hGRGA: A Scalable Genetic Algorithm using Homologous Gene Schema Replacement

Sumaiya Iqbal and Md Tamjidul Hoque[*]
Computer Science, University of New Orleans, Louisiana, 70148, USA

Abstract

In this article, we propose a new evolutionary algorithm, referred as *homologous Gene Replacement Genetic Algorithm* (hGRGA) that includes a novel and generic operator called *homologous Gene Replacement* (hGR). The hGR operator improves the chromosomes in gene level to promote their overall functionality. The hGRGA effectively encodes the key idea of the natural evolutionary process that locates and utilizes good local schema present in the genes of a chromosome through hGR operator. The proposed hGRGA is evaluated and compared with two variants of GA and two other state-of-the-art evolutionary computing algorithms based on widely-used benchmark functions with a motivation to apply to wider varieties of optimization problems. The simulation results show that the new algorithm can offer faster convergence and better precision while finding optima. Our analysis shows that hGR is effectively a scalable operator that makes hGRGA well suited for real world problems with increasing size and complexity.

1. INTRODUCTION

A stochastic search based evolutionary algorithm is applicable to wide variety of scientific research and engineering applications. Moreover, such an algorithm can be applied to non-convex and multimodal problems without the assumption of initial guess, differentiability and continuity of the objective functions. A rich literature is available on various evolutionary heuristics and swarm intelligence based search algorithms with numerous applications. These includes *Particle Swarm Optimization* (PSO) (Kennedy and Eberhart, 1995), *Artificial Colony Optimization* (ACO) (Dorigo et al., 1996), *Differential Evolution* (DE) (Storn and Price, 1997), *Genetic Algorithm* (GA) (Holland, 1992), *Artificial Bee Colony* (ABC) (Karaboga, 2005a), *Glowworm Swarm Optimization* (GSO) (Krishnanand and Ghose, 2005), *Cuckoo Search Algorithm* (CSA) (Yang and Deb, 2009), *Firefly Algorithm* (FA) (Fateen and Bonilla-Petriciolet, 2014), *Bat Algorithm* (BA) (Yang and Hossein Gandomi, 2012), *Monkey Algorithm* (MA) (Zhao and Tang, 2008), *Krill Herd Algorithm* (KHA) (Gandomi and Alavi, 2012), *Wind Driven Optimization* (WDO) (Bayraktar et al., 2010), *Social Spider Algorithm* (SSA) (Yu and Li, 2015). These algorithms have been modified as well as hybridized towards improved performances in both continuous and combinatorial scopes (Bao et al., 2015; Gong et al., 2014; Iqbal et al., 2015; Maher et al., 2014; Rashid et al., 2015; Zhang et al., 2014). To design effective nature-inspired evolutionary algorithms for solving combinatorial and continuous optimization problems remains to be a demanding research topic due to their wide applicability, especially with increasing size and complexity. In this study, we formulate a new genetic algorithm including a novel feature called *homologous gene replacement* (hGR) operator for continuous optimization problems. This operator aims at underpinning the key ideas of biological evolutionary process based classical genetic algorithm by improving a chromosome using its local genes.

John Holland first described the idea of genetic algorithm (GA) which was inspired by the Darwinian principles of biological evolution and adaptation in nature (Holland, 1975/1992). Henceforth, researchers have extensively investigated the components, theory and performance of GA (Davis, 1991; DeJong, 1975; Goldberg, 1994; Muhlenbein, 1992; Schmitt, 2001; Srinivas and Patnaik, 1994; Vose, 1993). A set of population and a set of genetic operators (GOs) such as crossover and mutation are the constituents of the basic GA. The population consists of a set

[*] Corresponding author
Email address: thoque@uno.edu

of chromosomes that represents the solutions and the genetic operators are applied on these chromosomes throughout the GA evaluations. Two primary challenges that are involved in the design process of a search algorithm are to effectively intensify the existing knowledge (*exploitation*) and to discover new knowledge by diversification (*exploration*). The canonical GA, also referred as simple GA (SGA) incorporates *crossover* and *mutation* to exploit and explore respectively (Goldberg, 1987; Goldberg, 1989; Goldberg, 1994). Several studies can be found in the literature that focus on modifying crossover and mutation operators to extract more information by GA search (Vasconcelos et al., 2001; Yazdani and Jolai, 2013). The idea of *elitist selection* or *elitism* was introduced in *Breeder Genetic Algorithm* (BGA) and was applied along with uniform crossover and mutation (Mühlenbein and Schlierkamp-Voosen, 1993a; Mühlenbein and Schlierkamp-Voosen, 1993b). Some other modeling of GA include (λ+μ)-*Evolutionary Strategy* (ES) (Back et al., 1991) based algorithms such as *Genitor* (Whitley and Kauth, 1988; Whitley and Starkweather, 1990; Whitley, 1989), CHC (*Cross generational elitist selection*, *Heterogeneous selection* and *Cataclysmic mutation*) algorithm (Eshelman, 2014) and executable model of GA (Voset and Liepinsl, 1991; Whitley, 1993b; Whitley, 1994), whereas some exploit parallelism by *Tournament Selection* (Goldberg, 1990; Goldberg and Deb, 1991), *Island Models* (Belding, 1995; Gorges-Schleuter, 1990; Starkweather et al., 1990; Tanese, 1989; Whitley and Starkweather, 1990) and cellular automata (Whitley, 1993a; Whitley, 1994). The tuning of population size and the operator rates resulted various GAs (Koumousis and Katsaras, 2006; Kühn et al., 2013). Further, Hoque et al. (2007) introduced useful *chromosome correlation factor* (CCF) based genetic operator named *twin removal* (TR) that can provide improved diversification. GA with TR operator was named TRGA which was able to avoid premature convergence as well as stalling in several applications (Higgs et al., 2012; Hoque et al., 2007; Hoque et al., 2011).

Researchers investigated the performance of GA in gene level as well. A variant called IGA in Jiao and Wang (2000) includes the idea of immunity through vaccination and immune selection that modifies some bits of the genes according to the prior knowledge to gain higher fitness with greater probability. Other variants include 'Ge_ GA' (Yang et al., 2003) that works by maintaining a gene pool and GTGA (Wang et al., 2006) that applies local search and gene therapy which is the insertion of eminent genes and the removal of morbid genes. However, these genes based GAs have been described for a particular application which is the travelling salesman problem. Here, instead we present a generic genetic operator that focuses on locating and utilizing healthy gene to improve the fitness of the corresponding chromosome and discuss this operation for generalized continuous optimization problem.

According to the principle of GA (Mitchell, 1995) there exists good *schemata* or *templates* in chromosomes that contribute to that chromosome's higher fitness. In nature, each organism has specific functions that are determined by the instruction set encoded within its chromosomes. Chromosome, which is equivalent to a DNA sequence, consists of genes those are connected together in the chromosome. Each of these genes is responsible for a function of that organism. Therefore, fully functional genes can result in a fitter chromosome so as a fitter organism. Inspired by this idea, we introduce a genetic operator called homologous gene replacement (hGR). It locates the best gene template in a chromosome and replaces relatively less fit genes of that particular chromosome with the best gene schema of that chromosome with a hope to have fast assessment for a better fit. We call this replacement homologous as the best gene and the target genes to be replaced belong to the same chromosome. Therefore, hGR distributes the good schema from local gene level to the full chromosome and eventually boosts up the overall functional capacity of that chromosome. The proposed GA using hGR operator is named *hGRGA*. While hGR acts as an efficient exploitation operator, we include TR to have balanced exploration in our evolutionary algorithm. We tested hGRGA on 24 benchmark functions for numerical optimization. The promising simulation results of hGRGA support the hypothesis that the local schema within a gene can be utilized to increase the overall functionality of the chromosome. Moreover, we analyze the adaptability of hGR operator for higher dimensional problems, which confirms the effectiveness of our approach towards larger applications as well.

The rest of the paper is organized as follows. **Section 2** reviews the basics of genetic algorithm and describes the other two existing GA variants studied under this work to compare with the proposed hGRGA algorithm. **Section 3** introduces the idea and formulation of hGR operator along with a detail discussion of the structure of hGRGA. **Section 4** presents the benchmark functions used to evaluate the proposed algorithm, comparison of hGRGA with two existing

GA variants and two other state-of-the-art evolutionary computing algorithms. **Section 4** further assesses the effect of hGR operator and scalability of hGRGA empirically. Finally, we briefly conclude in **Section 5**.

2. BACKGROUND

In this section, we review the basics of genetic algorithm and two other existing genetic algorithm variations, SGA and TRGA. In this study, we formulate each optimization problem as a minimization problem. The following terminologies and notations are used throughout the paper.

- $\boldsymbol{x} = [x_1, \ \cdots, \ x_d]^T$ is a d –dimensional solution.
- $f(\boldsymbol{x})$ is the objective function to be optimized (or minimized)
- N is the population size or number of chromosomes or individuals
- d is the number of variables in a solution or genes in a chromosome or dimensions of a search space or problem size
- $C = [g_1, \ \cdots, \ g_d]^T$ is a chromosome with d genes. Each solution $(\boldsymbol{x})$ is encoded in a chromosome (C) in binary where each gene (g_i) is mapped to each variable (x_i).
- m is the number of bits (b_i) to represent each gene or variable. The value of m can vary for different search ranges to ensure desired level of precision in representing variables.
- $L = d \times m$ is the bit length of a chromosome

The canonical GA starts with generating initial population randomly. Then, the fitness values of the chromosomes are quantified by evaluating the objective function. After that, the chromosomes are tweaked using genetic operators according to the predefined rates to create new generation population. It is heuristically surmised that each successive generation will contain fitter chromosomes than previous generation as the new population is always derived utilizing fitter chromosomes with good genes of the current population. GA is an iterative algorithm and can run for a predetermined number of generations. We set the termination criteria as either the desirable solution is found or the predefined number of generations are executed. The flow of operations in each GA cycle is shown in **Figure 1**.

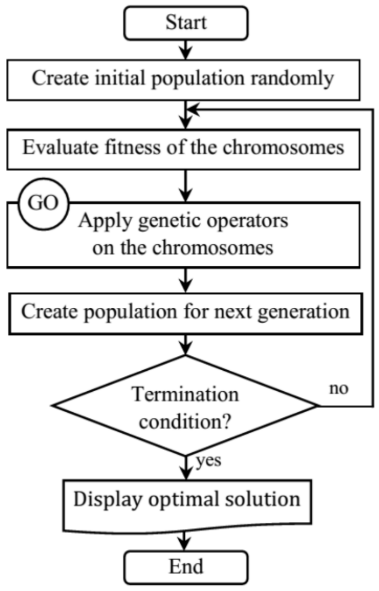

Start
Create initial population randomly
Evaluate fitness of the chromosomes
GO
Apply genetic operators on the chromosomes
Create population for next generation
Termination condition?
no
yes
Display optimal solution
End

Figure 1: Flowchart of the basic genetic algorithm. Here, GO is *genetic operators*.

2.1 Simple Genetic Algorithm (SGA)

Our implementation of SGA under this work involves three classical genetic operators: *i*) GO – 1: *Elitism*, *ii*) GO – 2: *Crossover*, and *iii*) GO – 3: *Mutation*. These operators have been extensively studied in different flavors (Digalakis and Margaritis, 2002; Yoon and Moon, 2002) on numerous applications (Haupt and Haupt, 2004; Michalewicz, 2013). We utilized single point crossover and single bit mutation operator.

In the elitism process, a predefined proportion (E_p) of highly fit chromosomes termed *elites* are directly transferred to the next generation from current generation. Elitism is useful since it guarantees that the best solution found thus far survives and the solutions of GA are non-decreasing over time.

Crossover imitates the natural evolutionary process of mating between two chromosomes to breed new chromosomes. In this process, two selected individuals called *parents* (P_1 and P_2) are combined by exchanging their genes to produce two new individuals for next generation called *off-springs* (O_1 and O_2). We applied fitness proportionate (also known as roulette wheel) selection mechanism to select parents for crossover. By this process, chromosomes with higher fitness values get higher priority to pass their genes into next generation. This selection process relies on the idea that better parents can produce better off-springs after mating. Crossover is SGA's basic operator to intensify the search within the currently available solutions. The amount of crossover is controlled by the crossover probability (C_p), thus the number is crossover is $n_C = (N \times C_p)/2$ where each crossover produces two children.

Mutation interprets the evolutionary process of having genetic diversity in nature. In this process, a candidate is randomly selected for mutation (M_c) and then it is randomly changed to produce a new individual for next generation population. In this work, we flipped one single bit of the chromosome. Mutation serves as the basic operator to explore the search space in SGA, which is important to inhibit premature convergence into local optimal solution. We apply mutation at a predetermined probability (M_p) which is usually kept low (Digalakis and Margaritis, 2002). Therefore the number of mutation is $n_M = N \times M_p$. The SGA implemented under this work follows the general workflow of GA shown in **Figure 1** with the GO field expanded in **Table 1**.

