality of the solutions from six popular selection methods namely roulette wheel, linearly-ranked selection, exponentially ranked selection, stochastic universal sampling, tournament selection and truncation selection.

Recombination is responsible for generating the next generation of solutions that are closer to the required minima. It is further broken down into: (a) Crossover of genes takes two candidate solutions (chromosomes) and divides them, swapping components to produce two new candidates. The division could be from a single point onwards or from multiple points. Once the point has / points have been selected, a type of crossover operation is undertaken such as arithmetic, geometric, LaPlace, double Pareto, simulated binary crossover (Emami, Mozaffari et al. 2016) or (b) mutation where a single candidate has some as some aspect of it chromosome changed.

(Sawyerr, Ali et al. 2011) developed two variants of the GA based on real value representation namely (a) Standard Real-Coded Genetic Algorithm (SRCGA) and (b) Real-Coded Genetic Algorithm with mathematical orthogonal Projection (RCGA-P). (Sawyerr, Ali et al. 2011) laid out a description of the SRCGA and RCGA-P used in this research. It first starts with the encoding operator; where all individual solutions have their genes encoded as floating-point representations. In the subsequent notation, t refers to the generation or more specifically the current iteration of solutions upon which one would like to find the optimal and I refers to the ith element in the set of solutions

For selection and evaluation, the linear ranked selection (LRS) method was chosen. At a generation *t*, LRS is made up of two parts, i.e. (a) determine the expected value of each solution, $a_{i,}$ at the current generation EV(X_i) and (b) converting this expected value to discreet number of solutions. (Sawyerr, Ali et al. 2011)further gave, that since EV(a_i) at *t* would be a real value, stochastic universal sampling (SUS) would be applied to get the discreet number of offspring that each solution a_i should have such that the total population sum of expected values should be equal to the count of the population N. The aim of the selection is to generate a mating pool of solutions *m* such that for population P:

$$P_t = \{a_1, a_2, \ldots, a_m\} \quad where\ m \leq N$$ (Sawyerr, Ali et al. 2011)

Specifically, linear selection is implemented by (a) evaluating the fitness of each solution through the result of some objective function calculation (b)sorting in descending order for global maximization or ascending order for minimization. (c) each solution is ranked after sorting with the first given a rank of 1 and others given in sequence thereafter ending with N for the last solution, in the population(Sawyerr, Ali et al. 2011). The rank of a_i is used to calculate EV(a_i) at each generation *t* which is given as:

$$EV(x_i) = D - \frac{2 \times (D - 1.0) \times (i-1)}{N-1}, 1.0 \leq D \leq 2.0$$ (Sawyerr, Ali et al. 2011)

For crossover, by (Sawyerr, Ali et al. 2011)a single point is applied pairwise based on a random probability P_c to all member solutions that made it into P_t . Then, the arithmetic crossover operation is carried out on X_i and X_{i+1}. as follows:

$$b_i^j = \alpha^j x_i^j + (1-\alpha^j) x_{i+1}^j$$
$$b_{i+1}^j = \alpha^j x_{i+1}^j + (1-\alpha^j) x_i^j$$

where α_j is uniform over the interval [−0.5,1.5] where, j = 1, 2, . . ., Ω (Ω, being the dimension of the chromosome). The new population called Z_t refers to the new population just before mutation with:

$$Z_t = \{b_1, b_2, \ldots, b_m\}$$

For mutation, by (Sawyerr, Ali et al. 2011) based on a mutation probability P_m each b^j_i of b_i element of Z_t is carried out:

$$c_i^j = b_i^j + \beta^j (\max{}^j - \min{}^j)$$

where β^j is approximately uniform over the interval [−0.01, 0.01] where, j = 1, 2, . . ., n, $\max^j$ and $\min^j$ are the maximum and minimum bounds of a ∈ in search space S, respectively. The new population, M_t refers to the candidate solutions that would comprise the next generation P_{t+1}.

