CARPATHIAN J. MATH.
Volume **39** (2023), No. 2,
Pages 433 - 458

Online version at https://www.carpathian.cunbm.utcluj.ro/
Print Edition: ISSN 1584 - 2851; Online Edition: ISSN 1843 - 4401
DOI: https://doi.org/10.37193/CJM.2023.02.08

A Comparative Study between Haploid Genetic Algorithms and Diploid Genetic Algorithms

ADRIAN PETROVAN, OLIVIU MATEI and PETRICĂ C. POP

ABSTRACT. In this paper, we make a comprehensive comparison in terms of the quality of the achieved solutions, the corresponding execution time and impact of the genetic operators on the quality of the results between the Haploid Genetic Algorithms (HGAs) and Diploid Genetic Algorithms (DGAs). The standard genetic algorithms, referred to in our paper as HGAs are characterized by the fact that they are using a haploid representation relating an individual with a chromosome, while the DGAs are using diploid individuals which are made of two chromosomes corresponding to the dominant and recessive genes. Even though the general opinion is that DGAs do not provide much benefit as compared to classical GAs, based on extensive computational experiments, we do show that the DGAs are robust, have a high degree of consistency and perform better, sometimes almost twice as well, than the HGAs, but are slower due to the high number of operations to be performed, caused by the duplication of the genetic information. However, the quality of the solutions achieved by the DGAs compensate their relative high execution time. The better quality of the DGAs, proving the efficiency of using diploid genes, is given by the homogeneity of the population which covers the search space thoroughly and in this way being capable of avoiding the local optima.

1. INTRODUCTION

Genetic algorithms (GAs) are adaptive heuristic search techniques, based on the principles of genetics and natural selection, inspired from the theory of natural evolution developed by Charles Darwin [10] based on the ”survival of the fittest” principle. These algorithms were introduced in practice by Holland [22], and the mechanism is similar to the biological process of evolution, according to which only species that are better adapted to the environment are able to survive and evolve over generations, while those less adapted do not survive and eventually disappear, as a result of natural selection. In other words, GAs have the ability to deliver a ”good-enough” solution ”fast-enough”, making them very attractive in solving optimization problems. However, only few aspects of the natural characteristics have been taken into consideration by the classical GA, some other features such as the number of chromosomes carrying out the genetic information of a cell, dominant/recessive nature of genes, the dominance principle, the change of the DNA during the lifetime of an individual, etc. being barely investigated. That is why, there is a continuous quest for improving the performances of the genetic algorithms. GAs have been applied successfully to various problems, see for further details [38, 40, 41].

Usually, the population of genetic algorithms (GAs) consists of haploid individuals, meaning individuals with a single chromosome. As a result, the genetic operators involved in solving a problem using genetic algorithms use the only one chromosome as an informational entity.

Received: 25.03.2022. In revised form: 28.03.2022. Accepted: 04.04.2022
2010 *Mathematics Subject Classification.* 35A01, 65L10, 65L12, 65L20, 65L70.
Key words and phrases. *haploid genetic algorithms, diploid genetic algorithms, dominant and recessive genes.*
Corresponding author: Petrică C. Pop; petrica.pop@mi.utcluj.ro

It is also well-known that if a GA loses the diversity of the population it will be trapped into a local optimum and by default its genetic operators like crossover become ineffective [47]. Several methods have been considered in the literature in order to preserve population diversity in GA such as:

1. Methods based on increasing the population randomness by using high mutation rate or immigration (injection) of new random individuals [7, 27];
2. Niching methods introduced by De Jong [11] were developed in order to decrease the effect of genetic drift resulting from the selection operator in the standard GA. A niche can be seen as a subspace in the environment that may support several types of life. These methods preserve the population diversity and allow GAs to explore many peaks in parallel. Fitness sharing methods are probably the best known and most used niching methods, have been introduced by Holland [22] and improved during time by several researchers. For more information on the main features and developments of this method we refer to [42]. Other proposed niching methods are: the crowding methods and the clearing methods, for further information regarding these approaches, we refer to [11, 42] .
3. The restricted mating strategy applies conditions such as restriction or encouragement, in order to select an individual and its mate. The first such strategies were based on similarity relations between the parents [12]. Although these methods have been shown to benefit GA performance, they are costly in computational terms. Galan et al. [17] described an approach that has two main features: it permits mating preferences to be defined either in terms of similarity between individuals or in terms of fitness of individuals, and it lends itself to a self-adaptive implementation in which each individual from the population has its own mating preference.
4. Another idea of maintaining population diversity was based on the use of multiploid (polyploid chromosomes) individuals considered by Collingwood et al. [6], having as a source of inspiration the real situation from nature: there exist many organisms having polyploid genotypes consisting of multiple sets of chromosomes with different mechanisms for determining the dominant genes. Several papers investigated the diploidy case, i.e. individual comprising of two chromosomes starting with Goldberg and Smith [19] who investigated the use of diploid representations and dominance operators in GAs in order to improve performance in environments that vary with time. Bhasin and Mehta [2] reviewed the work done so far concerning the diploid genetic algorithms and their conclusion was that DGAs have not been explored sufficiently. Next we will present some of the most important achievements in the area of DGAs. Yukiko and Nobue [47] described a diploid genetic algorithm (DGA) for preserving the population diversity using the idea of meiosis to convert the genotype to phenotype. Lieckens et al. [26] introduce a very simple diploid GA that limits the GA mainly in its discrete time, non-overlapping populations setup and its representation of genotypes, and they also provided formal methods to be used to study finite population models of diploid genetic algorithms, while Bull [4] presents a new variant of evolutionary algorithm that harnesses the haploid-diploid cycle present in eukaryotic organisms. In [45] Yang investigates the effect of the cardinality of genotypic representation and the existence of uncertainty in the dominance scheme for DGAs in dynamic environments. Other much practical approaches on the use of DGA were presented by Bhasin et al. [3] focusing on the performances of DGA in relation to the greedy approach in the case of the dynamic traveling salesman problem. Pop et al. successfully used DGAs in order to solve the generalized traveling salesman

problem [35], the generalized minimum spanning tree problem [36] and the family traveling salesman problem [37]. Dulebenets [14] propose a Diploid Evolutionary Algorithm (DEA) that can assist the cross-docking operators with the design of cost-efficient truck schedules, that can facilitate the flow of different products within the cross-docking facilities, ensure the "just-in-time" deliveries within supply chains, and improve sustainability of the supply chain operations. Petrovan et al. presented some of the advantages of diploid GAs in comparison to standard GAs using real-coded chromosome representation [33] and a behavioural study of the crossover operators in diploid genetic algorithms [34]. Jasuja [23] used DGA on multi-objective optimization of a classification problem for feature selection, showing the advantages of DGA in dealing with this kind of problems. Recently, Dulebenets [15] proposed an adaptive polyploid memetic algorithm for scheduling trucks at a cross-docking terminal whose main feature is that the number of chromosome copies is controlled through the adaptive polyploid mechanism based on the objective function improvements achieved and computational time changes.