Table 1. Pseudo codes of genetic operators elitism, single point crossover and single point mutation.

```
    GO – 1: Elitism
1)  Evaluate the fitness of the chromosomes of the current population and sort according to the fitness.
2)  Transfer $n_E = N \times E_p$ number of the chromosomes from the sorted sequence with higher fitness
    into new population.
    GO – 2: Single point crossover
3)  Select $P_1$ and $P_2$ using fitness proportionate selection
        ➢ $P_1 = [b_1, b_2, \dots, b_L]$ and $P_2 = [b_1, b_2, \dots, b_L]$
        ➢ $L = d \times m$ is the bit length of a chromosome
4)  Select a locus ($l_c$) of crossover, where $l_c \in \{1, \dots, L-1\}$
5)  Exchange the part of parent at the locus to generate $O_1$ and $O_2$
        ➢ $O_1 = P_1[b_1, \dots, b_{l_c}] + P_2[b_{l_c+1}, \dots, b_L]$
        ➢ $O_2 = P_2[b_1, \dots, b_{l_c}] + P_1[b_{l_c+1}, \dots, b_L]$
6)  Accept $O_1$ and $O_2$ into next generation.
7)  Repeat step (3) to (6) for $n_C$ times.
```

```
GO – 3: Single bit mutation
8)  Select a candidate chromosome for mutation, $M_C = [b_1, b_2, \ldots, b_L]$
9)  Select a locus ($l_m$) of mutation, where $l_m \in \{1, \ldots, L\}$
10) Flip the bit of $M_C$ at $l_m$ to generate mutated chromosome, $\overline{M_C}$
        ➢ $\overline{M_C} = M_C\left[b_1, \ldots, \overline{b_{l_m}}, \ldots, b_L\right]$
11) Accept $\overline{M_C}$ into next generation
12) Repeat step (8) to (11) for $n_M$ times
```

2.2 Twin Removal Genetic Algorithm (TRGA)

TRGA (Hoque et al., 2005; Hoque et al., 2007) is an improved variation of GA that includes an additional operator, GO – 4: *Twin Removal* (TR). In SGA, mutation is the primary operation for diversification. TR overcomes the limitations of mutation operator in presence of large number of similar chromosomes (*twins*) within population, causing *genetic drift* (Coello, 2000) phenomena. TR is governed by the allowable degree of similarity between chromosomes by calculating the *chromosome correlation factor* (CCF) (Hoque et al., 2007). The value of CCF equal to 0.8 is empirically found to be the most effective in avoiding the stall condition . We allow the CCF to vary from 1.0 (100% similarity) to 0.8 (80% similarity) by a low rate of decrement (∂CCF) in consecutive generations. As the generations proceed, the population become more correlated. Thus we allow lesser similarity to mitigate this correlation among chromosomes. After a chromosome is detected as a twin and replaced by a random new chromosome, we mark it as '*new*' to avoid redundant computation involved in repetitive twin evaluation. TRGA follows the GA steps listed in **Figure 1** where the GO field is composed of GO – 1 to 3 (**Table 1**) and the additional GO – 4 described in **Table 2**.

Table 2. Pseudo code of twin removal genetic operator.

```
GO – 4: Twin removal
1) Count the number of loci with identical bit values ('similarity') between a pair of chromosomes
   under consideration
2) Replace the less fitted chromosome by a new random chromosome, if $similarity > CCF$ and
   mark it as 'new'
3) Update $CCF = CCF - \partial CCF$ if $CCF > 0.8$
4) Repeat step (1) to (3) for each pair of chromosomes without chromosomes labeled as 'new'
```

3. HOMOLOGOUS GENE REPLACEMENT BASED GA

The aim of this paper is to introduce a new genetic algorithm named hGRGA that focuses on the working principle of GA in – "locating, prioritizing, and recombining good building blocks (*schemas*) of solutions (chromosomes)", however with a different view. The well-known fundamental theorem of genetic algorithm called *schema theorem* (Goldberg, 1989; Holland, 1975/1992) states the concept of schema that is a set of bit strings, specifically "*short*", that have above-average-fitness and contribute to higher fitness of a chromosome. Therefore, we look into the relatively shorter length bit string corresponding to each gene (or variable) instead of the longer bit string of the full chromosome to discover schemas. It is also well-studied under *schema theorem* that crossover and mutation break the existing schema of a chromosome at random points, therefore a better solution schema can be lost even with a good selection procedure (Hoque et al., 2007; Mitchell, 1995; Whitley, 1994).

Elitism can eliminate such disruptive effects of crossover and mutation to some extent as elitism operator separates a subset of chromosomes called elites with relatively higher performance before crossover and mutation and propagates these chromosomes directly to the next generation. An immediate impact of elitism is to bias the selection procedure to utilize the improved schemas within elites. Homologous gene replacement (hGR) operator further boosts up the fitness of elites and enhances the benefit provided by elitism (Iqbal and Hoque, 2016a; Iqbal and Hoque, 2016b). The hGR operator finds the best gene within each elite chromosome and replaces the homolog genes of the respective elite that are relatively worse by the best gene schema. Unlike other GAs that employ gene based operator on the chromosomes and accept them with a probability, we apply hGR on the elites for fourfold reasons.

1) We hypothesize that the high-average-fitness schema exist in elites with higher chance than in other individuals as the elite individuals have relatively higher fitness values than others in the population. We intend to prioritize those schema fragments of elites only.
2) We increase the fitness of the elites by hGR relative to the other chromosomes so that the fitness proportionate selection is strongly guided towards highly fitted elites to select them as parents for crossover. Therefore, we have an immediate impact of hGR on crossover to operate on those elite's schema and generate better offsprings from parents with higher fitness.
3) We want the schemas of the improved individuals (specifically, elites) that are produced by hGR operator to survive the disruption effect of crossover and mutation. The elites have separate entries in the population to pass their genetic material intact to the next generation. At the same time, they can participate in crossover as parents as hGR is applied ahead of crossover and mutation in the cycle of hGRGA (**Figure 3**). The children produced by crossover have separate entries in the next generation population. In our implementation, we do not select elite as a candidate for mutation. Thus, the good schemas distributed by hGR can be tweaked to generate better offsprings during crossover as well as can stay undamaged throughout evolution by avoiding disruption.
4) We aim to avoid the saturation of population with the schema of the best gene that may occur by applying hGR to all the individuals.

The biological metaphor behind hGR operation is that the unique functions of each organism are encoded in its chromosome and a gene is a region of a chromosome responsible for a specific trait or function (Townsend, 2003). Thus, genes are the functional units of a chromosome and an organism can be considered as fitter when the constituent genes of its chromosome are fully functional. Therefore, the hGR operator in hGRGA aims towards propagating the good and short schema present in the best gene of an elite with above-average fitness from local gene to the full chromosome to improve its overall functional capacity.

We apply hGR operator at a predefined rate denoted by r^{hGR}. This rate of gene replacement (r^{hGR}) determines the number of weaker genes to be replaced, $n_{hGR} = r^{hGR} \times d$ in each elite. Here, d is the number of genes of a chromosome. To determine the best gene template of an elite chromosome (E_c), we evaluate the fitness contribution of each variable or gene within the elite. While computing the relative fitness of a particular gene, we equally weight the rest of the genes (variables) using a common value. We used two different common values ('0' and '1') to deactivate or silence the effect of other genes while evaluating the relative fitness of a particular gene of an elite chromosome. The reasoning behind using two different values is the value '0' can completely deactivate other variables while quantifying the relative fitness of a particular gene or variable. This is useful only when the variables can be isolated and can be evaluated separately. However, the relative contribution of a variable can depend on the active values of other variables in case of *epistasis* where the variables interact with each other. To take the later case into consideration, we weight the other variables by a common non-zero value '1'. We repeat the process of applying hGR (**Table 3**) using both of the common values to generate two different new elites from the original one. Finally, we keep the better one in the next generation population. For a particular elite, we sort the genes according to their fitness, pick the best gene (g_{best}) from the sorted sequence and insert it in place of the unhealthy genes from the sorted sequence (the least fitted gene first). The original elite (before applying hGR) and the new elite (after applying hGR) are denoted by $E_{c(org)}$ and $E_{c(hGR)}$, respectively.

However, the schema pattern of different elites can be different to accept different number of gene replacement. Thus, it is desirable to keep the value of n_{hGR} adaptive to the performance of different elites. To make this happen, we apply hGR at the rate r^{hGR} in initial trial (t). If the corresponding elite accepts the gene replacement according to the prior assumption of having potential benefit, we increase the rate by ∂r^{hGR}. We keep on applying hGR for t number of times with increased rate till the overall fitness of the elite keeps on increasing with respect to the original one. In each trial, the number of gene replacement can be formulated in the following way,

$$\text{For, } t = 1, n_{hGR} = \lceil d \times (r^{hGR} + 0.\partial r^{hGR}) \rceil$$
$$t = 2, n_{hGR} = \lceil d \times (r^{hGR} + 1.\partial r^{hGR}) \rceil$$
$$\dots \dots \dots$$
$$\text{Therefore, } n_{hGR} = \lceil d \times [r^{hGR} + (t-1)\partial r^{hGR}] \rceil \quad (1)$$

We stop applying hGR if it degrades the overall fitness of the elite or, there are no more genes to be replaced ($n_{hGR} > d$). The steps of hGR are listed in **Table 3.**

Table 3. Pseudo code of homologous gene replacement.

```
    GO – 5: Homologous gene replacement
1)  Assign, $E_{c(org)} \leftarrow E_c$ and $t \leftarrow 1$
2)  Evaluate the genes of $E_{c(org)}$ and sort
        ➢ $g_{best} \leftarrow$ best gene of $E_{c(org)}$
3)  Compute number of less fit genes to be replaced, $n_{hGR} = \lceil d \times [r^{hGR} + (t-1)\partial r^{hGR}] \rceil$
4)  Generate $E_{c(hGR)} \leftarrow E_{c(org)}$ with $n_{hGR}$ less fit genes replaced by $g_{best}$, if $n_{hGR} \le d$
5)  Repeat step (3) to (4) with $t = t + 1$ if $f(E_{c(hGR)}) < f(E_{c(org)})$
6)  Repeat step (2) to (5) for $n_E$ number of $E_c$
```

We control the application of hGR by r^{hGR} and ∂r^{hGR} so that we can use this operator with care to have optimal number of gene replacement. With predefined r^{hGR} and ∂r^{hGR}, the number of trial for gene replacement,t,has an upper bound (lemma 1).

Lemma 1. With fixed rate and increase of rate of hGR (r^{hGR} and ∂r^{hGR}), the number of application of hGR operator, t, on each elite chromosome has the upper bound equal to $\left(\frac{1-r^{hGR}}{\partial r^{hGR}}\right) + 1$.

Proof. By hGR operation, we can replace maximum d genes of a single elite chromosome if the performance continues to increase with it. Therefore, we can write,

Number of genes replaced by hGR, $n_{hGR} \le d$

$\Rightarrow \lceil d \times [r^{hGR} + (t-1)\partial r^{hGR}] \rceil \le d$ by Equation (1)

$\Rightarrow r^{hGR} + (t-1)\partial r^{hGR} \le 1$

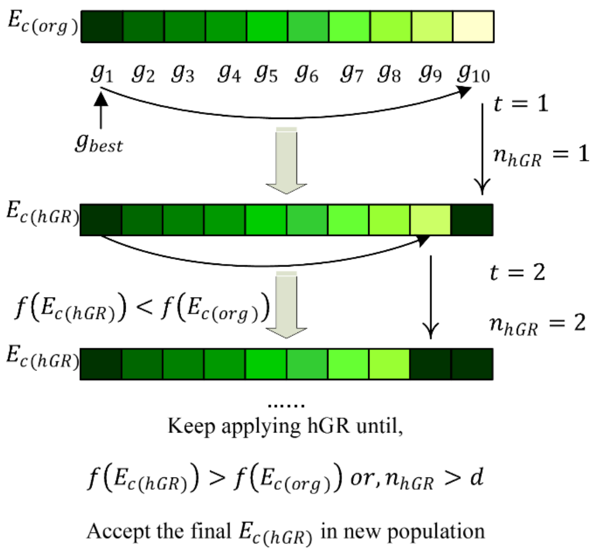
$\Rightarrow (t-1)\partial r^{hGR} \le 1 - r^{hGR}$

$$\Rightarrow (t-1) \le \frac{1-r^{hGR}}{\partial r^{hGR}}$$

$$\Rightarrow t \le \left(\frac{1-r^{hGR}}{\partial r^{hGR}}\right) + 1$$

$$\text{Thus, } t = O\left(\left(\frac{1-r^{hGR}}{\partial r^{hGR}}\right) + 1\right) \quad (2)$$

Therefore, hGR can be tested on an elite for maximum t times given by the **Equation 2** to extract the best possible information out of that elite. This upper bound of t is a constant defined by r^{hGR} and ∂r^{hGR} and independent of d. Thus, hGR is applied maximum t times irrespective of the problem size, d. Only the number of gene replacement (n_{hGR}) is adapted dynamically with d (**Equation 1**) at each t. Note that, another possible way of applying hGR is to replace each of the other genes by the best one and check for the improvement. However, it would increase the time complexity of the operator with the increase of problem size (number of genes). Therefore, it is useful to apply hGR at a pre-defined low rate that is increased slowly as the individual's fitness improves. This improves computational efficiency while offers similar fitness gain from hGR. A sample application of hGR with d = 10, r^{hGR} = 0.1 and ∂r^{hGR}= 0.05 is presented graphically in **Figure 2**. In the figure, the darker the color of a gene is, the higher is the relative fitness.



$E_{c(org)}$
g_1 g_2 g_3 g_4 g_5 g_6 g_7 g_8 g_9 g_{10}
g_{best}
$t = 1$
$n_{hGR} = 1$
$E_{c(hGR)}$
$t = 2$
$n_{hGR} = 2$
$f(E_{c(hGR)}) < f(E_{c(org)})$
$E_{c(hGR)}$
......
Keep applying hGR until,
$f(E_{c(hGR)}) > f(E_{c(org)})$ $or, n_{hGR} > d$
Accept the final $E_{c(hGR)}$ in new population

Figure 2: Sample illustration of hGR operation. Assume, d = 10, r^{hGR} = 0.1 and ∂r^{hGR} = 0.05. The darker is the color of a gene, the higher is its fitness.

The proposed hGRGA combines elitism with hGR, crossover, mutation and TR in one cycle or epoch of GA. We apply hGR on the elites before crossover to guide the selection procedure towards better chromosomes for next generation. Therefore, the hGR operator can effectively enhance the exploitation capacity of GA. In addition, we utilize the TR operator to have balanced diversification. We observed two different level of impacts out of hGR operation.

1) ***Fast impact*:** hGR is able to optimize all the variables simultaneously using the best one and can produce improved elites that are either the global optimum or near-global optima. Therefore, it can result a very fast convergence as soon as it can correctly identify the best variable (or gene) by disseminating the best gene and emphasizing on the best gene schema through crossover. This case can occur when the variables are independent or separable or the optimum lies in a point in the search space with same values in all the variables.
2) ***Slow impact*:** hGR may not result significantly fast convergence. Nevertheless, the operator may result reasonable improvement in the fitness of some elites by distributing good genes, which is enough to bias the selection and boost up the power of crossover to extract more information and result a better solution. This case can occur, but not limited to when the variables are nonseparable or interrelated.