$$M_t = \{c_1, c_2, \ldots, c_m\}$$
$$where$$
$$c_i^j = \begin{cases} b_i^j + \beta^j (u^j - a_i^j)\ if\ b_i^j\ is\ mutated \\ b_i^j \quad otherwise \end{cases}$$

In the case of RCGA-P, (Sawyerr, Ali et al. 2011) provided an extra process called projection to give a new Population, Q_t = {s_1, s_2,…,s_m}; "the projection-based exploration search method is based on the concept of orthogonal projection of vector x on vector y. Vector projection was used to enhance the exploration of points in the search space by using two solution points to locate a better point in the solution landscape. It is a two-parent operator that produces only one offspring. The principle is as follows: Suppose two solutions g and h are randomly selected from P_t and are evaluated, if f (g) is better than f (h) then we project h on g, otherwise we project g on h. For any two n dimensional vectors, the projection of g on h generates a vector ˆh defined by" (Sawyerr, Ali et al. 2011):

$$\wedge h = \frac{g^T h}{h^T h} h = \frac{g^T h}{\|h\|^2} h = \left(\frac{\|g\| \cos(\theta)}{h} h \right)$$

Lastly elitism is applied to ensure only elite solution finally make it into the next generation t+1 of population P. The above process is done iteratively until the desired minima or termination condition is reached.

2.3 Biogeography-Based Optimization

(Xing, Gao 2014) defined biogeography as a field of physical geography concerning the study of the distribution of species around the world over time. (Simon 2013) defined it as, 'the study of the speciation, extinction and geographical distribution of biological species'. (Simon 2008) algorithm was based on the work of (Wilson, MacArthur 1967) that showed that using mathematical models, the pattern of species richness of an area can be explained through a combination of historical factors such as habitat area, the rate at which species arrive or depart a habitat rate and the rate species were lost or extinct in a habitat.

At the heart of the science of biogeography is the island. (Simon 2008) preferred to use the term habitat for the biogeography-based optimization algorithm (BBO). (Gilpin, Hanski 1997) surmised that an island, or habitat, can be the total of factors in a geographical area that can support life for animal species. Examples of this would include log of wood that supports mushrooms, pond for fish and amphibians. These can be said to be the constructs or variables fundamental to the habitat. Their value provides an idea of suitability for species in a habitat. (Simon 2008) called these as independent variables or more appropriately called suitability index variables (SIVs). The value of these SIVs collectively against other physical characteristics of rainfall, vegetation cover, temperature help to determine whether an animal species can survive in the habitat. (Simon 2008) called these physical attributes collectively habitat suitability index (HSI). Mathematically, both the SIV and HIS could be represented as vectors and in the case of real-coded BBO each cell contains a real number which is a measurement of the value of the said characteristic. (Simon 2008) observed some correlation between HSI and species count, with high HIS favouring large species count and low HSI having small species count. (Simon 2008) also explained such arrangement had different dynamics with regards immigration and extinction. The higher a HSI of a habitat the higher the emigration and lower immigration of species; lower HSI tends to lead to higher species immigration. If HSI remains low consistently over time, then the likelihood of extinction of species within the habitat increases paving way for future immigration.

As with optimization problems, the major goal is to find/explore for the global minima. From an optimization perspective, (Simon 2008) believes a higher HSI means a better solution, while a poorer solution would be indicated by a lower HSI. Thus, the process of optimization is therefore to repeatedly use the operations of migration (i.e. immigration and emigration) and mutation to ensure the highest possible HSI.

In migration the information is shared between habitats that depend on emigration rates μ and immigration rates λ of each solution. The immigration rate λ and the emigration rate μ are dependent on species found in the habitat. The equilibrium number of species is given as S_0 where $\lambda_s = \mu_s$ for that species. (Simon 2008) gives the mathematical formulae as follows:

$$\lambda_s = I\left(1-\left(\frac{S}{S_{\max}}\right)\right) \qquad for\ 0 \le S \le S_{\max} \qquad\qquad \mu_s = E\left(\frac{S}{S_{\max}}\right) \qquad for\ 0 \le S \le S_{\max}$$

Where λ_s and μ_s are the immigration emigration rate of the species s respectively, I and E are the maximum immigration and emigration rate of the habitat. In addition, each solution is modified depending on a user defined parameter probability, *pmod*. Each individual has its own μ and λ and are dependent on the species count (S) in the particular habitat. S_{max} refers to the maximum number of species count possible in a habitat. Poor solutions accept more useful information from good solutions, which improves the exploitation ability of the algorithm.