Given these considerations, as well as the lack of a comparative approach to the behavior of diploid genetic algorithms compared to haploid ones, the aim of this article is to make a comprehensive comparison in terms of the quality of the achieved solutions, the corresponding execution time and the impact of the genetic operators on the quality of the results between the Haploid Genetic Algorithms (HGAs) and Diploid Genetic Algorithms (DGAs) based on extensive computational experiments on a set comprising 17 benchmark functions with different properties and difficulties, used for the performance evaluation of genetic algorithms.

The remaining part of the paper is organized as follows: Section 2 presents the basic concepts from the theory of genetic algorithms. The proposed genetic crossover operators in the case of DGA that are used in this study are described in Section 4. The experiments performed and the achieved results are presented and detailed in Section 5. Finally, the conclusions and future research directions are presented in the last section.

2. Ploidy in genetic algorithms

In nature, the cells contain various number of sets of chromosomes, varying from 1 to 64. This feature is known as *ploidy* [44]. When there is at least a pair of chromosomes, a characteristic of the cells, given by a gene (allele) is determined based on the phenotype, which is a phenomenon which uses the dominant or recessive character of the respective gene, and the genetic information thus determined will be transmitted to the offspring. Therefore, within the diploid or more complex individuals a greater diversity of characteristics is transmitted due to the additional information content and the way of transmitting it to the offspring.

However, the use of genetic algorithms (GA) in solving different problems focuses on haploid representation, in particular because of their ease of implementation. The haploid representation retains only one set of each gene, thus avoiding the entire process of determining the dominant allele that must be transmitted further in the population, the recessive ones being avoided.

2.1. The haploid genetic algorithms.

In the case of conventional genetic algorithms an individual consists of one chromosome, which is a possible (feasible) solution of the optimization problem. The representation of the solution depends on the specificity of each problem, as well as the way of solving it from the computer point of view. The algorithm starts with a set of randomly generated solutions (represented by chromosomes) called population. The solutions (called individuals) of a population are taken and used to form a new population, motivated by a hope, that the new population will be better than the old one. The solutions (offspring) chosen to form a new population are selected on the basis of a strategy, which is appropriate to their form of representation. The better chromosomes that exist within the population (determined on the basis of their fitness), the more likely they are to be selected.

As a result, the crossover operator has a significant role in a genetic algorithm. The specialized literature presents numerous crossing techniques [21], starting from some very simple ones to very elaborate crossing techniques. The advantage of choosing one in favor of the other depends mainly on the results that they offer in terms of increasing the convergence character of the solution towards the optimal global solution. Depending on the type of problem, whose optimization function falls into one or more mathematical patterns, choosing the type of crossover can result in finding an optimal solution in a shorter time. The result of any crossing technique represents two offspring, each carrying genetic information from both parents. In certain circumstances, in some newly formed offspring one or more of their genes may be subject to mutation, with a low random probability. This implies that the value of a gene is changed.

2.2. **The diploid genetic algorithms.** In the case of diploid genetic algorithms (DGA), individuals consist of two coupled haploid chromosomes.

$$I = (C_1, C_2) \tag{2.1}$$

where C_i are the chromosomes, with $i \in \{1, 2\}$.

This type of representation is called the diploid (bi-chromosomal). Also, each individual carries additional information called phenotype, which is necessary for the selection of the individuals who will participate in the crossover process for the formation of new offspring. In this way, the diploid representation of superior life forms is mimicked [29]. The superiority of this representation lies in the fact that each person carries twice as much information as compared to the classical haploid approach, thus a greater diversity being ensured in terms of potential workable solutions [19]. The final values of the phenotype of an individual are decided by the dominance schemes that play an important role in the algorithm's performance. It is particularly important to design and experiment with a good dominance scheme to guarantee the performance of the diploid GA as compared to the haploid one.

One of the most important dominance schemes described in the literature have been proposed by Ng and Wong [31], in which the dominant allele will always be part of the phenotype. If there is a conflict between two dominant or two recessive alleles, the selected one is strictly random. Another elaborated approach to the dominance scheme is offered by Yang and Yao [46] and is called the dominance learning scheme in which a dominant probability vector is defined. Within this scheme each element has a dominance probability which represents the probability that a genotypic allele can be expressed within the phenotype.

3. Genetic operators in the case of DGA

In this section we will describe the genetic operators in the case of diploid genetic algorithms (DGA).

3.1. **Crossover operators.** Due to the importance of crossover operator in GAs, we present an adaptation in the case of DGA of some of the crossover operators for real numbers, as described by Herrera et al. [21] in the case of HGA.

Further, we assume that the two individuals I_1 and I_2 to undergo recombination have the following structure:

$$\begin{aligned} I_1 &= (C_1^1, C_2^1),\ \text{where } C_1^1 = (c_{11}^1, ..., c_{1D}^1) \text{ and } C_2^1 = (c_{21}^1, ..., c_{2D}^1), \\ I_2 &= (C_1^2, C_2^2),\ \text{where } C_1^2 = (c_{11}^2, ..., c_{1D}^2) \text{ and } C_2^2 = (c_{21}^2, ..., c_{2D}^2), \end{aligned} \tag{3.2}$$

where D is the dimension of the chromosomes.