Note that, hGR dynamically adjusts the optimal number of genes to be replaced with the dimensions (d) of the problem space by **Equation 1**. This makes the hGRGA scalable to higher dimensions or problem size if an elite's current bit pattern can accept the gene replacement towards improvement. Therefore, the number of epochs (generations) needed to converge is independent of the problem size. It is justified empirically in **Section 4.2**. The complete flow chart of each cycle of the proposed hGRGA is shown in **Figure 3**. We terminate a cycle when the maximum number of generations have been evaluated or the optimal solution is reached.

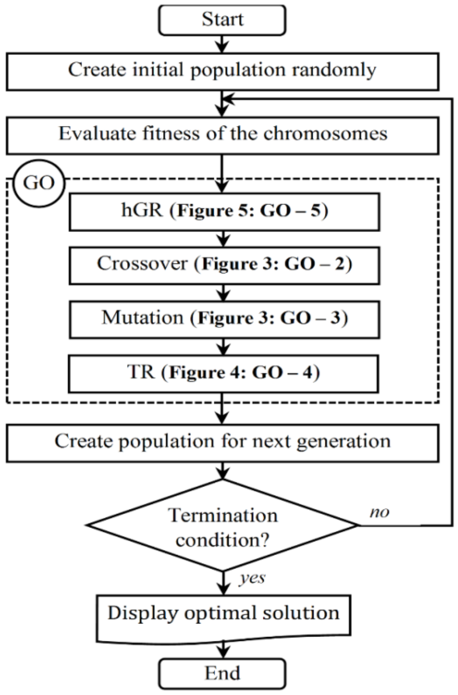

Start
Create initial population randomly
Evaluate fitness of the chromosomes
GO
hGR (Figure 5: GO – 5)
Crossover (Figure 3: GO – 2)
Mutation (Figure 3: GO – 3)
TR (Figure 4: GO – 4)
Create population for next generation
Termination condition?
no
yes
Display optimal solution
End

Figure 3: Flowchart of the proposed hGRGA algorithm.

4. SIMULATION RESULTS

We assess the performance of hGRGA on twenty-four benchmark test functions including fifteen base functions, whereas the rest of the functions are rotated, shifted or hybrid form of the base functions. Among them eighteen functions are considered to compare hGRGA with other GA variants, SGA and TRGA. However, we conduct the performance comparison of hGRGA with the other state-of-the-art evolutionary algorithms, including Differential Evolution (DE) (Price et al., 2006) and Artificial Bee Colony (ABC) Algorithm (Karaboga, 2005b; Karaboga and Basturk, 2007; Karaboga and Basturk, 2008) on the full test set. All of these functions are scalable to higher number of dimension as well as adopt additional challenging properties such as separability and modality (Dieterich and Hartke, 2012). We aggregated the base functions from the latest *Competition on Single Objective Real Parameter Numerical Optimization* at CEC 2013 and 2014 (Liang et al., 2013a; Liang et al., 2013b) and a survey of test functions by Jamil and Yang (2013) for global optimization. We categorize these functions into five types based on their properties. **Table 4** introduces the properties of each category of the functions, the function names, definitions and global minima ($\mathbf{x}^*$) in the landscapes along with the minimum function value ($f(\mathbf{x}^*)$).

Table 4. Description (name, definition, coordinates and functional value of minima) of benchmark test functions.

Name	**Definition**	$\boldsymbol{x}^*$ **and** $\boldsymbol{f(x^*)}$
	Type I: unimodal and separable	
Bent Cigar	$f_1(\boldsymbol{x}) = x_1^2 + 10^6 \sum_{i=2}^{d} x_i^2$	$\boldsymbol{x}^* = (0, 0, \dots, 0)$ $f_1(\boldsymbol{x}^*) = 0$
Discus	$f_2(\boldsymbol{x}) = 10^6 x_1^2 + \sum_{i=2}^{d} x_i^2$	$\boldsymbol{x}^* = (0, 0, \dots, 0)$

		$f_2(\boldsymbol{x}^*) = 0$
	Type II: unimodal nonseparable	
Zakharov	$f_3(\boldsymbol{x}) = \sum_{i=1}^{d} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^4$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_3(\boldsymbol{x}^*) = 0$
Schwefel's 1.2	$f_4(\boldsymbol{x}) = \sum_{i=1}^{d}\left(\sum_{j=1}^{i} x_j\right)^2$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_4(\boldsymbol{x}^*) = 0$
Schwefel's 2.2	$f_5(\boldsymbol{x}) = \sum_{i=1}^{d}\lvert x_i\rvert + \prod_{i=1}^{d}\lvert x_i\rvert$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_5(\boldsymbol{x}^*) = 0$
	Type III: multimodal and separable	
Rastrigin	$f_6(\boldsymbol{x}) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_6(\boldsymbol{x}^*) = 0$
Schwefel's 2.26	$f_7(\boldsymbol{x}) = 418.9829d - \sum_{i=1}^{d} x_i sin\left(\sqrt{\lvert x_i\rvert}\right)$	$\boldsymbol{x}^* = (420.9687, \ldots, 420.9687)$ $f_7(\boldsymbol{x}^*) = 0$
Michalewicz	$f_8(\boldsymbol{x}) = -\sum_{i=1}^{d} sin(x_i)\, sin^{2m}\left(\frac{ix_i^2}{\pi}\right), m = 10$	$\boldsymbol{x}^* = (2.20, 1.57), d = 2$ $f_8(\boldsymbol{x}^*) = -9.66015, d = 10$
Styblinski-Tang	$f_9(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$	$\boldsymbol{x}^* = (-2.90, \ldots, -2.90)$ $f_9(\boldsymbol{x}^*) = -39.16599d$
	Type IV: multimodal and nonseparable	
Happy Cat	$f_{10}(\boldsymbol{x}) = \left\lvert\sum_{i=1}^{d} x_i^2 - d\right\rvert^{1/4} + \frac{\left(\frac{1}{2}\sum_{i=1}^{d} x_i^2 + \sum_{i=1}^{d} x_i\right)}{d} + \frac{1}{2}$	$\boldsymbol{x}^* = (-1, -1, \ldots, -1)$ $f_{10}(\boldsymbol{x}^*) = 0$
Griewank	$f_{11}(\boldsymbol{x}) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_{11}(\boldsymbol{x}^*) = 0$
Ackley	$f_{12}(\boldsymbol{x}) = -ae^{\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)} - e^{\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right)} + a + e^1$ $a = 20, b = 0.2, c = 2\pi$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_{12}(\boldsymbol{x}^*) = 0$
Rosenbrock	$f_{13}(\boldsymbol{x}) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$\boldsymbol{x}^* = (1, 1, \ldots, 1)$ $f_{13}(\boldsymbol{x}^*) = 0$
Expanded Schaffer's F6	$f_{14}(\boldsymbol{x}) = \left[\sum_{i=1}^{d-1} g(x_i, x_{i+1})\right] + g(x_d, x_1)$	$\boldsymbol{x}^* = (0, 0, \ldots, 0)$ $f_{14}(\boldsymbol{x}^*) = 0$
	$g(x_1, x_2) = \sum_{i=1}^{d-1}\left(\frac{sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} + 0.5\right)$	
Expanded Griewank's plus Rosenbrock's	$f_{15}(\boldsymbol{x}) = \left[\sum_{i=1}^{d-1} f_{11}\left(f_{13}(x_i, x_{i+1})\right)\right] + f_{11}\left(f_{13}(x_d, x_1)\right)$	$\boldsymbol{x}^* = (1, 1, \ldots, 1)$ $f_{15}(\boldsymbol{x}^*) = 0$
	Type V: hybrid	

Hybrid Function – 1: Bent Cigar + Rastrigin + Schwefel's 2.26	$f_{16}(\boldsymbol{x}) = f_1(\boldsymbol{x_1}) + f_6(\boldsymbol{x_2}) + f_7(\boldsymbol{x_3})$ $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$	$\boldsymbol{x_1^*} = (0, \dots, 0), \boldsymbol{x_2^*} = (0, \dots, 0)$ $\boldsymbol{x_3^*} = (420.9687, \dots, 420.9687)$ $f_{16}(\boldsymbol{x}^*) = 0$
Hybrid Function – 2: Schwefel's 2.2 + Rastrigin + Griewank	$f_{17}(\boldsymbol{x}) = f_5(\boldsymbol{x_1}) + f_6(\boldsymbol{x_2}) + f_{11}(\boldsymbol{x_3})$ $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$	$\boldsymbol{x_1^*} = (0, \dots, 0), \boldsymbol{x_2^*} = (0, \dots, 0)$ $\boldsymbol{x_3^*} = (0, \dots, 0)$ $f_{17}(\boldsymbol{x}^*) = 0$
Hybrid Function – 3: Rosenbrock + Griewank + Discus	$f_{18}(\boldsymbol{x}) = f_{13}(\boldsymbol{x_1}) + f_{11}(\boldsymbol{x_2}) + f_2(\boldsymbol{x_3})$ $\boldsymbol{x} = [\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}]$	$\boldsymbol{x_1^*} = (1, \dots, 1), \boldsymbol{x_2^*} = (0, \dots, 0)$ $\boldsymbol{x_3^*} = (0, \dots, 0)$ $f_{18}(\boldsymbol{x}^*) = 0$
Rotated Rastrigin Function	$f_{19}(\boldsymbol{z}) = f_6(\boldsymbol{z}), \quad \boldsymbol{z} = \boldsymbol{R} * \boldsymbol{x}$	$\boldsymbol{z}^* = (0, 0, \dots, 0)$ $f_{19}(\boldsymbol{z}^*) = 0$
Shifted Rastrigin Function	$f_{20}(\boldsymbol{z}) = f_6(\boldsymbol{z}), \quad \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o_{new}} + \boldsymbol{o_{old}}) * 5.2/100$ $\boldsymbol{o_{old}} = old\ optimum = global\ optimum\ of\ f_6(\boldsymbol{x})$ $\boldsymbol{o_{new}} = new\ shifted\ global\ optimum$	$\boldsymbol{z}^* = shifted\ optimum = \boldsymbol{o_{new}}$ $f_{20}(\boldsymbol{z}^*) = 0$
Rotated Griewank Function	$f_{21}(\boldsymbol{z}) = f_{11}(\boldsymbol{z}), \quad \boldsymbol{z} = \boldsymbol{R} * \boldsymbol{x}$	$\boldsymbol{z}^* = (0, 0, \dots, 0)$ $f_{21}(\boldsymbol{z}^*) = 0$
Shifted Griewank Function	$f_{22}(\boldsymbol{z}) = f_{11}(\boldsymbol{z}), \quad \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o_{new}} + \boldsymbol{o_{old}})$ $\boldsymbol{o_{old}} = old\ optimum = global\ optimum\ of\ f_{11}(\boldsymbol{x})$ $\boldsymbol{o_{new}} = new\ shifted\ global\ optimum$	$\boldsymbol{z}^* = shifted\ optimum = \boldsymbol{o_{new}}$ $f_{22}(\boldsymbol{z}^*) = 0$
Rotated Ackley Function	$f_{23}(\boldsymbol{z}) = f_{12}(\boldsymbol{z}), \quad \boldsymbol{z} = \boldsymbol{R} * \boldsymbol{x}$	$\boldsymbol{z}^* = (0, 0, \dots, 0)$ $f_{23}(\boldsymbol{z}^*) = 0$
Shifted Ackley Function	$f_{24}(\boldsymbol{z}) = f_{12}(\boldsymbol{z}), \quad \boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{o_{new}} + \boldsymbol{o_{old}})$ $\boldsymbol{o_{old}} = old\ optimum = global\ optimum\ of\ f_{12}(\boldsymbol{x})$ $\boldsymbol{o_{new}} = new\ shifted\ global\ optimum$	$\boldsymbol{z}^* = shifted\ optimum = \boldsymbol{o_{new}}$ $f_{24}(\boldsymbol{z}^*) = 0$

The unimodal functions with single minima in the landscape belong to type I and II. These functions are primarily utilized to test the convergence speed of a searching algorithm. Type I functions are relatively easier being separable, whereas type II functions are more challenging with nonseparable variables. Type III and IV encompasses ten multimodal functions with the most complex property of having large number of local minima distributed in the landscape. These functions examine the capacity of the algorithm to get out of the local minima and reach the ultimate global minima. Type IV constitutes the most difficult multimodal functions being nonseparable as one variable affects the behavior of other variables. Type V contains hybrid, rotated and shifted functions whose base functions belong to Type I – IV. For hybrid functions, different subsets of the solution variables are passed to different constituent functions with diverge properties. Therefore, these function set resembles the complexity involved in real world optimization problems. The rotated functions are obtained by rotating the problem landscape using an orthogonal rotation matrix (**R**) (Salomon, 1996) to generate even more challenging landscape having asymmetrical grid structure (Liang et al., 2005). Rotation prevents the global optima to lie along the coordinate axes while preserves the original properties of the functions (Liang et al., 2005). For the shifted functions, the global optimum is shifted to a random position using a shifting vector ($\boldsymbol{o_{new}}$). The shifting is done so that the global optimum has different variable values for different dimensions and it does not lie at the origin or in the center of the search space (Liang et al., 2005). The rotation matrices ($\mathbf{R}$) and shifting vectors ($\boldsymbol{o_{new}}$) utilized in this work are the ones used in CEC 2014 (Liang et al., 2013a) and are collected from a shared link[1] available online. **Table 5** reports the search domain (range of variable values) and the numbers of bits used to encode each variable (gene) value for different functions along with the corresponding function references. It is important to note that the number of bits required to represent each variable of the solution can be different for different functions due to their different ranges of variable values defined by the corresponding search domains.

Table 5. Search domain and number of bits used to represent one gene (variable) of a chromosome for different functions. The full binary string for one gene includes decimal part, fractional part and sign part (one bit).