In mutation the function is used to increase the diversity of the population to get the good solutions. The aim of this scheme is to make an island with low habitat suitability index (HSI) more likely to mutate its suitability index variables (SIVs). (Simon 2008) defined the mutation function as:

$$m(S) = m_{\max}\left(\frac{1-P_s}{P_{\max}}\right)$$

where m_{max} is a user defined parameter, m(S) is the mutation rate for a habitat that contains S species, and P_{max} is the maximum probability. *Ps* is the probability that a habitat contains S number of species with *Ps* changing over time and given as:

$$P_s = \begin{cases} -(\lambda_s + \mu_s)P_s + \mu_{s+1}P_{s+1} & S = 0, \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1}P_{s-1} + \mu_{s+1}P_{s+1} & 1 \le S \le S_{\max} - 1, \\ -(\lambda_s + \mu_s)P_s + \lambda_{s-1}P_{s-1} & S = S_{\max} \end{cases}$$

The above procedures are conducted per the optimization process until the desired global minima or the termination condition is reached.

3. Experiment

3.1 Pseudocodes/ Algorithms

3.1.1 SRCGA Algorithm

The SRCGA algorithm proposed by (Sawyerr, Ali et al. 2011) makes use of three algorithms:

```
Linear-ranked Selection (Jebari, Madiafi 2013):
1. Calculate Sum v= 1/(n-2,001)
2. For each individual $1 \leq i \leq n$ do
    A) Generate a random number $\alpha \in [0, v]$
    B) For each $1 \leq j \leq n$ do
        If ($p(j) \leq \alpha$)
            Select jth individual
            Break
```

```
Stochastic Universal Sampling (Jebari, Madiafi 2013):
1. Calculate the mean F`
2. Generate a random number $\alpha \in [0, 1]$
3. Let sum = f(1), delta = $\alpha$ X F`, j=0
4. Do
    1. If (delta <sum)
        1. Select the jth individual
        2. Delta= delta +sum
    2. Else
        1. J=j+1
        2. Sum = sum+ f(j)
5. While (j<n)
```

```
SRCGA Main algorithm(Sawyerr, Ali et al. 2011):
1. initialize N solution points in randomly generated population P using uniform distribution
    1. t=0
    2. set $P_t = \{a_1, a_2, . . . , a_N \}$ from search space S.
    3. Set convergence = false
2. While ($t \leq$ Max) and NOT Convergence
    1. Evaluate $a_i$ for all a in $P_t$
    2. Perform LRS and SUS by selecting $m \leq N$ solutions with $P_t$ to form ˆ Pt .
    3. Perform arithmetic crossover (AC) with probability pc on ˆ Pt and use to create candidates. Save the candidates in $Z_t$ .
    4. with probability $p\mu$, on each component j of bi $\in Z_t$ mutate to create $M_t$ .
    5. for i= 1 to m do $P_t(i)=M_t$ to create Pt+1.
    6. If m = N, then elitism is applied.
    7. t = t +1
```