By combining different chromosomes belonging to different individuals, we can observe that we obtain a higher number of distinct offspring in comparison to the classical HGA. In this way, we ensure an increased diversity of the created offspring, an aspect that is also reflected on the basin of formation of the new population.

3.1.1. *Simple crossover (one cut) (SX-DGA).* A position $i \in \{1, 2, ..., D-1\}$ is chosen randomly and the resulting four offspring $O_k = (O_1^k, O_2^k)$ for $k \in \{1, .., 4\}$ are created as follows:

$$\begin{aligned} O_1 &= \Big((c_{11}^1, ..., c_{1i}^1, c_{1i+1}^2, ..., c_{1D}^2), (c_{21}^1, ..., c_{2i}^1, c_{2i+1}^2, ..., c_{2D}^2)\Big) \\ O_2 &= \Big((c_{11}^1, ..., c_{1i}^1, c_{2i+1}^2, ..., c_{2D}^2), (c_{21}^1, ..., c_{2i}^1, c_{1i+1}^2, ..., c_{1D}^2)\Big) \\ O_3 &= \Big((c_{11}^2, ..., c_{1i}^2, c_{1i+1}^1, ..., c_{1D}^1), (c_{21}^2, ..., c_{2i}^2, c_{2i+1}^1, ..., c_{2D}^1)\Big) \\ O_4 &= \Big((c_{11}^2, ..., c_{1i}^2, c_{2i+1}^1, ..., c_{2D}^1), (c_{21}^2, ..., c_{2i}^2, c_{1i+1}^1, ..., c_{1D}^1)\Big) \end{aligned} \tag{3.3}$$

3.1.2. *Two point crossover (TPX-DGA).* The two point crossover [30] derives from the simple crossover, but uses two cutting points $i, j \in \{1, 2, ..., D-1\}$ with $i < j$. The resulting offspring $O_k = (O_1^k, O_2^k)$ for $k \in \{1, .., 4\}$ are created as follows:

$$\begin{aligned} O_1 &= \Big((c_{11}^1, ..., c_{1i}^1, c_{1i+1}^2, ..., c_{1j}^2, c_{1j+1}^1, ..., c_{1D}^1), (c_{21}^1, ..., c_{2i}^1, c_{2i+1}^2, ..., c_{2j}^2, c_{2j+1}^1, ..., c_{2D}^1)\Big) \\ O_2 &= \Big((c_{11}^1, ..., c_{1i}^1, c_{1i+1}^2, ..., c_{1j}^2, c_{1j+1}^1, ..., c_{1D}^1), (c_{21}^2, ..., c_{2i}^2, c_{2i+1}^1, ..., c_{2j}^1, c_{2j+1}^2, ..., c_{2D}^2)\Big) \\ O_3 &= \Big((c_{11}^2, ..., c_{1i}^2, c_{1i+1}^1, ..., c_{1j}^1, c_{1j+1}^2, ..., c_{1D}^2), (c_{21}^1, ..., c_{2i}^1, c_{2i+1}^2, ..., c_{2j}^2, c_{2j+1}^1, ..., c_{2D}^1)\Big) \\ O_4 &= \Big((c_{11}^2, ..., c_{1i}^2, c_{1i+1}^1, ..., c_{1j}^1, c_{1j+1}^2, ..., c_{1D}^2), (c_{21}^2, ..., c_{2i}^2, c_{2i+1}^1, ..., c_{2j}^1, c_{2j+1}^2, ..., c_{2D}^2)\Big) \end{aligned} \tag{3.4}$$

3.1.3. *Uniform crossover (UX-DGA).* In the case of uniform crossover, we do not divide the chromosomes into segments, rather each gene is selected at random from one of the corresponding genes of the chromosomes that constitutes the individuals. The four offspring $O_k = (O_1^k, O_2^k)$ for $k \in \{1, 2, 3, 4\}$ are created as follows:

$$O_k = \Big((o_{11}^k, ..., o_{1D}^k), (o_{21}^k, ..., o_{2D}^k)\Big), \tag{3.5}$$

and are built from genes as follows:

$$o_{ij}^k = \begin{cases} c_i^1 \text{ if } u = 0 \\ c_i^2 \text{ if } u = 1 \end{cases} \tag{3.6}$$

where $k \in \{1, 2, 3, 4\}$, $i \in \{1, 2\}$ and $j \in \{1, ..., D\}$.

3.1.4. *Arithmetic crossover (AX-DGA).* The arithmetic crossover linearly combines the chromosomes of the parent individuals and produces four offspring $O_k = (O_1^k, O_2^k)$ for $k \in \{1, 2, 3, 4\}$, with the associated genes:

$$\begin{aligned} O_1 &= \left(\lambda c_{1i}^1 + (1-\lambda)c_{2i}^1, \lambda c_{1i}^2 + (1-\lambda)c_{2i}^2\right) \\ O_2 &= \left(\lambda c_{1i}^1 + (1-\lambda)c_{2i}^1, \lambda c_{2i}^2 + (1-\lambda)c_{1i}^2\right) \\ O_3 &= \left(\lambda c_{2i}^1 + (1-\lambda)c_{1i}^1, \lambda c_{1i}^2 + (1-\lambda)c_{2i}^2\right) \\ O_4 &= \left(\lambda c_{2i}^1 + (1-\lambda)c_{1i}^1, \lambda c_{2i}^2 + (1-\lambda)c_{1i}^2\right) \end{aligned} \tag{3.7}$$

where $\lambda \in [0, 1]$ is randomly generated and $i \in \{1, ..., D\}$.