Name [Ref.]	**Search domain**	**Number of bits per variable (gene) [decimal + fraction + sign]**
Bent Cigar (Liang et al., 2013a)	[-100, 100]	7 + 12 + 1 = 20
Discus (Liang et al., 2013a)	[-100, 100]	7 + 12 + 1 = 20
Zakharov (Jamil and Yang, 2013)	[-5, 10]	4 + 12 + 1 = 17
Schwefel's 1.2 (Liang et al., 2013b)	[-100, 100]	7 + 12 + 1 = 20
Schwefel's 2.2 (Jamil and Yang, 2013)	[-100, 100]	7 + 12 + 1 = 20
Rastrigin (Liang et al., 2013a; Liang et al., 2013b)	[-5.2, 5.2]	3 + 17 + 1 = 21
Schwefel's 2.26 (Jamil and Yang, 2013)	[-500, 500]	9 + 16 + 1 = 26
Michalewicz (Jamil and Yang, 2013)	[0, π]	2 + 19 + 1 = 22
Styblinski-Tang (Jamil and Yang, 2013)	[-100, 100]	3 + 25 + 1 = 29
Happy Cat (Liang et al., 2013a)	[-5, 5]	3 + 18 + 1 = 22
Griewank (Liang et al., 2013a)	[-600, 600]	10 + 16 + 1 = 27
Ackley (Liang et al., 2013a; Liang et al., 2013b)	[-32, 32]	6 + 16 + 1 + 23
Rosenbrock (Liang et al., 2013a; Liang et al., 2013b)	[-2.048, 2.048]	2 + 22 + 1 = 25
Expanded Schaffer's F6 (Liang et al., 2013a)	[-100, 100]	7 + 12 + 1 = 20
Expanded Griewank's plus Rosenbrock's (Liang et al., 2013a)	[-10, 10]	4 + 16 + 1 = 21
Hybrid Function – 1: Bent Cigar + Rastrigin + Schwefel's 2.26	[-100, 100], [-5.2, 5.2], [-500, 500]	9 + 17 + 1 = 27
Hybrid Function – 2: Schwefel's 2.2 + Rastrigin + Griewank	[-100, 100], [-5.2, 5.2], [-600, 600]	10 + 17 + 1 = 28
Hybrid Function – 3: Rosenbrock + Griewank + Discus	[-2.048, 2.048], [-5.2, 5.2], [-600, 600]	10 + 22 + 1 = 33
Rotated Rastrigin Function	[-5.2, 5.2]	3 + 17 + 1 + 21
Shifted Rastrigin Function	[-100, 100]	7 + 16 + 1 = 24
Rotated Griewank Function	[-600, 600]	10 + 16 + 1 = 27
Shifted Griewank Function	[-100, 100]	7 + 16 + 1 = 27
Rotated Ackley Function	[-32, 32]	6 + 16 + 1 = 24
Shifted Ackley Function	[-100, 100]	7 + 16 + 1 = 24

[1] http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/Forms/AllItems.aspx

4.1 Comparison of Optimal Solutions and Convergence

In this section, we analyze the performances of SGA, TRGA and hGRGA in terms of optimal solution quality and convergence speed. Several studies have been conducted on the optimal setup of the parameters involved in GA (Digalakis and Margaritis, 2002; Hoque, 2015). We set the number of chromosomes (N) equal to 200 and keep the elite probability (E_p), crossover probability (C_p) and mutation probability (M_p) as 10%, 80% and 5%, respectively. Additionally, the twin removal (TR) operator is applied with the chromosome correlation factor (CCF) decreased from 100% to 80% with a rate (∂CCF) of 0.015% in each iteration. For hGR, we start with a rate (r^{hGR}) equal to 0.1. Therefore, 10% of the less functional genes are treated and replaced by the best gene. Upon improvement we increase the rate (∂r^{hGR}) by 5%. We ran the three GA variants including the proposed hGRGA on 15 benchmark test functions in their base form and 3 hybrid test functions for 20 times with 2000 generations (or epochs) each time. We report the best (minimum) and mean function value (best performance in bold face) found out of 20 runs along with the standard deviation (*std.*) and p-values in **Table 6**. As all the algorithms considered here are the variants of GA, we analyze the converge process in terms of the GA generations/evaluations/epochs needed to converge. For that, we record the average number of epochs required to achieve the minimum function value in the 20 runs (best performance is underlined in **Table 6**). Furthermore, the convergence process as generation progresses is illustrated in **Figure 4**. The number of genes in each chromosome (number of variables in a solution or dimensions of the search space) denoted by d is set to 10 for the results in **Table 6**. We implemented the algorithms and conducted simulations using MATLAB R2013a. The code is available in the supplementary material.

For two type I unimodal functions, all three algorithms could locate the unique global minima, $f_1(\boldsymbol{x}^*) = 0 = f_2(\boldsymbol{x}^*)$ (**Table 6**). However, these functions are easier relative to the other functions in the set as their variables are separable and can be independently optimized. Therefore, the hGR operator provides a deterministic advantage as discussed in **Section 3** by locating the best variable (or gene) and distributing it to the full solution vector (or chromosome) that causes simultaneous optimization of all the variables. These unimodal functions are tested primarily to assess the exploitation and convergence speed of the algorithms in terms of the number of epochs required to converge. The hGRGA converged about 82% and 85% faster than SGA and TRGA for f_1 and f_2, respectively (**Table 6**). These steeper descents are also visible from **Figure 4 (*i*) and (*ii*)**.

Type II functions are unimodal and nonseparable and test the algorithm's ability to handle high numbers of interrelated variables as well as the convergence speed. None of the algorithms could reach the global minima for f_3 and f_4, however hGRGA provides better results for both of the functions compared to SGA and TRGA (**Table 6**). The No-Free-Lunch (NFL) theorem states that it is only possible to develop a promising global optimization technique for a class of problems, and no universal best performing optimization algorithm can be theoretically designed. Therefore, it is worthwhile to develop new evolutionary algorithms as these algorithms have the advantage of being easy to implement and give effective solutions where deterministic algorithms fail. The convergence processes of hGRGA and TRGA for f_3 closely follow the same pattern (**Figure 4 (*iii*)**). However, hGRGA demonstrated superior convergence quality for f_4 compared with both SGA and TRGA (**Figure 4 (*iv*)**). For f_5, hGRGA converged almost straightway to the global minima, $f_5(\boldsymbol{x}^*) = 0$, in 84.07% (and 86.35%) fewer mean epochs than those of TRGA (and SGA) (**Table 6**).

Table 3: Numerical performance of SGA, TRGA and hGRGA on 18 benchmark test functions.

		Function value				Epoch
		best	*mean*	*std.*	*p-value*	(*mean*)
Type I: unimodal and separable						
f_1	hGRGA	0	**0**	0	~	<u>31.45</u>
	TRGA	0	**0**	0	~	170.95
	SGA	0	**0**	0	~	176.25

f_2	hGRGA	0	**0**	0	~	25.85
	TRGA	0	**0**	0	~	174.15
	SGA	0	**0**	0	~	172.65
Type II: unimodal nonseparable						
f_3	hGRGA	0.0220	**11.5021**	5.9948	5.83e-08	1728.3
	TRGA	5.7535	11.7553	4.1417	9.99e-11	1688.4
	SGA	6.8436	28.9594	15.5858	9.49e-08	1833.7
f_4	hGRGA	0.3423	**399.01**	393.99	2.28e-04	726.85
	TRGA	356.1891	1251.6	606.52	1.89e-08	1109.1
	SGA	725.6919	3938.3	1343.1	5.71e-11	1014.7
f_5	hGRGA	0	**0**	0	~	25.75
	TRGA	0	**0**	0	~	161.65
	SGA	0	**0**	0	~	188.70
Type III: multimodal and separable						
f_6	hGRGA	0	**0**	0	~	33.40
	TRGA	0.9950	3.5837	1.5953	4.89e-09	632.20
	SGA	0.9952	4.2306	1.4757	8.40e-11	195.90
f_7	hGRGA	1.27e-04	**0.0601**	0.2645	0.3227	65.80
	TRGA	129.309	309.905	110.289	1.18e-10	1410.6
	SGA	308.918	780.969	252.393	2.25e-11	422.70
f_8	hGRGA	-9.6595	**-9.6078**	0.0503	6.07e-68	1540.2
	TRGA	-9.6598	-9.5674	0.0822	1.09e-61	1606.7
	SGA	-9.6550	-9.5413	0.1282	4.72e-56	1642.3
f_9	hGRGA	-391.662	**-391.662**	1.1e-12	2.9e-278	518.90
	TRGA	-391.661	-391.560	0.1202	1.99e-68	1622.9
	SGA	-391.633	-391.379	0.1636	7.03e-66	1437.2
Type IV: multimodal and nonseparable						
f_{10}	hGRGA	0	**0.0928**	0.0605	1.53e-06	288.75
	TRGA	0.1158	0.2889	0.1008	8.39e-11	242.70
	SGA	0.1420	0.2780	0.0859	1.04e-11	131.40
f_{11}	hGRGA	0	**0**	0	~	57.45
	TRGA	0.1299	0.4884	0.3967	2.60e-05	364.20
	SGA	0.0426	0.4896	0.4659	1.56e-04	363
f_{12}	hGRGA	8.8e-16	**8.8e-16***	0	0	32.35
	TRGA	8.8e-16	0.8915	1.0378	0.0011	1062.9
	SGA	8.8e-16	1.3127	1.0121	1.38e-05	1174
f_{13}	hGRGA	0	**0.0433**	0.1568	0.2317	581.70
	TRGA	0.3201	14.9310	22.1397	0.0071	1328.3
	SGA	1.03e-10	12.5631	18.9428	0.0079	1614.3
	hGRGA	0	**0.009**	0.0441	~	77.6

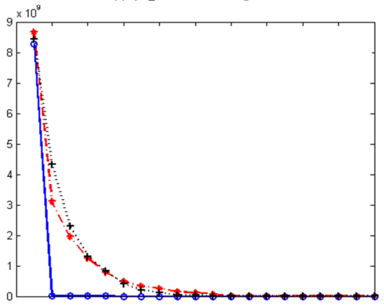
f_{14}	TRGA	0.3587	1.1864	0.4825	1.12e-09	1307.6
	SGA	0.3757	1.6178	0.4794	4.95e-12	547.75
f_{15}	hGRGA	9.92e-09	**0.4986**	0.5661	8.82e-04	349.90
	TRGA	0.5773	2.3420	1.4656	8.59e-07	805.90
	SGA	0.2745	2.4656	1.6061	1.50e-06	412.65
			Type V: hybrid			
f_{16}	hGRGA	778.1345	**1070**	172.6945	9.05e-27	390.05
	TRGA	1211	1289.8	44.5091	1.83e-29	1009
	SGA	1207	1318.2	63.0958	7.93e-17	913.3
f_{17}	hGRGA	0	**0**	0	~	48.05
	TRGA	0.6659	1.7936	0.6104	5.51e-11	365.4
	SGA	0.3211	2.017	0.999	2.66e-08	274.5
f_{18}	hGRGA	1.1309	1.4272	0.1937	3.15e-18	405.2
	TRGA	2.85e-11	**0.8432**	0.8064	1.65e-04	465.4
	SGA	2.85e-11	1.0871	1.5435	0.0034	370.85

~ indicates undefined *p*-value because of zero variance in the function values.
The best mean function value is highlighted by bold.
The fastest convergence in terms of number of epoch is underlined.
* The theoretical global minima of Ackley function is moved to 8.8e-16 from 0 with 4 digit precision of π due to numerical error.

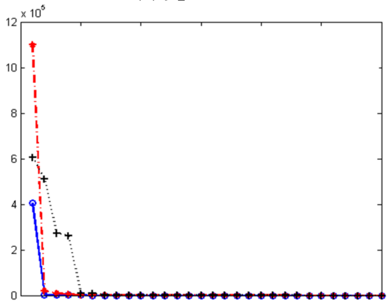
The ten multimodal functions of type III and IV are the most challenging functions that endorse the algorithm's strength of avoiding the local traps and obtaining the global minima. Rastrigin (f_6) and Schwefel 2.26 (f_7) functions have large number of local minima in their corresponding symmetrical and asymmetrical landscapes. Both SGA and TRGA converged prematurely in the local minima for f_6 and f_7. To compare, hGRGA successfully found the global minima ($f_6(x^*)$ = 0) for f_6 and converged very close to the global minima ($f_7(x^*)$ = 0) for f_7 (**Table 6**). Moreover, the convergences of hGRGA for these functions were considerably faster (**Table 6**), specifically no less that about 82% and 84% for f_6 and f_7 than two other GA variants, respectively. The similar effectively superior performance from hGRGA is also achieved for f_9 with global minima, $f_9(x^*)$ = -391.66. **Figure 4 (*vi*), (*vii*)** and **(*ix*)** shows that the proposed hGRGA get near to the global minima almost instantly for f_6, f_7 and f_9. The Michalewicz (f_8) function has a complicated landscape with steep valleys and the global minima residing in a small area. Moreover, the global minimum ($f_8(x^*)$ = -9.6601) occurs for different values of different variables. The hGRGA performed better with 5.24% mean error compared to both the SGA and the TRGA which resulted 11.89% and 9.27% mean error, respectively. The better convergence progress is also visible in **Figure 4 (*viii*).**

For all six nonseparable functions with multimodal search space, SGA and TRGA were attracted and trapped in the local minima. For a relatively new and complex Happy Cat ($f_{10}(x^*)$ = 0) function, hGRGA performed better than SGA and TRGA in terms of mean function value (**Table 6**). SGA took the lowest minimum number of epochs to converge; however it converged to a local minima. To compare, hGRGA found the global minima. Rosenbrock is a classical test function with its global minima ($f_{13}(x^*)$ = 0) inside a sharp and narrow valley. SGA and TRGA resulted mean distances of 12.56 and 14.93 respectively from the global minima, whereas hGRGA converged to 0.0433 and provided faster convergence as well (**Table 6**). The convergence patterns given by SGA and TRGA for f_{10} and f_{13} were similar (**Figure 4 (*x*) and (*xiii*)**) and corresponding patterns provided by hGRGA were far better. For Griewank ($f_{11}(x^*)$ = 0), Ackley ($f_{12}(x^*)$ = 0) and expanded Scaffer's F6 ($f_{14}(x^*)$ = 0) functions, hGRGA reached the global minima (**Table 6**). Moreover, the convergence speed of hGRGA exceeds the other two algorithms by no less than 84.17%, 96.95% and 88.83% for f_{11}, f_{12} and f_{14} respectively. TRGA provided better search progress for these functions than SGA (**Figure 4 (*xi*), (*xii*)** and **(*xiv*)**), whereas hGRGA clearly outperformed the both. The f_{15} is composed of both Rosenbrock and Griewank function. The hGRGA could reach 78.71% and 79.77% closer to the global minima ($f_{15}(x^*) = 0$) than SGA and TRGA (**Table 6**) as well as took 56.58% and 15.20% lower number of epochs than SGA and TRGA (**Table 6** and **Figure 4 (*xv*)**).