3.1.2 RCGA-P Algorithm

The RCGA-P algorithm proposed by Sawyerr et.al. (Sawyerr, Ali & Adewumi, 2011) makes use of the same algorithms as above but adds a fourth for projection:

```
Projection Algorithm
```

```
1. For each target offspring i do
    1. Randomly get next target offpsring j
    2. Obtain fitness i,j
    3. Obtain norm i,j
    4. If (fitness j> fitness I)
        1. Replace each element/gene of i with with projection formula
        2. Update new offspring
    5. Else
        1. Replace each element/gene of j with with projection formula
        2. Update new offspring
```

```
RCGA-P Main algorithm(Sawyerr, Ali et al. 2011):
1. initialize N solution points in randomly generated population P using uniform distribution
    1. t=0
    2. set $P_t$ = {$a_1$, $a_2$, . . . , $a_N$ } from search space S.
    3. Set convergence = false
2. While (t ≤ Max) and NOT Convergence
    1. Evaluate $a_i$ for all a in $P_t$
    2. Perform LRS and SUS by selecting m ≤ N solutions with $P_t$ to form ˆ Pt .
    3. Perform arithmetic crossover (AC) with probability pc on ˆ Pt and use to create candidates. Save the candidates in $Z_t$ .
    4. with probability pμ, on each component j of bi ∈ $Z_t$ mutate to create $M_t$ .
    5. For each $c_i$ ∈$M_t$ (i = 1, 2, . . . , m)
        1. generate the pair ($c_i$, $c_j$ ),
        2. create $s_i$ (i = 1, 2, . . . , m) using mathematical projection and generate $Q_t$ = {$s_1$, $s_2$, . . . , $s_m$ }.
    6. for i= 1 to m do $P_t(i)=Q_t$ to create Pt+1.
    7. If m = N, apply elitism.
    8. t=t+1
```

3.1.3 RCBBO Algorithm

From (Gong, Cai et al. 2010), the RCBBO algorithm comprised of:

```
Migration(Gong, Cai et al. 2010):
1. for i = 1 to N
    1. Select $X_i$ with probability $\alpha\lambda_i$
    2. if rndreal (0, 1) < $\lambda_i$ then
        1. for j = 1 to N do
            1. Select Xj with probability $\alpha\mu_j$
            2. if rndreal (0, 1) < lj then
                1. Randomly select an SIV σ from Xj
                2. Replace a random SIV in Xi with σ
            3. end if
        2. end for
    3. end if
2. end for
```

Mutation (Gong, Cai et al. 2010):

```
1. for i = 1 to N
    1. Compute the probability Pi
    2. Select SIV Xi(j) with probability αPi
    3. if rndreal (0, 1) < $m_i$ then
        1. Replace Xi(j) with a randomly generated SIV
    4. end if
2. end for
```

RCBBO Main algorithm (Simon 2013):

```
1. Initialize a population of candidate solutions {$x_k$} for k ϵ [1, N]
2. While not (termination criterion)
    1. For each $x_k$, set emigration probability $\mu_k$, α fitness of $x_k$, with $\mu_k$ ϵ [0,1]
    2. For each individual $x_k$, set immigration probability $x_k = 1 — \mu_k$
    3. {zk} ← {xk}
    4. For each individual $z_k$
        1. For each solution feature s
            1. Check migration against $x_{k \text{ of}}$ Zk
            2. If immigrating then
                1. Call migration
                2. zk(s) ←Xj{s)
            3. End if
        2. Next solution feature
        3. Call mutation {zk}
    5. Next individual
3. {xk} <- {zk}
4. Apply elitism
5. Next generation
```

3.2 Simulation Model

The object-oriented analysis and design method was used to model the problem and execute the experiments. At the heart of the simulation was six classes, i.e. the:

(a) *Main class* which was the main application that initiates the programme for finding the optimization required. The optimization parameters are given in section 3.3. Its primary method is the creation of the thread that performs the task of optimization
(b) *Thread class*, using the factory pattern, returns a thread or threads corresponding to the nature of the problem at hand. So, if the optimization parameter indicates SRCGA is to be optimized the thread class returns a thread for a collection of SRCGA to be optimized. The main work of the returned class is to drive the process of optimization via the evolve method; also, to write the results to an output file
(c) *Interface class* contains the methods that are to be implemented by each bio-inspired algorithm. In the case of the RCGAs it would be the elitism, selection, mutation, and cross-over methods, while for RCBBO it would be the migration, elitism and mutation methods
(d) *Collection classes* are central to the optimization process as they contain a collection of candidate solutions; they implement the bio-inspired methods of their respective interfaces and return the optimal solution as required by the algorithm (i.e. SRCGA, RCGA-P, RCBBO)
(e) *Solution classes* are the representation of the encoded possible solutions whose fitness are determined to find global minima
(f) *Benchmark class* refers to the test or objective function whose return value help determine the fitness of a candidate solution. In the case of this research 52 benchmark classes were created