3.1.5. *BLX-α crossover (BLX-DGA).* In the classical GAs, the blend crossover (BLX-α) was introduced by Eshelman and Schaffer [16] and it provided a good searching ability for separable fitness functions, while facing some difficulties in the optimization of non-separable fitness functions. In the case of DGA the crossover is extended as follows: for each individual $k = 1, 2$ at each locus $i \in \{1, ..., D\}$, the following values are calculated $C_{ki}^{min} = min(c_{1i}^k, c_{2i}^k)$, $C_{ki}^{max} = max(c_{1i}^k, c_{2i}^k)$ and $J_k = C_{ki}^{max} - C_{ki}^{min}$. With these values thus calculated, two gametes are generated for each individual, $G_k = (g_{1i}^k, g_{2i}^k)$ in the following form: g_{ji}^k = is a randomly (uniformly) chosen number from the interval $[C_{ki}^{min} - J_k\alpha, C_{ki}^{max} + J_k\alpha]$ where $i \in 1, ..., D$, $j = 1, 2$ and $k = 1, 2$. From these gametes four offspring are built in the following form:

$$O_1 = (g_{1i}^1, g_{1i}^2), O_2 = (g_{1i}^1, g_{2i}^2), O_3 = (g_{2i}^1, g_{1i}^2), O_4 = (g_{2i}^1, g_{2i}^2) \tag{3.8}$$

3.1.6. *Max-Min Arithmetic crossover (MMAX-DGA).* The Max-Min arithmetic crossover operator described by Herrera et al. [21] in the case of classical HGA, can be extended to DGA as follows: for each individual $k = 1, 2$ at each locus $i \in \{1, ..., D\}$, the following values are calculated: $C_{ki}^{min} = min(c_{1i}^k, c_{2i}^k)$, $C_{ki}^{max} = max(c_{1i}^k, c_{2i}^k)$. MMAX-DGA combines two diploid individuals and generates six offspring as follows:

$$\begin{aligned} O_1 &= \left(\lambda c_{1i}^1 + (1-\lambda)c_{2i}^1, \lambda c_{1i}^2 + (1-\lambda)c_{2i}^2\right) \\ O_2 &= \left(\lambda c_{1i}^1 + (1-\lambda)c_{2i}^1, \lambda c_{2i}^2 + (1-\lambda)c_{1i}^2\right) \\ O_3 &= \left(\lambda c_{2i}^1 + (1-\lambda)c_{1i}^1, \lambda c_{1i}^2 + (1-\lambda)c_{2i}^2\right) \\ O_4 &= \left(\lambda c_{2i}^1 + (1-\lambda)c_{1i}^1, \lambda c_{2i}^2 + (1-\lambda)c_{1i}^2\right) \\ O_5 &= \left(C_{1i}^{min}, C_{2i}^{min}\right) \\ O_6 &= \left(C_{1i}^{max}, C_{2i}^{max}\right) \end{aligned} \tag{3.9}$$

where $\lambda \in [0, 1]$ is randomly generated, $i \in \{1, ..., D\}$ and $k = 1, 2$.

3.1.7. *Linear crossover (LX-DGA).* The process begins with gametes creation for each individual, practically three gametes from the two chromosomes of each individual, according to the following formulae, provided by Schlierkamp-Voosen and Mühlenbein [42]:

$$g_{1i}^k = \frac{1}{2}c_{1i}^k + \frac{1}{2}c_{2i}^k;\ g_{2i}^k = \frac{3}{2}c_{1i}^k - \frac{1}{2}c_{2i}^k;\ g_{3i}^k = -\frac{1}{2}c_{1i}^k + \frac{3}{2}c_{2i}^k \tag{3.10}$$

where $i \in 1, ..., D$. Having these gametes, the resulting nine offspring are built by combining gametes generated from the chromosomes of each individual.

From the point of view of a taxonomy, the presented crossover operators are part of four major groups of classification [21]: the simple (SX-DGA), two-point (TPX-DGA) and uniform (UX-DGA) crossover operators belong to the group of discrete crossover operators, arithmetical (AX-DGA) and linear (LX-DGA) are part of the group of aggregation based crossover operators, BLX-α-DGA is part of the group of neighborhood based crossover operators and max-min-arithmetic (MMAX-DGA) belongs to the group of hybrid crossover operators.

3.2. **Mutation operator.** Due to the fact that in the case of the diploid individual the phenotype expresses its global characteristic within the population, the mutation, that certain individuals suffer, is present in its internality, both chromosomes being affected at random positions. This mechanism ensures the diversity of the individual in relation to the population it comes from.

4. Computational experiments

In this section we present a comprehensive comparison between haploid and diploid genetic algorithms based on the extensive computational results.

In the field of evolutionary computation, the comparison of different algorithms is commonly done by using a large set of standard benchmarks or test functions from the literature. Trying to design a complete study using a whole set of test functions, in order to determine if one algorithm is better than the other for each function, is a pointless task. This is why, in our opinion when evaluating an algorithm, we must emphasize the type of problems in which its performance is good, in order to characterize the type of problems for which the algorithm is suitable and for which it is not. For a survey of benchmark functions for solving optimization problems, we refer to Jamil and Yang [24].

In order to compare the investigated haploid and diploid genetic algorithms in an unbiased manner, and to validate this comparison, we used a collection of 17 test functions with diverse properties in terms of modality, regularity, separability, and valley landscape.

The test functions used in our computational experiments are presented hereafter, together with their main features.

1. **Schwefel's function** [43] defined as:

$$F_1(x_1, x_2, ..., x_D) = 418.9829 \times D - \sum_{i=1}^{D} x_i sin(\sqrt{|x_i|}) \tag{4.11}$$

subject to $-500 \leq x_i \leq 500$, for $i = 1, ..., D$. The considered Schwefel function is continuous, differentiable, separable, scalable, multimodal, has many local optima and the global minimum is located at $x^\star = (420.9687, ..., 420.9687)$ and $F_1(x^\star) = 0$.

2. **Ackley's function** [24] defined as:

$$\begin{aligned} F_2(x_1, x_2, ..., x_D) = &-20\, exp\Big(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\Big) - \\ &- exp\Big(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\Big) + 20 + e \end{aligned} \tag{4.12}$$

subject to $-32 \leq x_i \leq 32$, for $i = 1, ..., D$. The considered Ackley function is continuous, differentiable, non-separable, scalable, multimodal, and has one global minimum $F_2(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

3. **Rastrigin's function** [13] defined as:

$$F_3(x_1, x_2, ..., x_D) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) \right) \tag{4.13}$$

   subject to $-5.12 \leq x_i \leq 5.12$, for $i = 1, ..., D$. Rastrigin's function is continuous, differentiable, convex, separable, multimodal, and has one global minimum $F_3(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

4. **Griewank's function** [20] defined as:

$$F_4(x_1, x_2, ..., x_D) = 1 + \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) \tag{4.14}$$

   subject to $-100 \leq x_i \leq 100$, for $i = 1, ..., D$. Griewank's function is continuous, differentiable, non-separable, scalable, multimodal, and has one global minimum $F_4(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

5. **Rosenbrock's function** [24] defined as:

$$F_5(x_1, x_2, ..., x_D) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right] \tag{4.15}$$

   subject to $-2.048 \leq x_i \leq 2.048$, for $i = 1, ..., D$. Rosenbrock's function is continuous, differentiable, non-separable, scalable, unimodal, and has one global minimum $F_5(x^\star) = 0$ at $x^\star = (1, ..., 1)$.