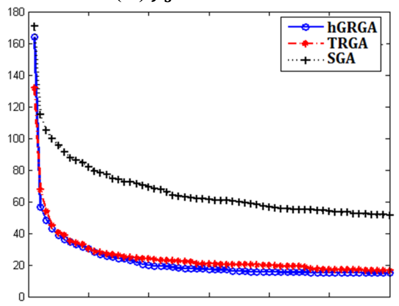
(*i*) f_1: *Bent Cigar*



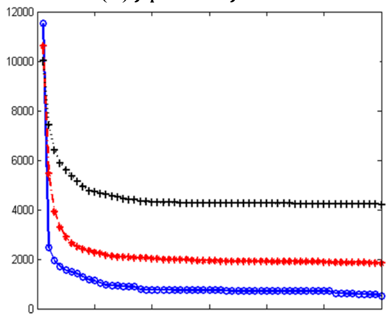
(*ii*) f_2: *Discus*



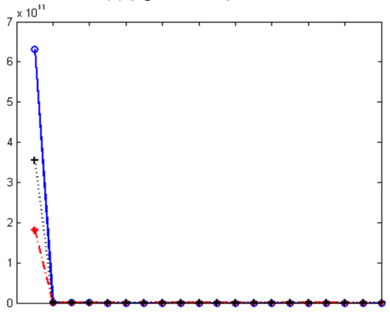
(*iii*) f_3: *Zakharov*



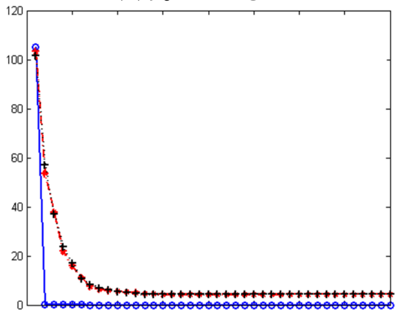
(*iv*) f_4: *Schwefel* 1.2



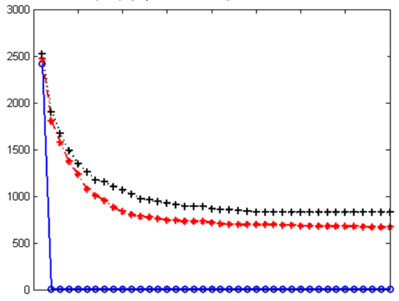
(*v*) f_5: *Schwefel* 2.2



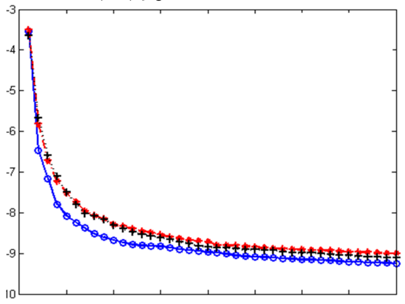
(*vi*) f_6: *Rastrigin*



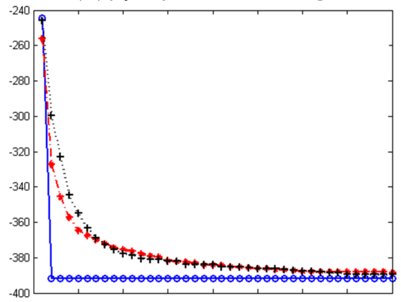
(*vii*) f_7: *Schwefel* 2.26



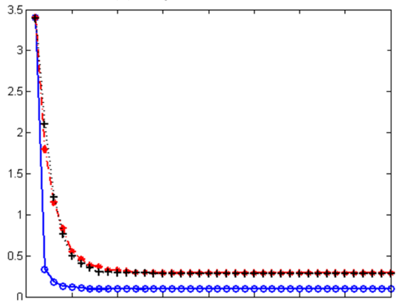
(*viii*) f_8: *Michalewicz*



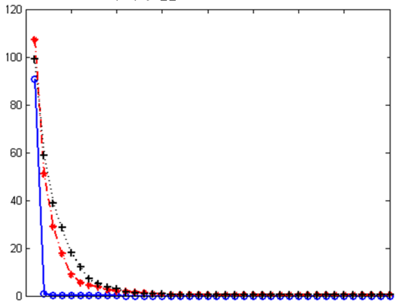
(*ix*) f_9: *Styblinski − Tang*



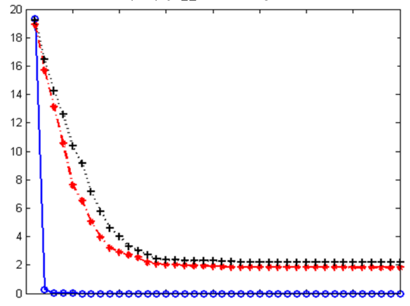
(*x*) f_{10}: *Happy Cat*



(*xi*) f_{11}: *Griewank*



(*xii*) f_{12}: *Ackley*



(*xiii*) f_{13}: *Rosenbrock* (*xiv*) f_{14}: *Expanded Scaffer's F6* (*xv*) f_{15}: *Griewank plus Rosenbrock*

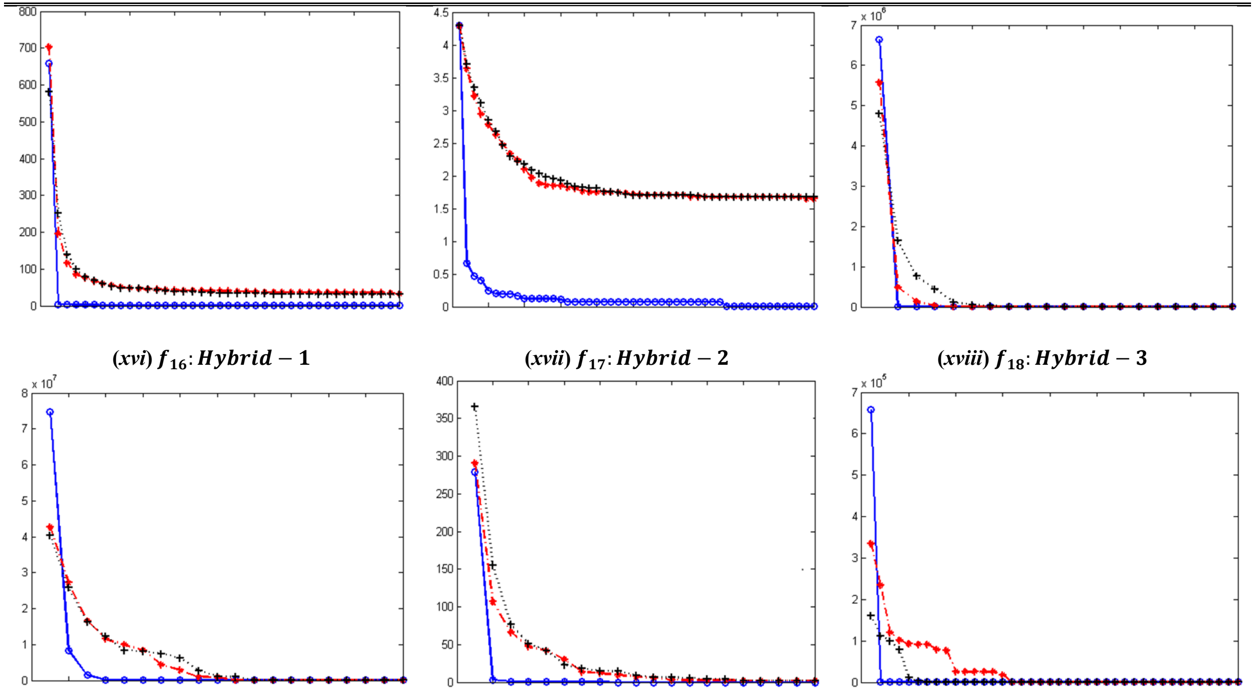


Figure 4: Mean convergence process of SGA, TRGA and hGRGA for 18 test functions with d = 10. The *x*-axis and *y*-axis indicate the epoch and the average function value out of 20 times at that epoch, respectively. Performances of SGA, TRGA and hGRGA are shown by dotted line (*black, plus marker*), dash-dot line (*red, asterisk marker*) and solid line (*blue, circle marker*), respectively.

For the three hybrid functions, random subsets of the variables were fed into different component functions that can make the dimensions of the search space completely unrelated. For f_{16}, hGRGA went approximately 17% and 18.8% closer to the global minima than TRGA and SGA; whereas converged 61.3% and 57.3% faster than TRGA and SGA respectively (**Table 6**). The search process of hGRGA throughout evolution was also promising than other algorithms (**Figure 4 (*xvi*)**). The proposed algorithm successfully found the global minima for Hybrid – 2 function ($f_{17}(x^*) = 0$), on the other hand SGA and TRGA failed. In addition, hGRGA outperformed SGA and TRGA in terms of the mean epoch required to converge by no less than 82.5% (**Table 6** and **Figure 4 (*xvii*)**). For third Hybrid function, average performances of SGA and hGRGA were close; whereas TRGA found the best local minima (**Table 6**). However, **Figure 4 (*xviii*)** shows that the local minima found by hGRGA at same epoch was relatively better than those obtained by SGA and TRGA.

4.2 Assessment of Scalability

We discussed in **Section 3** that the hGR operator is dynamically adaptable to different number of variables (or dimensions) to extract the maximum information out of a chromosome. However, the available information can be limited and depends on the chromosome's current schema pattern. The hGRGA can control the number of gene replacement with respect to the value of d by **Equation 1**, which makes the algorithm robust to variable dimension value. To spotlight this strength of our algorithm, we applied SGA, TRGA and hGRGA on selected subset of test functions with dimension values 50 and 100 in addition to 10. This reduced test set consists of 12 functions including two functions from each of type I, II, III and three functions from each of type IV and V. **Figure 5** shows the box plots of the number of epoch or generation required to converge by three algorithms for 10, 50 and 100 variables, iterated for 20 times with 2000 epochs for these 12 functions. The boxes are labeled with the mean function value achieved.

For type I unimodal functions f_1 and f_2, SGA and TRGA found the global minima with d values 10 and 50. Both SGA and TRGA failed to converge for 100 variables. Moreover, the box plot (**Figure 5 (*i*)-(*ii*)**) shows that SGA and TRGA require higher number of generations as the dimension value increases. To compare, hGRGA gave consistent performance for all dimension values. The similar promising performance by hGRGA was noticed for type II function f_5 with nonseparable variables (**Figure 5 (*iv*)**). All three algorithms converged into local minima for Zakharov function (f_3) using large number of epochs for all dimensions (**Figure 5 (*iii*)**). Therefore, the benefit of hGR was not sufficient to fast converge into the global minimum for Zakharov function; yet the quality of the local optimum found by hGRGA is better for all dimensions. Therefore, f_3 is an instance where the hGR gave *slow impact* (discussed in Section 3).

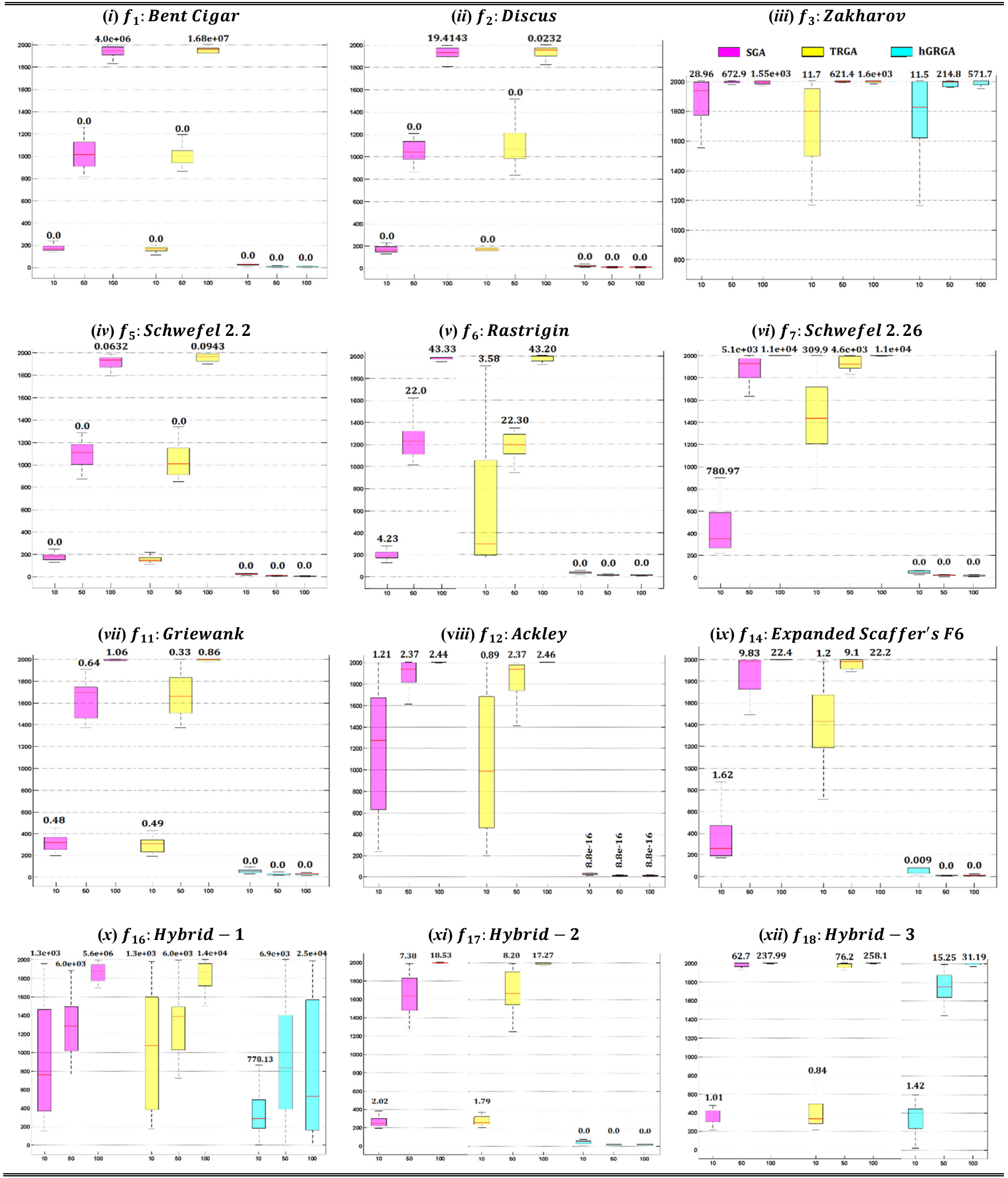


Figure 5: Box plot of number of epoch required to converge by SGA (*magenta*), TRGA (*yellow*) and hGRGA (*cyan*) for d = 10, 50 and 100. The x-axis and y-axis indicate the dimension and statistics of required epoch, respectively. Each box shows the mean epoch required to converge by a *red* straight line within it and is labeled by the corresponding mean function value achieved.