3.3 Experiment & Parameter Selection

The following table captures the entire parameter value used in the experiment

Table 1 : *Experiment Parameters utilized for the Genetic and Biogeophraphy-based Optimization Algorithms*

Parameter Description	RCGA value	RCBBO value
Dimension (D)	4,10 and 20	4, 10 and 20
Maximum Number of Generation(T)	10,000	10,000
Number of trials per benchmark function	100	100
Number of Elitist Solution (E)	1	1
Size of Population(N)	10	10
Probability of Mutation (p_μ)	0.001	Nil
Probability of Crossover (p_c)	0.6	Nil
LRS expected value distribution (D*)*	1.1	Nil
Maximum Species Count (*SMax*)	Nil	5
Maximum Immigration Rate(*IMax*)	Nil	0.0025
Maximum Emigration Rate(*EMax*)	Nil	0.005
Maximum Mutation Rate(*MMax*)	Nil	0.005
Maximum Modification Rate (*ModMax*)	Nil	1

4. Results, Analysis & Conclusion

4.1 Results

In this section, the research presents performance comparison measures of SRCGA, RCGA-P and RCBBO using 52 optimization test problem. (Sawyerr, Ali et al. 2011)) mentions these performance criteria as robustness, convergence, and speed. robustness

The convergence measure (CM) provides an objective conclusion on how each solves the benchmark problem. In the experiment, a run terminates when the algorithm arrives at the best solution possible or when the maximum number of generations (T) has been reached. For all three algorithms convergence occurs when the best solution value f_{min} meets:

$$| f_{\min} - f(x^*) | \le \varepsilon, \quad \varepsilon = 10^{-4}$$

Figure 1 convergence criteria used for performance experiments

By summing the function calls of the benchmark problems made by the algorithm, the speed of such an algorithm is calculated. This approach is an objective measure are characteristics of the underlying machine, or performance of the algorithms themselves, which are a function of the ability of the programmer are not considered. For each algorithm, by taking the average of the function call counts over the entire runs the average speed otherwise called the average function evaluations (AFE) is derived.

The third criteria of robustness measure (RM) is obtained by cross-checking the success rate (SR) of the algorithms against one another. (Sawyerr, Ali et al. 2011) explained, that for the number of runs in each trial, the count of runs where the best solution was obtained is the success rate. In other words, the more times an algorithm find a solution, as given in figure 2 above, to the number of benchmark functions used as its objective function the higher the SR and thus the more robust the algorithm.

Table 2 : *AFE of the algorithms against benchmark problems along the test dimensions*

Dimension (D)	SRCGA	RCGA-P	RCBBO
4	62006.54	259419.88	**53994.74433**
10	**7166.338066595**	274116.72	158277.9333333
20	3745.12	24336.64	**1436.312335**
Grand Total	**72917.998066595**	557873.24	213708.9899983

The results show that overall, SRCGA has the highest speed in finding a solution with RCBBO and RCGA-P coming in second and third respectively. When D = 4, 20, RCBBO was the fastest.

Table 3: *CM of the algorithms against benchmark problems along the test dimensions*

	SRCGA	RCGA-P	RCBBO
MFE	72918	557873	213709
SR	13503	13900	10278
SR Ratio	90%	**93%**	69%

An overall initial summation of the AFE and SR as well as the calculation of SR ratio in the table below yields interesting results. First being that SRCGA appears more superior to both RCGA-P and RCBBO with respect to AFE. Secondly, RCGA-P has the most superior SR ratio (i.e. highest number of convergences per run) than SRCGA and RCBBO. However, for D = 10, SRCGA was unable to solve the modified langerman problem as its SR was zero. Also, for D = 20, RCGA-P was unable to solve the Epistatic Michalewicz and Shekel Foxhole problems; RCBBO was also unable to solve Shekel Foxhole 7, Multi-Gauss, Exponential, Epistatic Michalewicz and Easom problems. Excluding these results such that SR exists for all problems, over all runs for all three algorithms, a revised table is given below.