6. **Alpine N. 1 function** [24] defined as:

$$F_6(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} \mid x_i \sin(x_i) + 0.1x_i \mid \tag{4.16}$$

   subject to $0 \leq x_i \leq 10$, for $i = 1, ..., D$. Alpine N. 1 function is continuous, non-convex, non-differentiable, separable, non-scalable, multimodal, and has one global minimum $F_6(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

7. **Sphere function** [24] defined as:

$$F_7(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} x_i^2 \tag{4.17}$$

   subject to $-5.12 \leq x_i \leq 5.12$, for $i = 1, ..., D$. The Sphere function is continuous, convex, differentiable, separable, scalable, unimodal, and has one global minimum $F_7(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

8. **Step function** [24] defined as:

$$F_8(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} \left(\lfloor x_i + 0.5 \rfloor \right)^2 \tag{4.18}$$

   subject to $-100 \leq x_i \leq 100$, for $i = 1, ..., D$. The Step function is discontinuous, non-differentiable, separable, scalable, unimodal, and has one global minimum $F_8(x^\star) = 0$ at $x^\star = (0.5, ..., 0.5)$.

9. **Sum Squares function** [24] defined as:

$$F_9(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} i x_i \tag{4.19}$$

subject to $-10 \leq x_i \leq 10$, for $i = 1, ..., D$. The Sum Squares function is continuous, convex, differentiable, separable, scalable, unimodal, and has one global minimum $F_9(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

10. **Quartic function** [24] defined as:

$$F_{10}(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} ix_i^4 + random[0, 1] \tag{4.20}$$

subject to $-1.28 \leq x_i \leq 1.28$, for $i = 1, ..., D$. The Quartic function is continuous, differentiable, separable, scalable, unimodal, and has one global minimum $F_{10}(x^\star) = 0$ at $x^\star = (1, ..., 1)$.

11. **Qing's function** [24] defined as:

$$F_{11}(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} \left(x_i^2 - i\right)^2 \tag{4.21}$$

subject to $-500 \leq x_i \leq 500$, for $i = 1, ..., D$. The Qing's function is continuous, non-convex, differentiable, separable, scalable, multimodal, and has one global minimum $F_{11}(x^\star) = 0$ at $x^\star = (\pm\sqrt{i}, ..., \pm\sqrt{i})$.

12. **Dixon-Price function** [24] defined as:

$$F_{12}(x_1, x_2, ..., x_D) = (x_1 - 1)^2 + \sum_{i=2}^{D} i \left(2x_i^2 - x_{i-1}\right)^2 \tag{4.22}$$

subject to $-10 \leq x_i \leq 10$, for $i = 1, ..., D$. The Dixon-Price function is continuous, differentiable, non-separable, scalable, unimodal, and has one global minimum $F_{12}(x^\star) = 0$ at $x^\star = (2^{-\frac{2^i-2}{2^i}}, ..., 2^{-\frac{2^i-2}{2^i}})$.

13. **Exponential function** [24] defined as:

$$F_{13}(x_1, x_2, ..., x_D) = -exp\left(-0.5\sum_{i=1}^{D} x_i^2\right) \tag{4.23}$$

subject to $-1 \leq x_i \leq 1$, for $i = 1, ..., D$. The Exponential function is continuous, differentiable, non-separable, scalable, multimodal, and has one global minimum $F_{13}(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

14. **Periodic function** [1] defined as:

$$F_{14}(x_1, x_2, ..., x_D) = 1 + \sum_{i=1}^{D} sin^2(x_i) - 0.1e^{(\sum_{i=1}^{D} x_i^2)} \tag{4.24}$$

subject to $-10 \leq x_i \leq 10$, for $i = 1, ..., D$. The Periodic function is continuous, non-convex, differentiable, non-separable, scalable, multimodal, and has one global minimum $F_{14}(x^\star) = 0.9$ at $x^\star = (0, ..., 0)$.

15. **Powell Sum function** [24] defined as:

$$F_{15}(x_1, x_2, ..., x_D) = \sum_{i=1}^{D} |x_i|^{i+1} \tag{4.25}$$

subject to $-1 \leq x_i \leq 1$, for $i = 1, ..., D$. The Periodic function is continuous, convex, non-differentiable, separable, scalable, unimodal, and has one global minimum $F_{15}(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

16. **Ridge function** [24] defined as:

$$F_{16}(x_1, x_2, ..., x_D) = x_1 + d\left(\sum_{i=2}^{D} x_i^2\right)^{\alpha} \tag{4.26}$$

where d and α are constants that usually are set to $d = 2$ and $\alpha = 0.5$, $-5 \leq x_i \leq 5$, for $i = 1, ..., D$. The Ridge function is non-convex, differentiable, non-separable, unimodal, and has one global minimum that depends on the hypercube it is defined on. On the hypercube $[-\gamma, \gamma]^D$, $F_{16}(x^\star) = -\gamma$ located at $x^\star = (-\gamma, 0, ..., 0)$.

17. **Schwefel's Double-Sum function** [24] defined as:

$$F_{17}(x_1, x_2, ..., x_D) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2 \tag{4.27}$$

subject to $-65.536 \leq x_i \leq 65.536$, for $i = 1, ..., D$. The Schwefel's Double-Sum function is symmetric, non-separable, unimodal, and has one global minimum $F_{17}(x^\star) = 0$ at $x^\star = (0, ..., 0)$.