For the multimodal functions with separable variables (f_6 and f_7) in **Figure 5 (*v*)-(v*i*)** and with nonseparable variables (f_{11}, f_{12} and f_{14}) in **Figure 5 (*vii*)-(*ix*)**, SGA and TRGA could not converge for any of the dimensions. Moreover, SGA and TRGA resulted higher (worse) mean functional values achieved with greater number of required epochs as d increases. On the other hand, hGRGA converged into the theoretical global minima for all the five function in all d values except for f_{14} ($d = 10$). Furthermore, hGRGA took approximately similar number of generations to converge irrespective of d values (**Figure 5 (*v*)-(*ix*)**). This result exhibits that hGR can determine the optimal number of gene replacement required depending on the d values (**Equation 1**) and dynamically adapts hGRGA with the scale of underlying problem to result *fast impact* for all dimension values.

In case of hybrid functions f_{16}, the solution quality decreases with the increase of dimension for all three algorithms. However the mean epoch required to converge is lower for hGRGA compared to those of SGA and TRGA for all dimensions (**Figure 5 (*x*)**). For f_{17}, hGRGA consistently found global minima with similar number of GA evaluations (**Figure 5 (*xi*)**) irrespective of the scale of dimension unlike other two algorithms (*fast impact*). All three GAs gave comparable performances for hybrid function f_{17} in terms of required number of epoch to converge; however hGRGA resulted considerably better quality solution for higher dimensions ($d = 50, 100$) in terms of functions value than those given by SGA and TRGA (*slow impact*).

4.3 Effect of hGR Operator

Here we intend to quantify the advantage of using the new hGR operator in hGRGA. For this, we compute the fitness of the elites in the 1st epoch before and after applying hGR operator. In our setup with 200 chromosomes in the population and 10% elite probability, we have 20 elites. **Figure 6** shows the function values of the 20 elites before and after applying hGR with 100 variables (dimension) for the reduced subset of 12 functions mentioned in **Section 4.2**.

Figure 6 (*i*)-(*iv*) for four unimodal functions highlight the different amount of benefits to different elites provided by hGR operator. These showcases justify our earlier discussion in **Section 3** that different elite may allow dissimilar amount of gene replacement depending on its current schema. It is therefore important to understand how these improvements of elites can facilitate the overall convergence process. Even before the application of hGR, the elites are the subset of chromosomes in the population with relatively higher fitness values. Now when the hGR operator boosts up the fitness of some elites even further, the fitness proportionate selection becomes reasonably biased towards those improved elites. Thereafter, they can participate in crossover with higher chance to generate better offspring that can accelerate the exploitation as well as the convergence. The improvement of elites after applying hGR for f_1, f_2 and f_5 are visible in **Figure 6 (*i*), (*ii*)** and **(*iv*)** and the effects these improvements as discussed above are explicit in **Figure 5 (*i*), (*ii*)** and **(*iv*)** that shows significantly lower numbers of epoch were required to achieve far better solutions by hGRGA. In contrast, **Figure 5 (*iii*)** shows that hGRGA required similar number of epoch to converge for Zakharov function when compared to other GAs. However, the superior solution quality (mean function value) achieved by hGRGA is notable in **Figure 5 (*iii*)**. Therefore, we assume that though hGR improved the fitness values of the elites (**Figure 6 (*iii*)**), the resulting effect was not enough to converge in the global optimum. Therefore, hGRGA continued to search throughout the available generations, however could achieve better local optimum out of similar number of generations as SGA and TRGA.

The hGR yielded (93.5 – 99.9)% and (94.5 – 99.9)% improved function values (decreased for minimization problem) for different elites in case of two separable multimodal functions, Rastrigin (f_6) and Schwefel 2.26 (f_7) respectively, shown in **Figure 6 (*v*)-(*vi*)** . This also focuses the reasoning behind the significant quicker convergence of hGRGA over SGA and TRGA for these functions (**Figure 5 (v)-(*vi*)**). Due to this significant improvement in the elite's fitness, it is likely for the highly fit chromosomes to be selected for crossover by the fitness proportionate selection algorithm. Subsequently, the improved parent chromosomes produce offspring with higher fitness after crossover.

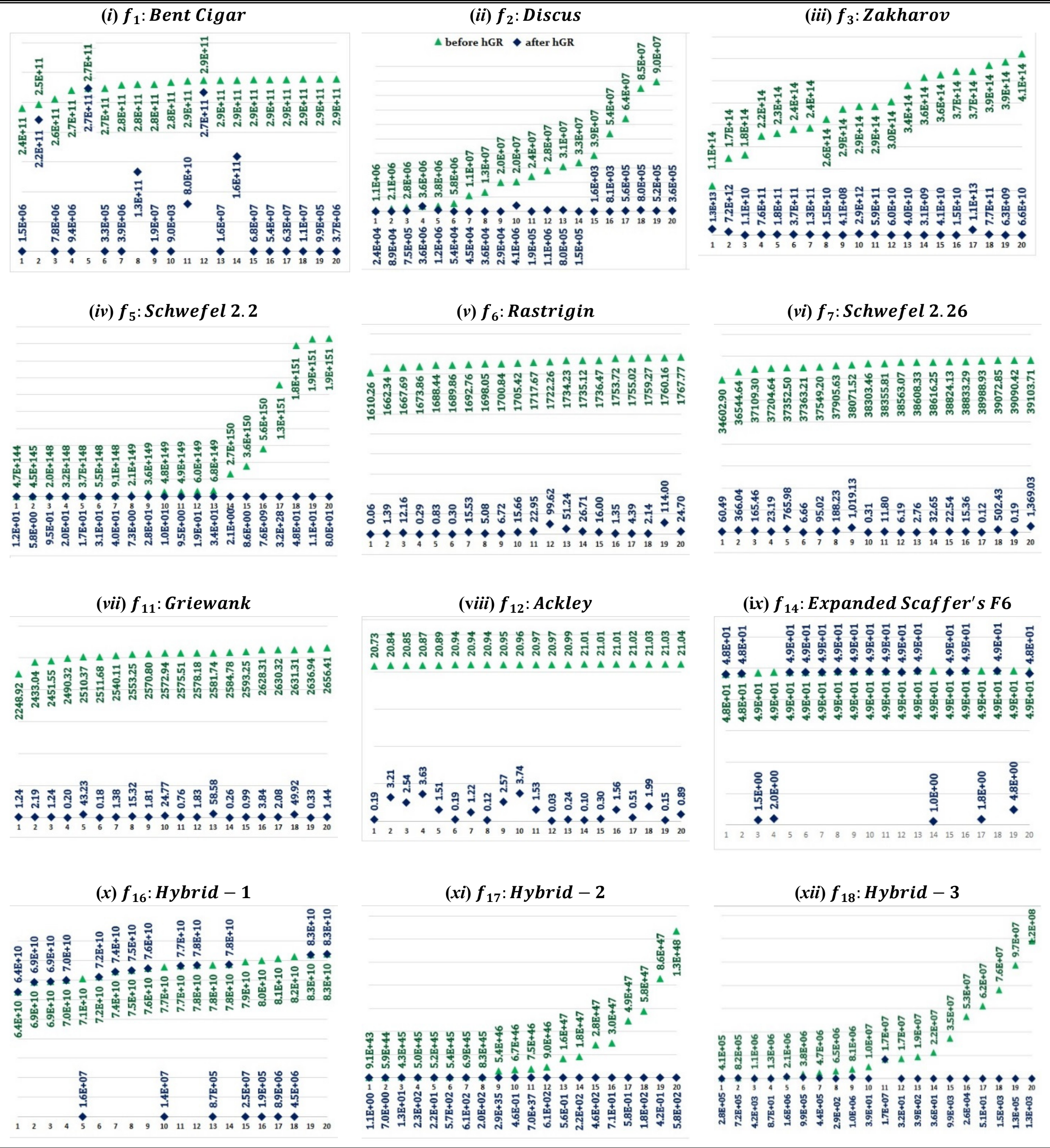


Figure 6: Comparison of 20 elite's fitness before applying hGR (*green triangle, fitness values are written in green*) and after applying hGR (*blue diamond, fitness values are written in blue*) operator for number of variables (dimension), d = 100. The *x*-axis shows the elite index and *y*-axis indicates the fitness of the elite, respectively.

The gene replacement contributed (97.7 – 99.9)% and (82.6 – 99.0)% betterment to all the elite's fitness for Griewank and Ackley function (**Figure 6 (*vii*)-(*viii*)**). However the scenario in **Figure 6 (*ix*)** is slightly different for another nonseparable multimodal function, Expanded Schaffer's F6 (f_{14}). Here the hGR could notably refine only 5 elites with (90.3 – 97.9)% and moderately furbish 3 elites with (0.2 – 1.87)% additional fitness values. As hGR searches for a good gene within the respective elites to be treated, it is possible to have no such local good schema in that elite. This is reasonable according to the *schema theorem* (Goldberg, 1989; Holland, 1975/1992) as well that states that GA works by emphasizing and recombining good schemas that confers higher fitness to a good solution. Therefore, if the current bit pattern of an elite does not any such schema, hGR may not give any benefit by distributing it. However, it is interesting to note that the convergence or exploitation speed of hGRGA for this function (f_{14}) with 100 dimension are still promising (**Figure 5 (*ix*)**). It is due to the intrinsic characteristics of GA to ensure the survival of the fittest. Therefore, it is only sufficient to boost up some elites to supervise the fitness proportionate selection and crossover operator in hGRGA.

Figure 6 (*x*)-(x*ii*) focuses the variable advantages of using gene replacement for hybrid functions. For hybrid – 2 function (**Figure 6 (*x*)**), the advantage was enough to have *fast impact* on the convergence (**Figure 5 (*x*)**). For hybrid – 1 and 3 functions, the hGR could moderately improve 30% and 50% of the elite chromosomes respectively. This result is possible for the aforementioned reason as well that an elite may not have a good schema in it to be utilized through hGR. Though hGRGA could achieve better solution quality due to these improvements, the impact was not enough to achieve global optimum or fast convergence.

4.4 Comparison of hGRGA with other state-of the-art evolutionary algorithms

To evaluate the effectiveness of hGRGA, we compare its performance with two well-known evolutionary algorithms, Differential Evolution (DE) (Price et al., 2006; Storn and Price, 1997) and Artificial Bee Colony (ABC) Algorithm (Karaboga, 2005b; Karaboga and Basturk, 2007; Karaboga and Basturk, 2008) on the full test set including 24 functions (**Table 4** and **5**). The source codes are obtained from publicly available links[2]. DE is a population-based stochastic search technique that applies mutation, crossover and selection on real values. Here we consider the classical DE with mutation strategy DE/rand/1 and the values of the control parameters such as population size, scaling factor and crossover rate are kept 10 × *d* (number of variables), 0.5 and 0.3 respectively (Ronkkonen et al., 2005; Storn and Price, 1997; Vesterstrom and Thomsen, 2004). ABC is a swarm based evolutionary algorithm (Srinivasan and Seow, 2003)that imitates the foraging behavior of intelligent honeybees. The control parameters for ABC algorithm such as colony size, number of food sources and limit are set to 125, half of the colony size and 100 respectively (Karaboga and Basturk, 2007). We keep the same parameter values for hGRGA as mentioned in **Section 4.1**. Note that, **Table 5** shows that the search domains of three different constituent functions of the hybrid functions are different. We used this while comparing three GA variants. For the experiments of this section, we scaled the search domains of component functions to a common rage [-100, 100] except for Schwefel 2.26 function.

In this experiment, we tested the three algorithms on 30-dimensional problems, thus the number of variables or dimensions, *d* is equal to 30. Each simulation is repeated for 20 runs. We report both mean with standard deviation (stdev) and median function values out of the 20 runs in **Table 7** as the mean function value has a tendency to emphasize large values and median can overcome this limitation (Auger, January 14, 2016). To compare the cost of three different algorithms appropriately, we did not control the algorithms by maximum number of iterations or generation. Rather, we allow a maximum *number of function evaluations* (NFE) equal to $(d \times 1e + 4)$. Thus, we stop a simulation if the NFE exceeds $(d \times 1e + 4)$ or the function value is less than or equal to the known global optimal value with a tolerance, $(f(x^*) + 1e - 10)$. A simulation or run is considered as successful if the algorithm can achieve a function value no worse than $f(x^*) + 1e - 10$ within no greater than $d \times 1e + 4$ number of function evaluations. We report the rate of successful run called success rate (SR) and mean NFE required in successful runs out of 20 runs in **Table 7**. Note that, the mean NFE required is only meaningful to document and use for comparison when an algorithm can obtain non-zero success rate while optimizing a function. Therefore, mean NFE entry has a dummy character ('-') in case of zero successful rate.

Table 7. Performance comparison among DE ABC and hGRGA on 24 30-dimensional test functions.