Table 4: *Adjusted CM of the algorithms against benchmark problems where SR exists along the test dimensions*

	SRCGA	RCGA-P	RCBBO
MFE	70919.3	293497.8	151655.7
SR	13503	13900	10278
SR Ratio	97.144%	**100%**	73.942%

This update reveals that SRCGA still maintains the best speed of the three with RCGA-P having the best success ratio.

Table 5 : *Robustness Measurement of the algorithms against benchmark problems along test dimensions*

Dimension (D)	SRCGA	RCGA-P	RCBBO
4	200.2164033	**137.4691535**	1574.151148
10	256.029889565058	**112.593935481629**	363240.26756936
20	**62.06572569**	82.74548692	345.4633825
Grand Total	518.312018555958	**332.808575901629**	365159.88209986

The results showed that after normalising and summing the SR ratio against the SPBest for each dimension and algorithm RCGA-P was the most robust algorithm with SRCGA and RCBBO second and third respectively. When D = 20, SRCGA was the most robust.

4.2 Analysis

Overall, as a group, the RCGAs displayed a higher robustness when compared with RCBBO especially with D = 20 where for all problems, RCBBO was not the most robust algorithm for all the runs. For D = 4 and 10, RCBBO did show superior performance over the others for Easom, Paviani, Ackley 1, Levy Montalvo 1, Levy Montalvo 2, Paviani, Salomon respectively. With the exception of D = 20 where SRCGA showed more robustness, RCGA-P otherwise was the most robust algorithm.

The statistical result shows that with increasing dimensions, the success rate of all the algorithms was reduced. Also, certain characteristics of problems made RCBBO perform better namely continuous, differentiable. On the whole GAs as a group showed to be more robust than RCBBO.

4.3 Conclusion

The result confirmed that the real-coded GAs are generally more robust than the real-coded BBO with the application of the projection feature enhancing this robustness. As such, problems whose have a range of applicable objective functions could benefit from the use of GAs. However performance based on speed showed that the real-coded BBO proffered a better alternative to the projection based GA variant (RCGA-P) in all cases; therefore problems for which convergence is of higher priority could consider BBOs..

Acknowledgements

Ayodele Boglo acknowledges Dr. Babatunde Sawyerr and Dr. E.P. Fasina, both lecturers at the Department of Computer Science, University of Lagos Nigeria, who stirred interest in bio-inspired computational algorithms and optimization.

References

Adewumi, A. (2016). *Fundamental of optimization: Models and methods,* Unpublished manuscript.

Ardeh, M. A. *Benchmark functions.* Retrieved 4th April, 2018, from http://benchmarkfcns.xyz/fcns

Binitha, S., & Sathya, S. S. (2012). A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering, 2*(2), 137-151.

Davis, L. (1991). Handbook of genetic algorithms.

Emami, M., Mozaffari, A., Azad, N. L., & Rezaie, B. (2016). An empirical investigation into the effects of chaos on different types of evolutionary crossover operators for efficient global search in complicated landscapes. *International Journal of Computer Mathematics, 93*(1), 3-26.

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction, 2nd edition* (2nd ed.) Wiley.

Floreano, D., & Mattiussi, C. (2008). *Bio-inspired artificial intelligence: Theories, methods, and technologies* MIT press.

Gilpin, M. E., & Hanski, I. A. (1997). *Metapopulation biology: Ecology, genetics, and evolution*

Gong, W., Cai, Z., Ling, C. X., & Li, H. (2010). A real-coded biogeography-based optimization with mutation. *Applied Mathematics and Computation, 216*(9), 2749-2758.

Herrera, F., Lozano, M., & Verdegay, J. L. (1998). Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review, 12*(4), 265-319.