Designing a "good" test set of functions in order to compare the performance of two algorithms is a difficult task; in our case the collection of benchmark instances was selected based on the following arguments:

- Dimensionality of the solution space is an important aspect that affects the complexity of the problem, see for further information [18]. In our experiments we used only multidimensional problems and in order to have the same degree of difficulty, for all the considered test functions we have selected two dimensions: $D = 25$ and $D = 50$ of the search space.
- Separability is a notion closely related to the concept of epistasis, which in the area of evolutionary algorithms measures how much the contribution of a gene to the fitness of a given individual relies upon the values of the other genes. Obviously, the non-separable functions are more difficult to be optimized because the search direction depends on more genes while the separable functions can be optimized for each variable. In our case, we considered 8 separable functions (Ackley, Griewank, Rosenbrock, Dixon-Price, Exponential, Periodic, Ridge and Schwefel double-sum) and the other are non-separable.
- Multimodality is a notion related to multimodal functions, i.e. functions that have two or more local optima, while unimodal functions satisfy the following property: for some value m it is monotonically increasing for $x \leq m$ and monotonically decreasing for $x \geq m$. In the case of multimodal functions, the search process should be able to avoid the regions around local optima in order to approximate as good as possible the global optimum. Obviously, the most difficult case appears when the local optima are randomly distributed within the search space. In our case, we considered 8 multimodal functions (Schwefel, Ackley, Rastrigin, Griewank, Alpine N. 1, Qing, Exponential and Periodic) and the other are unimodal.
- An important group of selected benchmark test functions contains functions that have flat surfaces which are real obstacles for optimization algorithms, and in which it is very difficult to direct the search process of an algorithm toward the local optima because they do not give any information as to which direction is favorable. In our case, we considered 4 functions having this feature: Power Sum, Powell, Exponential and Ridge.

4.1. **Methodology.** The aim of these experiments is to prove beyond any doubt the advantages of DGA over HGA and to emphasize the differences between the two genetic representations (haploid vs diploid). Therefore, throughout these experiments we keep the running conditions of HGA, respectively DGA similar, while the genetic parameters are identical.

The proposed methodology is divided into two categories of experiments:

(1) Experiments 1 (section 4.2) compare the HGA and the DGA both in terms of quality of the results and running times. The algorithms are tested using all the considered benchmark functions with two dimensions: 25 and 50 respectively. The population size is the same in both cases (HGA and DGA).
(2) Experiments 2 (section 4.3) explores the performance of HGA and DGA with respect to various crossover operators.

Our proposed genetic algorithms: HGA and DGA, were implemented in Java and were tested on a PC with Intel Core i3-8100 @ 3.6GHz, 8GB RAM, Windows 10 Education 64 bit operating system.

Due to the specific nature of the benchmark functions, the genes are represented by real numbers and the solutions space is $\mathbb{R}^D$.

Therefore, a haploid individual is represented as:

$$I_h = (c_1, c_2, ..., c_D) \tag{4.28}$$

where D is the dimension of the benchmark function, and c_i are real numbers, $i = 1, .., D$.

A diploid individual consists of two coupled haploid chromosomes and it is represented as:

$$I_d = (C_1, C_2) = \Big((c_1^1, c_2^1, ..., c_D^1), (c_1^2, c_2^2, ..., c_D^2)\Big). \tag{4.29}$$

The fitness of an individual is based on the phenotype and it can be defined in various ways. Due to the infinity of genetic values for each locus of a chromosome, it is improper to establish a principle of confinement based strictly on the analyzed values. In this sense, the phenotype of an individual encoded by real values is determined based on the average of the values for each locus of an individual as:

$$phen_i = average(c_i^1, c_i^2), i = 1, ..., D. \tag{4.30}$$

Thus, the fitness of an individual is:

$$f = F(phen_1, ..., phen_D). \tag{4.31}$$

This kind of phenotype assures an even influence of the both chromosomes of the individual on the mating and survival chances of each individual.

Due to the importance and impact of the genetic parameters on the performance of the GAs, in both our developed algorithms: HGA and DGA, their values have been selected based on extensive computational experiments and statistical analyses. The chosen number of epochs was 200. Thus we ensure that the algorithms reach their optimum and no evolution is possible. As can be seen further, actually the convergence occurs even much earlier. The mutation rate was selected to be 5%, more or less this being the standard value in GAs.

In both our developed algorithms, we used the roulette wheel selection operator [32], because it is less invasive than the deterministic ones, such as (μ, λ) or $(\mu + \lambda)$, see for further information [28], and allows the population to undergo a more resilient evolution.

In roulette wheel selection, as in all selection methods, the fitness function assigns a fitness to possible solutions or chromosomes. This fitness level is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of individual in the population, its probability of being selected is

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}, \tag{4.32}$$

where N is the number of individuals in the population.

4.2. **Experiments 1. Haploid versus Diploid GAs Comparative Results.** In this experiment, the developed classic (haploid) and diploid algorithms have been tested on the 17 benchmark functions described previously. Because genetic algorithms make use of randomness during the optimization process, each instance is solved multiple times, i.e. 30 runs per instance, and the average values of the best solutions obtained by each algorithm were recorded. Also, the standard deviation of the best values were computed. In all experiments, the values of the parameters used by each algorithm were chosen to be the same: the population size was 1000 individuals, the mutation rate was $p_m = 0.05$ and the number of generations for each run was 200. In both GAs, haploid and diploid ones, the selection was made by the roulette wheel technique and the two point crossover with probability $p_c = 1.0$ was used for the creation of the offspring. Two different tests were carried out, one for the size of a chromosome of 25 genes and the other for a double size of 50 genes.

In Table 1, we display the obtained statistical results by the HGA and DGA over 30 runs in the case of a chromosomal size of 25. Table 1 has the following structure: the first column contains the benchmark function, the second column displays the mathematical global minimum of the corresponding benchmark function, the next three columns provide the average values (Avg. val.) of the best solutions achieved, the standard deviation of the best values (Std. dev.) and the corresponding average computational times reported in seconds in order to obtain the solutions by the HGA, the next three columns provide the average values (Avg. val.) of the best solutions achieved, the standard deviation of the best values (Std. dev.) and the corresponding average computational times reported in seconds in order to obtain the solutions by the DGA, while the last column displays the percentage of performance improvements (I %) of the achieved results of DGA over HGA.