[2] DE source code link: http://www1.icsi.berkeley.edu/~storn/code.html
ABC source code link: http://mf.erciyes.edu.tr/abc/

	DE (Storn and Price, 1997)				ABC (Karaboga and Basturk, 2007)				hGRGA (This work)			
	mean $f(\boldsymbol{x})$ [stdev]	*median* $f(\boldsymbol{x})$	SR (%)	*mean* NFE	*mean* $f(\boldsymbol{x})$ [stdev]	*median* $f(\boldsymbol{x})$	SR (%)	*mean* NFE	*mean* $f(\boldsymbol{x})$ [stdev]	*median* $f(\boldsymbol{x})$	SR (%)	*mean* NFE
f_1	1.11E-05 [2.00E-06]	1.00E-05	0	-	**0** **[0]**	**0**	**100**	153946.9	**0** **[0]**	**0**	**100**	**20771.5**
f_2	**0** **[0]**	**0**	**100**	291675	**0** **[0]**	**0**	**100**	121079.8	**0** **[0]**	**0**	**100**	**19904.6**
f_3	112.598 [**11.97**]	109.099	0	-	204.9696 [23.54]	207.552	0	-	**95.141** [13.24]	**94.190**	0	-
f_4	20385.0 [2430.0]	20161.3	0	-	**5446.3** **[1012.3]**	**5487.8**	0	-	15878.3 [8559.4]	17967.0	**5**	**32103**
f_5	6.00E-06 [1.00E-06]	6.00E-06	0	-	**0** **[0]**	**0**	**100**	34388	**0** **[0]**	**0**	**100**	**19400.05**
f_6	71.406 [4.63]	72.079	0	-	**0** **[0]**	**0**	**100**	146008.8	**0** **[0]**	**0**	**100**	**23913.7**
f_7	**3.82E-04** **[0]**	**3.82E-04**	0	-	**3.82E-04** **[0]**	**3.82E-04**	0	-	1.79E-01 [7.94E-01]	**3.82E-04**	0	-
f_8	**-9.66015** **[0]**	**-9.66015**	**100**	**35105**	-3.994 [0.48]	-3.977	0	-	-8.919 [0.29]	-8.932	0	-
f_9	**-1174.9** **[0]**	**-1174.9**	**100**	236820	**-1174.9** **[0]**	**-1174.9**	**100**	24377.1	**-1174.9** **[0]**	**-1174.9**	**100**	**7227.35**
f_{10}	0.422 [0.04]	0.430	0	-	0.343 [0.03]	0.339	0	-	**0.092** **[0.06]**	**0.090**	**5**	**48169**
f_{11}	1.51E-10 [9.5E-11]	1.02E-10	45	2.98E+05	**0**	**0**	**100**	1.38E+05	**0** **[0]**	**0**	**100**	**4.13E+04**
f_{12}	1.00E-06 [0]	1.00E-06	0	-	**0** **[0]**	**0**	**100**	2.04E+05	**0** **[0]**	**0**	**100**	**2.34E+04**
f_{13}	25.150 [0.16]	25.203	0	-	5.406 [3.82]	4.755	0	-	**0.0078** **[0.014]**	**0.0013**	**5**	**88897**
f_{14}	8.352 [0.34]	8.365	0	-	0.707 [**0.148**]	0.694	0	-	**0.547** [1.57]	**0**	**85**	**21766.58**
f_{15}	5.092 [0.50]	5.192	0	-	0.723 [0.23]	0.716	0	-	**1.96E-04** **[4.3E-04]**	**5.0E-06**	**5**	**88834**
f_{16}	3.19E-02 [1.76E-02]	2.53E-02	0	-	**1.27E-04** **[2.0E-09]**	**1.27E-04**	0	-	9.95E+02 [9.6E+02]	8.06E+02	0	-
f_{17}	0.142 [0.016]	0.143	0	-	1.35E-03 [4.6E-03]	8.65E-11	65	208060.9	**0** **[0]**	**0**	**100**	**39948.5**
f_{18}	5.933 [0.172]	5.977	0	-	**0.200** **[0.159]**	**0.178**	0	-	10.162 [4.73]	8.540	0	-
f_{19}	171.624 [**9.69**]	192.896	0	-	82.065 [13.77]	82.774	0	-	**7.321** [32.74]	**0**	**95**	**36204.73**
f_{20}	**178.127** **[0]**	**178.127**	0	-	**178.127** **[0]**	**178.127**	0	-	218.670 [8.82]	219.168	0	-
f_{21}	1.86E-03 [1.98E-03]	1.53E-03	0	-	1.33E-06 [1.4E-06]	6.68E-07	0	-	**4.27E-10** **[1.9E-09]**	**0**	**95**	**102672.6**
f_{22}	6.47E-09 [4.9E-09]	5.30E-09	0	-	**0**	**0**	**100**	**122636**	2.503 [0.55]	2.432	0	-
f_{23}	2.22E-04 [3.6E-06]	2.22E-04	0	-	1.999 [0.27]	2.057	0	-	**0** **[0]**	**0**	**100**	**27951.5**
f_{24}	20.262 [**2.01E-02**]	20.267	0	-	**12.621** [7.86E+00]	**16.769**	0	**-**	20.266 [9.91E-02]	20.257	0	-

A function value lower than 1e-10 is reported as zero.
Best values are indicated by bold face.
'-' indicates not applicable mean NFE value due to zero success rate.

4.4.1. Comparison in terms of mean and median function value

Here we analyze the performance of DE, ABC and hGRGA in terms of function value only as reported in **Table 7**. The two unimodal and separable functions (f_1 and f_2) were easily optimized to the global minima by all the algorithms, except f_1 for which DE ended up with a function value very close to global optima in every run. We observe three different scenario for three different functions (f_3, f_4 and f_5) of similar properties (unimodal and nonseparable). For f_3 and f_4, all three algorithms converged into local minima, however hGRGA and ABC gave better mean and median values than others for f_3 and f_4 respectively. On contrary, all three algorithms gave comparable optima for f_5 where DE performed slightly worse.

The function f_6, f_7 and f_8 and f_9 (type III) have separable variables, however are multimodal having large number of optima in the search space. For Rastrigin function (f_6), hGRGA and ABC could achieve the global minimum in every run while DE was trapped into a local minima. The results of Schwefel 2.26 (f_7) function is an instance where we observed the flaw involved in the mean or average statistics (Auger, January 14, 2016) at least in this particular application. All three algorithms gave similar median value for f_7 while the mean value given by hGRGA was slightly higher. We found that the function resulted by hGRGA was 3.82E-04 for 75% of the run whereas was 3.55 for only 1 run and 4.08E-03 for 4 runs. Therefore, the mean value emphasized those slightly lower performances for 25% of the runs where median value emphasized the results of 75% of the runs. DE performed the best and found the global optima for f_8. On the other hand, all three algorithms consistently found the minima for f_9.

The six functions f_{10} to f_{15} (type IV) have both of the challenging properties, nonseparability and multimodality. The proposed hGRGA performed quite well on these functions. For f_{11} (Griewank) and f_{12} (Ackley) functions, ABC and hGRGA consistently found the global minima (0) where DE prematurely converged into near-global minima. For Happy cat (f_{10}), Rosenbrock (f_{13}) and Expanded Griewank's plus Rosenbrock function (f_{14}), none of the algoroithms could result the global minimum value as mean or median. However, the corresponding mean and median values given by hGRGA were significantly better than those of DE and ABC for these three functions. For f_{15}, hGRGA could obtain the global minima for 85% of the runs and outperformed DE and AMC both interms of mean and median function values.

The six functions of type V are even more complicated due to their transformation from base forms to hybrid, rotated and shifted form. Hybrid – 1 and 3 functions are so complex that all three algorithms were trapped into the local minima in each run. For both of these functions, DE and ABC algorithm significantly outperformed hGRGA while hGRGA performed better than both DE and ABC for Hybrid – 2 function. For all three rotated function (f_{19}, f_{21} and f_{23}), hGRGA outperformed DE and ABC algorithm. For rotated Rastrigin (f_{19}) and Griewank (f_{21}) functions, hGRGA could reach the global minima for 95% of the run that made a median function value equal to the global minima (0). The corresponding mean values given by hGRGA for f_{19} and f_{21} functions are relatively higher than the medians for being trapped into local minima in 1 run only, yet the resulting mean and median values are far better than DE and ABC. For rotated Ackley function (f_{23}), only hGRGA could achieve the global minimum as mean and median function value, whereas the performance of DE and ABC were reasonable worse. DE and ABC resulted comparable mean and median functions values for shifted Rastrigin (f_{20}) and Griewank (f_{22}) function, while hGRGA gave much lower performances. For shifted Ackley function (f_{24}), hGRGA and DE performed comparably while ABC outperformed both of them.

4.4.2. Comparison in terms of success rate

Here we assess the success rate of the three algorithms (**Table 7**). It is well-accepted that the output of a stochastic process cannot be determined precisely (Lawler, 2006). Thus, the researchers run a stochastic search based evolutionary algorithm on a test function for multiple times and analyze the performance statistically as the algorithm may not be successful in all the runs. Here we say that a run is successful if the corresponding algorithm can achieve a function value no worse than a pre-specified optimum within the allowable number of functions evaluations (NFE). The success rate metric focuses how likely an algorithm can be successful, thus it measures the reliability of an algorithm.

Among five unimodal functions (f_1 to f_5), ABC and hGRGA were consistently reliable for three functions (f_1, f_2 and f_5) with 100% success where DE was found reliable for f_2 only. None of the algorithms could find the global minimum for f_3 in any run and hGRGA alone was 5% reliable for f_4. Out of ten multimodal functions (f_6 to f_{15}), DE was found 100% reliable for two functions (f_8 and f_9) and 45% successful in optimizing Griewank function (f_{11}). However, both ABC and hGRGA were 100% successful so as more reliable on Griewank function. In total, ABC was 100% successful for four multimodal functions, two with separable variables (f_6 and f_9) and two with nonseparable variables (f_{11} and f_{12}). The hGRGA consistently showed 100% reliability for these four functions like ABC. In addition, hGRGA obtained 5%, 5%, 85% and 5% success rate respectively for f_{10}, f_{13}, f_{14} and f_{15} while the success rate of DE and ABC was zero for these functions. Among nine functions of type V (hybrid, rotated and shifted), DE could not obtain the global minimum at any run. The ABC algorithm could solve two functions with non-zero success rate, 65% for Hybrid – 2 (f_{17}) and 100% for shifted Griewank f_{22} function. On the other hand, hGRGA secured 100% success rate for Hybrid – 2 (f_{17}) function and 95% success rate for two rotated functions (f_{19} and f_{21}) and 100% for f_{23}.

In summary, DE, ABC and HGRGA resulted success rate greater than zero for 4, 9 and 16 functions respectively. However, the actual value of the rate can be low. Precisely, DE, ABC and hGRGA were found successful respectively for 69, 173 and 239 runs out of the total 480 runs (each of the 24 functions were run for 20 times).Thus, the overall success rate or reliability of DE, ABC and hGRGA on these 24 test functions were 14.37%, 36.04% and 49.79% respectively.

4.4.3. Comparison in terms of number of function evaluations

In this part, we compare the cost of the three algorithms in terms of the number of function evaluations (NFE) needed to achieve a successful run outlined in **Table 7**. The termination criteria for a simulation were wither the pre-specified optima was achieved or maximum NFEs were exceeded. Therefore, it is only meaningful to compare the NFEs when two or more algorithms under comparison have some successful runs.

The type I (unimodal and separable) functions (f_1 and f_2) were primarily used to test the algorithm's convergence speed. The hGRGA converged into global minimum of f_1 with 86.5% lower NFEs than that of ABC. Additionally, The cost (NFEs) to converge by hGRGA was respectively 93.1% and 83.5% lower than those of DE and ABC for f_2. Similarly, hGRGA achieved the global minima for f_5 (unimodal and nonseparable) in 43.6% lower NFEs than that of ABC. We observed consistent performance by hGRGA on multimodal functions with both separable and nonseparable variables as well. For Rastrigin function (f_6), hGRGA was 83.6% faster while it was 96.9% and 70.4% faster than DE and ABC respectively for Styblinski-Tang function (f_9). Similarly, hGRGA took 86.1% and 70.0% lower NFEs to solve Griewank (f_{11}) function respectively than those of DE and ABC and 88.5% lower than that of ABC in case of Ackley (f_{12}) function. Only one function (f_{17}) of type V is applicable for this comparison on which the mean NFEs required to achieve 100% successful runs by hGRGA was 80.8% lower than that of 65% successful runs by ABC. Thus, we conclude that hGRGA is consistently better than ABC and DE in terms of NFEs (or cost) required to converge which is empirically justified for this particular application.

4.4.4. Comparison of overall performance

In this section, we perform an analysis of the overall performances given by DE, ABC and hGRGA by quantifying the *Success Performance* (SP) used to measure the expected number of function evaluations to achieve a per-specified function value (Hansen, May 4, 2006). Thus after, we plot the empirical distribution of functions with normalized success performance in **Figure 7**. We compute the SP and normalized SP ($\widehat{SP}$) by following equations:

$$SP = mean\ NFEs\ of\ successful\ runs \times (total\ runs\ (= 20)/number\ of\ successful\ runs) \quad (3)$$

$$\widehat{SP} = SP/SP_{best} \quad (4)$$

Here SP_{best} is the SP of the best algorithm, thus the lowest SP. The results of all functions were used for which at least one algorithm was successful at least in one run. Thus, we considered 18 functions out of 24 to perform this experiment. For six functions ($f_3, f_7, f_{16}, f_{18}, f_{20}$ and f_{24}), none of the algorithm could achieve the global optima within the allowable NFEs. Small values of SP and large values of cumulative frequency of functions for a SP value, therefore higher area under the distribution curve is preferable. **Figure 7** summarizes the superior performance of hGRGA in terms of SP. As hGRGA outperformed the other two algorithms in terms of NFEs in 17 functions (discussed in Section 4.4.3) out of 18 functions considered to the plot of **Figure 7**, the curve of hGRGA looks far better than those of others. The plots also highlights the ABC algorithm performed better than DE (specifically DE/rand/1) on these 18 functions.