Holland John, H. (1975). Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. *USA: University of Michigan,*

Hordri, N. F., Yuhaniz, S. S., & Nasien, D. (2013). A comparison study of biogeography-based optimization for optimization problems. *Int.J.Advance.Soft Comput.Appl, 5*(1)

Jamil, M., & Yang, X. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150-194.

Jebari, K., & Madiafi, M. (2013). Selection methods for genetic algorithms. *International Journal of Emerging Sciences, 3*(4), 333-344.

Jones, M. T. (2008). *Artificial intelligence: A systems approach* Laxmi Publications, Ltd.

Negnevitsky, M. (2005). *Artificial intelligence: A guide to intelligent systems* Pearson Education.

Pintea, C. (2014). *Advances in bio-inspired computing for combinatorial optimization problems* Springer.

Revees, C. R., & Rowe, J. E. (2004). Genetic algorithms–principles and perspectives.

Sarker, R. A., & Newton, C. S. (2007a). *Optimization modelling: A practical approach* CRC Press.

Sarker, R. A., & Newton, C. S. (2007b). *Optimization modelling: A practical approach* CRC Press.

Sawyerr, B. A., Ali, M. M., & Adewumi, A. O. (2011). A comparative study of some real-coded genetic algorithms for unconstrained global optimization. *Optimization Methods and Software, 26*(6), 945-970.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6), 702-713.

Simon, D. (2013). Evolutionary optimization algorithms: Biologically-inspired and population-based approaches to computer intelligence. hoboken.

Wilson, E. O., & MacArthur, R. H. (1967). The theory of island biogeography. *Princeton, NJ,*

Xing, B., & Gao, W. (2014). *Innovative computational intelligence: A rough guide to 134 clever algorithms* Springer.

Biographies

Ayodele Peter Boglo is an ICT Entrepreneur **-** founder of Carrtelcom Nigeria Limited and a Director of Corelytics Nigeria Limited. He earned B.Sc. (Hons) in Computer Science from The University of Ibadan, Oyo State Nigeria, MSc. in Engineering Management from University of Pretoria. His research interests include computational intelligence, simulation, optimization, complex systems, and Technology Strategy & Innovation. He is member of IEEE and the Nigeria Computer Society (NCS).

Barend Wilhelm Botha is a professional engineer registered with ECSA since 1994. He started his career as a mechanical system engineer in the defense industry in 1992 after obtaining both his B. Eng (Mech) and M. Eng (Mech) from the University of Pretoria. He later joined North-West University as senior lecturer where he obtained his PhD and acted as consultant to the Pebble Bed Modular Reactor project as well as quality manager to M-Tech Industrial. In 2008 he joined PBMR full time as a senior system engineer. After the closing of PBMR he joined the University of Pretoria lecturing in Reliability Engineering, Reliability Based Maintenance and Mechanical Design. In November 2013 he joined SNC-Lavalin as design manager applying systems engineering principles to integrate design with business through simulation of mining equipment and processes. After the closing of SNC-L SA he joined the University of Johannesburg as Senior Lecturer in Mechanical Engineering. In parallel he also developed courses in Engineering Systems Management and Reliability Management which he lectured in the Postgraduate School of Engineering Management. His current research focus is on systems thinking, engineering systems management, design management and optimization, operational excellence and related fields.

Babatunde A Sawyerr is a registered Computer Professional in Nigeria and currently serves as a Senior Lecturer and the acting Head of Department, Computer Science, University of Lagos. He obtained his BSc(Microbiology), Postgraduate Diploma, MSc and PhD in Computer Science from the University of Lagos. His professional memberships are: The Association of Computing Machinery (ACM), African Society for BioInformatics and

Computational Biology (ASBCB), Nigeria Computer Society (NCS) and SIGEvo Association for Computing Machinery (SIGEVO). His research interests include Computational Intelligence, Global Optimization, Bioinformatics and Computational Biology. Besides his teaching and lecturing responsibility, he also serves as a member of the leadership team of the Machine Intelligence Research Group (MIRG) at the Department.