Analyzing the results reported in Table 1, we can notice that in the case of all considered benchmark functions, DGA performed significantly better than HGA in terms of obtained average values. The improvements vary from 7.40% in the case of the Periodic function to 94.67% in the case of the Schwefel's Double-Sum function. Most of the achieved results show an improvement of over 40%, with few exceptions: the Periodic function 7.40% and the Griewank's function 32.11%. We can observe that in 2 out of 17 considered test functions, HGA delivers the same value in all 30 runs, while DGA matches the same value only in the case of the Step function. In the other cases, when the proposed GA's do not provide in all 30 runs the same solution, the standard deviation ranges from 7.806E-18 to 0.0097 for HGA and from 7.806E-18 to 0.0034 in the case of DGA, fact that proves the accuracy and robustness of both HGA and DGA. Regarding the running times, we can observe that the average computational times of the DGA are higher in comparison to those corresponding to the HGA, but this is somehow natural because DGA performs a higher number of operations, caused by the duplication of the genetic information.

TABLE 1. Statistical results of 30 runs obtained by HGA and DGA in the case of a chromosomal size of 25

Test function	Global minimum	HGA			DGA			I [%]
		Avg. val.	Std. dev.	Time	Avg. val.	Std. dev.	Time	
F1	0	35.1090	0.0000	5.432	13.2034	2.131E-13	13.918	62.39
F2	0	3.0915	5.529E-15	5.129	1.6469	2.797E-14	14.362	46.73
F3	0	4.6565	1.509E-15	5.147	1.6431	1.687E-14	13.951	64.71
F4	0	0.3226	7.327E-15	5.348	0.2190	1.540E-15	15.033	32.11
F5	0	39.1601	4.263E-13	4.326	23.3992	5.435E-13	12.261	40.25
F6	0	0.3666	5.107E-15	4.983	0.1172	2.997E-15	14.176	68.03
F7	0	0.1051	1.776E-15	4.486	0.0222	1.734E-16	11.967	78.88
F8	0	69.0000	0.0000	4.427	15.0000	0.0000	12.527	78.26
F9	0	0.0416	2.081E-17	4.415	0.0110	1.890E-16	11.969	73.56
F10	0	0.0054	0.0097	4.623	0.0026	0.0034	12.010	51.85
F11	0	6.3858	1.154E-13	4.411	2.6508	1.332E-15	11.921	58.49
F12	0	12.0806	7.727E-14	4.486	3.5564	6.661E-15	12.179	70.56
F13	-1	-0.2008	9.992E-16	3.750	-0.5981	1.887E-15	10.923	49.71
F14	0.9	1.0804	1.820E-14	6.519	1.0005	5.107E-15	16.739	7.40
F15	0	0.0318	7.806E-18	5.501	0.0141	7.806E-18	15.449	55.66
F16	$-\gamma$	-4.0178	3.153E-14	4.031	-4.5769	7.993E-14	10.948	56.92
F17	0	3.4722	9.787E-7	6.466	0.1850	8.171E-5	14.004	94.67

TABLE 2. Statistical results of 30 runs obtained by HGA and DGA in the case of a chromosomal size of 50

Test function	Global minimum	HGA			DGA			I [%]
		Avg. val.	Std. dev.	Time	Avg. val.	Std. dev.	Time	
F1	0	312.1440	1.023E-12	14.944	77.1097	1.193E-11	48.676	75.30
F2	0	6.1028	5.329E-14	14.987	3.3964	4.44E-16	48.092	43.51
F3	0	31.8316	8.526E-14	14.822	16.0374	1.953E-13	47.932	49.62
F4	0	6.6450	1.669E-13	14.978	1.3217	1.376E-14	49.499	80.11
F5	0	286.7859	1.534E-12	12.015	87.9287	2.131E-13	39.060	69.34
F6	0	1.9785	3.264E-14	14.179	0.8361	1.465E-14	46.111	57.74
F7	0	1.1297	2.131E-14	11.621	0.3274	8.326E-15	38.890	71.02
F8	0	382.0000	0.0000	12.026	88.0000	0.0000	40.791	76.96
F9	0	1.4719	6.661E-15	11.903	0.2360	2.942E-15	39.357	83.97
F10	0	0.1072	0.0100	12.029	0.0709	0.0203	39.819	33.86
F11	0	75.8249	1.676E-12	11.686	69.3935	2.415E-13	39.256	8.48
F12	0	684.3180	1.364E-12	11.828	71.0618	1.833E-12	41.456	80.62
F13	-1	-0.0107	2.550E-16	11.570	-0.1056	8.187E-16	37.640	9.59
F14	0.9	1.0881	7.105E-15	19.530	0.9921	8.659E-15	61.908	8.82
F15	0	0.0536	4.649E-16	16.211	0.0224	1.838E-16	51.87	58.21
F16	$-\gamma$	-1.4667	1.554E-14	11.109	-2.2185	2.309E-14	38.358	21.28
F17	0	19.6610	4.852E-6	13.135	4.2581	9.006E-3	43.862	78.34

In Table 2, we display the obtained statistical results by the HGA and DGA over 30 runs in the case of a chromosomal size of 50. Table 2 has the same structure as Table 1.

Analyzing the results reported in Table 2, we can observe that in the case of all considered benchmark functions, DGA performed significantly better than HGA in terms of obtained average values. The improvements vary from 8.48% in the case of the Qing's function to 83.97% in the case of the Sum Squares function. Most of the obtained results show an improvement of over 33%, with four exceptions: the Qing's function 8.48%, the Periodic function 8.82%, the Exponential function 9.59% and the Ridge function 21.28%. We can observe that in 1 out of 17 considered test functions, HGA delivers the same value in all 30 runs, while DGA matches the same value only in the case of the Step function. In the other cases, when the proposed GA's do not provide the same solution in all 30 runs, the standard deviation ranges from 4.649E-16 to 0.01 for HGA and from 1.838E-16 to 0.0203 in the case of DGA, fact that proves the accuracy and robustness of both HGA and DGA. Regarding the running times, we can observe that the average computational times of the DGA are higher in comparison to those corresponding to the HGA, but this is somehow natural because DGA performs a higher number of operations, caused by the duplication of the genetic information.