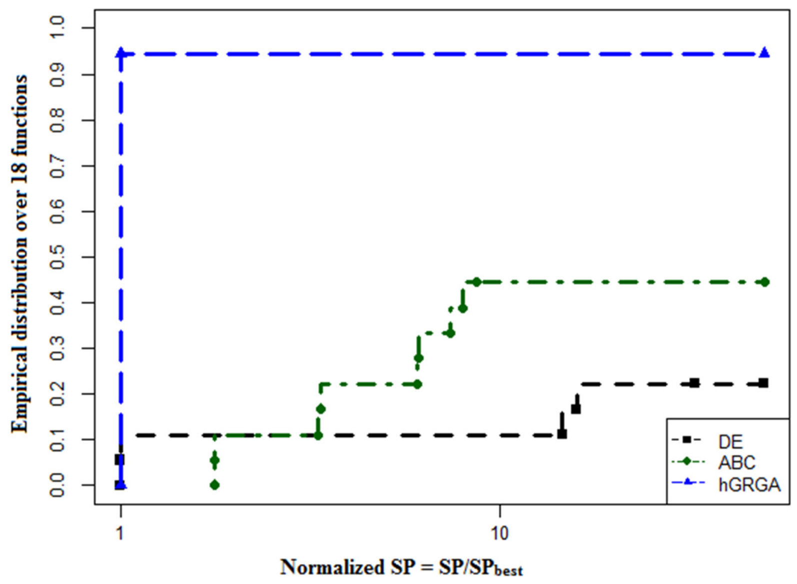


Figure 7: Empirical distribution of normalized success performance of DE, ABc and hGRGA on 18 test functions.

5. CONCLUSION

In this paper, we introduce a novel homologous gene replacement (hGR) operator that is dynamically adaptable to different number of variables. Using the hGR operator, we propose an improved version of popular nature-inspired genetic algorithm (hGRGA) for numerical optimization. The hGRGA algorithm includes four major operators that effectively balances the exploitation (or, intensification) and exploration (or, diversification) tasks. To have a proper trade-off between these two desirable but incompatible features is challenging in developing an optimization algorithm. The hGR operator repairs the elites in schema level, which are preserved by elitism in consecutive generations to avoid the disruption of these schemata and to ensure non-decreasing performance with time. The hGR operator further governs the crossover to strengthen the exploitation. Moreover, we employ twin removal in addition to mutation to make sure appropriate exploration. The resulting hGRGA delivered promising performance both in terms of finding minimum function value and speed of convergence for benchmark test functions having various complex properties. It clearly outperformed two existing variants of GA, namely SGA and TRGA, while showed competitive success performance as well as success rate when compared with other evolutionary techniques such as DE and ABC algorithm. In this paper, we tested the proposed hGRGA for continuous function optimization. GA has been effectively applied for hard combinatorial optimization problems as well. Therefore, it would be interesting to apply hGRGA to solve challenging modern combinatorial optimization as well as discrete engineering optimization problems.

ACKNOWLEDGEMENTS

SI and MTH gratefully acknowledge the Louisiana Board of Regents through the Board of Regents Support Funds, LEQSF (2013-16)-RD-A-19 and LEQSF(2016-19)-RD-B-07.

References

Auger, A., January 14, 2016. Performance Assessment in Optimization. Available: https://www.lri.fr/~auger/teaching-slides/05_PerformanceAssessment.pdf.

Back, T., Hoffmeister, F., Schwefel, H., 1991. A survey of evolutionary strategies. Proceedings of the 4th international conference on genetic algorithms. Morgan Kaufmann publishers, San Mateo, CA, pp. 92-99.

Bao, Z., Zhou, Y., Li, L., Ma, M., 2015. A Hybrid Global Optimization Algorithm Based on Wind Driven Optimization and Differential Evolution. Mathematical Problems in Engineering 2015.

Bayraktar, Z., Komurcu, M., Werner, D. H., 2010. Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics. Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE. IEEE, pp. 1-4.

Belding, T. C., 1995. The distributed genetic algorithm revisited. arXiv preprint adap-org/9504007.

Coello, C. A., 2000. An updated survey of GA-based multiobjective optimization techniques. ACM Computing Surveys (CSUR) 32, 109-143.

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

DeJong, K., 1975. An analysis of the behavior of a class of genetic adaptive systems. Ph. D. Thesis, University of Michigan, Ann Arbor.

Dieterich, J. M., Hartke, B., 2012. Empirical review of standard benchmark functions using evolutionary global optimization. arXiv preprint arXiv:1207.4318.

Digalakis, J. G., Margaritis, K. G., 2002. An experimental study of benchmarking functions for genetic algorithms. International Journal of Computer Mathematics 79, 403-416.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 26, 29-41.

Eshelman, L. J., 2014. The CHC adaptive search algorithm: How to have safe search when engaging. Foundations of Genetic Algorithms 1991 (FOGA 1) 1, 265.

Fateen, S.-E. K., Bonilla-Petriciolet, A., 2014. Intelligent firefly algorithm for global optimization. Cuckoo Search and Firefly Algorithm. Springer, pp. 315-330.

Gandomi, A. H., Alavi, A. H., 2012. Krill herd: A new bio-inspired optimization algorithm. Nonlinear Sci. Numer. Simul. 17, 4831 - 4845.

Goldberg, D. E., 1987. Simple genetic algorithms and the minimal, deceptive problem. Genetic algorithms and simulated annealing 74, 88.

Goldberg, D. E., 1989. Genetic Algorithms in Search. Optimization and Machine Learning, Reading, Massachusetts.

Goldberg, D. E., 1990. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. Complex Systems 4, 445-460.

Goldberg, D. E., 1994. Genetic and evolutionary algorithms come of age. Communications of the ACM 37, 113-120.

Goldberg, D. E., Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms 1, 69-93.

Gong, M., Cai, Q., Chen, X., Ma, L., 2014. Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. Evolutionary Computation, IEEE Transactions on 18, 82-97.

Gorges-Schleuter, M., 1990. Explicit parallelism of genetic algorithms through population structures. International Conference on Parallel Problem Solving from Nature. Springer, pp. 150-159.

Hansen, N., May 4, 2006. Compilation of Results on the 2005 CEC Benchmark Function Set. Available: http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf.

Haupt, R. L., Haupt, S. E., 2004. Practical genetic algorithms. John Wiley & Sons.

Higgs, T., Stantic, B., Hoque, M. T., Sattar, A., 2012. Refining genetic algorithm twin removal for high-resolution protein structure prediction. IEEE Congress on Evolutionary Computation, . IEEE, pp. 1-8.

Holland, J., 1975/1992. Adaptation in natural and artificial systems. Univ. of Michigan Press.

Holland, J., 1992. Genetic Algorithms. Scientific American Journal, 66 - 72.

Hoque, M., Chetty, M., Dooley, L., 2005. Critical Analysis of the Schemata Theorem: The Impact of Twins and the Effect in the Prediction of Protein Folding Using Lattice Model. GSIT, MONASH University.

Hoque, M. T., 2015. Genetic Algorithms based Improved Sampling. Tech. Report TR-2015/4.

Hoque, M. T., Chetty, M., Dooley, L. S., 2007. Generalized schemata theorem incorporating twin removal for protein structure prediction. Pattern Recognition in Bioinformatics. Springer, pp. 84-97.

Hoque, M. T., Chetty, M., Lewis, A., Sattar, A., 2011. Twin removal in genetic algorithms for protein structure prediction using low-resolution model. IEEE/ACM Transactions on Computational Biology and Bioinformatics 8, 234-245.

Iqbal, S., Hoque, M. T., 2016a. A Scalable Gene Replacement Operator for Genetic Algorithm. http://cs.uno.edu/~tamjid/TechReport/hGRGA_TR20161.pdf, TR-2016/1, Computer Science, University of New Orleans, LA, USA.

Iqbal, S., Hoque, M. T., 2016b. (Extended abstract) A Homologous Gene Replacement based Genetic Algorithm. GECCO, Denver, Colorado, USA.

Iqbal, S., Kaykobad, M., Rahman, M. S., 2015. Solving the multi-objective Vehicle Routing Problem with Soft Time Windows with the help of bees. Swarm and Evolutionary Computation 24, 50-64.

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

Jiao, L., Wang, L., 2000. A novel genetic algorithm based on immunity. IEEE Transactions on Systems, Man and Cybernetics, Part A 30, 552-561.

Karaboga, D., 2005a. An idea based on honeybee swarm for numerica optimization. Technical Report TR06, Erciyes University.

Karaboga, D., 2005b. An idea based on honey bee swarm for numerical optimization. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization 39, 459-471.

Karaboga, D., Basturk, B., 2008. On the performance of artificial bee colony (ABC) algorithm. Applied soft computing 8, 687-697.

Kennedy, J., Eberhart, R. C., 1995. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, 1942 - 1948.

Koumousis, V. K., Katsaras, C. P., 2006. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. Evolutionary Computation, IEEE Transactions on 10, 19-28.

Krishnanand, K., Ghose, D., 2005. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE. IEEE, pp. 84-91.

Kühn, M., Severin, T., Salzwedel, H., 2013. Variable Mutation Rate at Genetic Algorithms: Introduction of Chromosome Fitness in Connection with Multi-Chromosome Representation International Journal of Computer Applications 72, 31 - 38.

Lawler, G. F., 2006. Introduction to stochastic processes. CRC Press.

Liang, J.-J., Suganthan, P. N., Deb, K., 2005. Novel composition test functions for numerical global optimization. Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005. IEEE, pp. 68-75.

Liang, J., Qu, B., Suganthan, P., 2013a. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

Liang, J., Qu, B., Suganthan, P., Hernández-Díaz, A. G., 2013b. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 201212.

Maher, B., Albrecht, A. A., Loomes, M., Yang, X.-S., Steinhöfel, K., 2014. A firefly-inspired method for protein structure prediction in lattice models. Biomolecules 4, 56-75.

Michalewicz, Z., 2013. Genetic algorithms + data structures= evolution programs. Springer Science & Business Media.

Mitchell, M., 1995. Genetic Algorithm: An Overview. Complexity 1, 31 - 39.

Muhlenbein, H., 1992. How genetic algorithms really work: I. mutation and hillclimbing. Parallel Problem Solving from Nature 2. B. Manderick. Amsterdam, Elsevier.

Mühlenbein, H., Schlierkamp-Voosen, D., 1993a. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. Evolutionary computation 1, 25-49.

Mühlenbein, H., Schlierkamp-Voosen, D., 1993b. The science of breeding and its application to the breeder genetic algorithm (BGA). Evolutionary Computation 1, 335-360.

Price, K., Storn, R. M., Lampinen, J. A., 2006. Differential evolution: a practical approach to global optimization. Springer Science & Business Media.

Rashid, M. A., Khatib, F., Hoque, M. T., Sattar, A., 2015. An Enhanced Genetic Algorithm for Ab initio Protein Structure Prediction. IEEE Transactions on Evolutionary Computation, 1, doi:10.1109/TEVC.2015.2505317.

Ronkkonen, J., Kukkonen, S., Price, K. V., 2005. Real-parameter optimization with differential evolution. Proc. IEEE CEC, Vol. 1, pp. 506-513.

Salomon, R., 1996. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems 39, 263-278.

Schmitt, L. M., 2001. Theory of genetic algorithms. Theoretical Computer Science 259, 1-61.

Srinivas, M., Patnaik, L. M., 1994. Genetic algorithms: A survey. Computer 27, 17-26.

Srinivasan, D., Seow, T., 2003. Evolutionary Computation. CEC'03, Canberra, Australia, pp. 2292–2297.

Starkweather, T., Whitley, D., Mathias, K., 1990. Optimization using distributed genetic algorithms. International Conference on Parallel Problem Solving from Nature. Springer, pp. 176-185.

Storn, R., Price, K., 1997. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341 - 359.

Tanese, R., 1989. Distributed genetic algorithms. Proceedings of the third international conference on Genetic algorithms. Morgan Kaufmann Publishers Inc., pp. 434-439.

Townsend, A., 2003. Genetic Algorithms–a Tutorial. Citeseer.

Vasconcelos, J., Ramirez, J., Takahashi, R., Saldanha, R., 2001. Improvements in genetic algorithms. IEEE Transactions on Magnetics 37, 3414-3417.

Vesterstrom, J., Thomsen, R., 2004. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Evolutionary Computation, 2004. CEC2004. Congress on, Vol. 2. IEEE, pp. 1980-1987.

Vose, M. D., 1993. Modeling simple genetic algorithms. Foundations of genetic algorithms 2, 63-73.

Voset, M. D., Liepinsl, G. E., 1991. Punctuated equilibria in genetic search. Complex systems 5, 31-44.

Wang, C.-X., Cui, D.-W., Wan, D.-S., Wang, L., 2006. A novel genetic algorithm based on gene therapy theory. Transactions of the Institute of Measurement and Control 28, 253-262.

Whitley, D., 1993a. Cellular genetic algorithms. Proceedings of the 5th International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., pp. 658.

Whitley, D., 1993b. An executable model of a simple genetic algorithm. Foundations of genetic algorithms 2, 45-62.

Whitley, D., 1994. A genetic algorithm tutorial. Statistics and computing 4, 65-85.

Whitley, D., Kauth, J., 1988. GENITOR: A different genetic algorithm. Colorado State University, Department of Computer Science.

Whitley, D., Starkweather, T., 1990. Genitor II: A distributed genetic algorithm. Journal of Experimental & Theoretical Artificial Intelligence 2, 189-214.

Whitley, L. D., 1989. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. ICGA, Vol. 89, pp. 116-123.

Yang, H., Kang, L. S., Chen, Y. P., 2003. A gene-based genetic algorithm for TSP. Chinese Journal of Computers 26, 1753 - 58.

Yang, X.-S., Deb, S., 2009. Cuckoo Search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC), 210 - 214

Yang, X.-S., Hossein Gandomi, A., 2012. Bat algorithm: a novel approach for global engineering optimization. Engineering Computations 29, 464-483.

Yazdani, M., Jolai, F., 2013. A Genetic Algorithm with Modified Crossover Operator for a Two-Agent Scheduling Problem. Shiraz Journal of System Management 1, 1-13.

Yoon, H.-S., Moon, B.-R., 2002. An Empirical Study on the Synergy of Multiple Crossover Operators. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION 6, 212 - 223.

Yu, J. J. Q., Li, V. O. K., 2015. A Social Spider Algorithm for Global Optimization. Neural and Evolutionary Computing 30, 614 - 627.

Zhang, Z., Zhang, N., Feng, Z., 2014. Multi-satellite control resource scheduling based on ant colony optimization. Expert Systems with Applications 41, 2816-2823.

Zhao, R., Tang, W., 2008. Monkey algorithm for global numerical optimization. J. Uncertain Syst. 2, 165 - 176.