As a preliminary conclusion of this first experiment, it is obvious that the quality of the results obtained by running a DGA outperforms those achieved in the case of conventional genetic algorithms (HGAs) and at the same time, we can observe a degradation of the average values obtained as the size of the problem increases in the case of both considered GA's: HGA and DGA.

In Figures 1 and 2, we illustrate the convergence curves for both HGA and DGA. Analyzing the displayed charts, we may observe that the HGA converge faster than DGA in all cases, with 3 to 24 epochs. This behavior is the result of the increased diversity of the population in the case of DGA, a direct result of the increased number of descendants from which the new population is formed. The reason behind the high convergence rate of the HGA resides in the power of the best individuals to draw after them all the other individuals and often getting stuck in local optima. In order to make the charts understandable the Y axis is pruned due to the high diversity in the case of DGA. For Schwefel, Griewank, Rosenbrok, Sphere, Step, Sum Squares, Qing and Dixon-Price benchmark functions, both HGA and DGA algorithms converge after about the same number of epochs. In the case of Ackley, Rastrigin, Quartic, Periodic, Powell Sum and Ridge benchmark functions, HGA converges earlier than DGA. For all these functions, the diploid population is better than the haploid one, because as we can see in Figures 1 and 2, the red area representing the the diploid population fitness is bellow the gray area representing the fitness of the haploid population.

We define the *intra-generation diversity* of a current population as follows:

$$d_i = \mid I_w^i - I_b^i \mid \tag{4.33}$$

where d_i is the diversity of the population in epoch i and I_w^i and I_b^i are the worst, respectively the best individual of the generation i. Evidently, large populations are more likely to maintain genetic material and in general have higher genetic diversity, while small populations are more likely to experience the loss of diversity over time. A genetic bottleneck may occur when a population goes through a low number of individuals, resulting in a rapid decrease in genetic diversity.

In Table 3, we present the average intra-generation diversity of the haploid and diploid populations respectively. The average intra-generation diversity was computed as follows:

$$d = \frac{\sum_{i=1}^{E} d_i}{E} \tag{4.34}$$

where E is the number of epochs until the convergence (not the maximum number of epochs) and d_i is the diversity of the generation i.

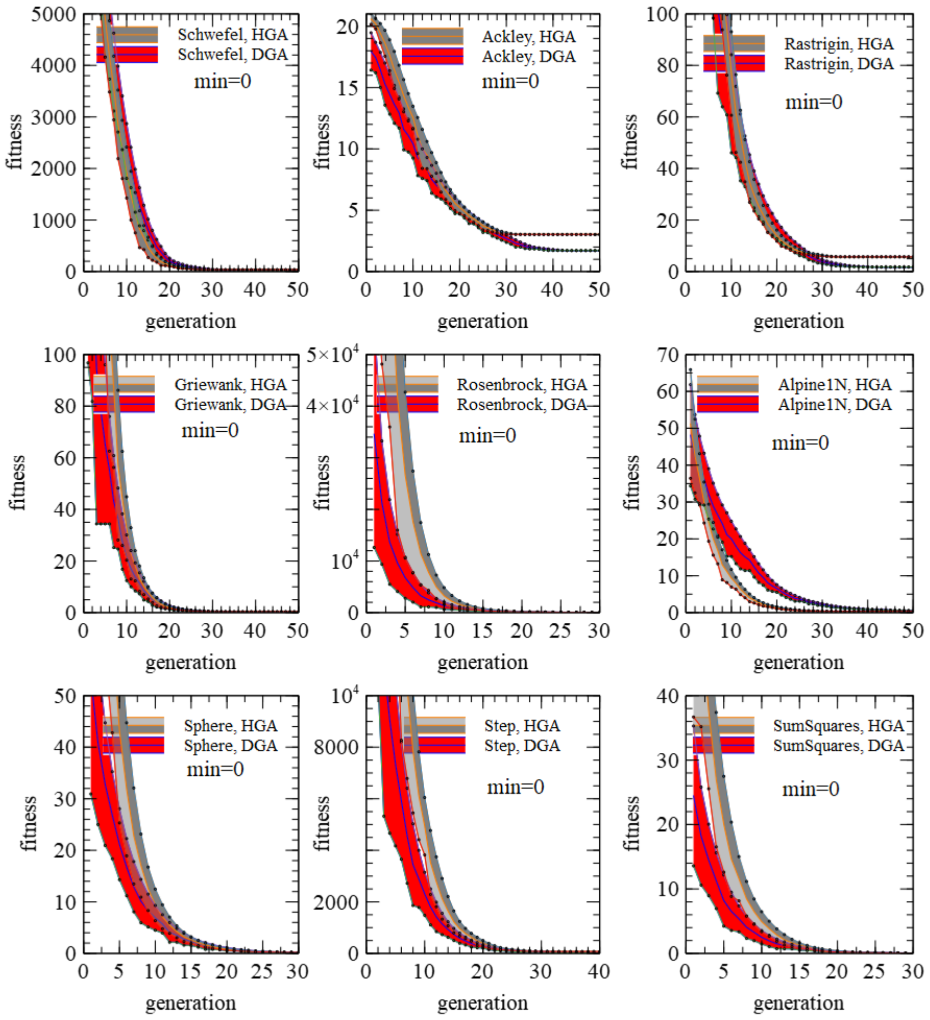

FIGURE 1. The evolution of the population for HGA (gray), respectively DGA (red) in the case of benchmark functions F1 - F9

As we expected, HGA has a larger intra-generation diversity, although the best individuals are significantly worse than the diploid best individuals because the phenotype-based fitness overcome the badness of the worst chromosome of each individual.

HGA converges faster than DGA, but the best individuals drag the worse ones after them in local optima. On the other hand, DGA covers the search space better and thoroughly, the convergence is slower, but the best chromosomes do not have the power to drag the population after them and a so called ”crowd wisdom” [25] avoids local optima better.

Starting from the previously reported results, the next set of experiments aims at evaluating the quality of the results both from the point of view of the average values for each test function and of the running times, in the conditions in which the population size is halved in the case of the diploid genetic algorithm. Each experiment was repeated 30 times with random values in every run, for each run mean values of the best values and

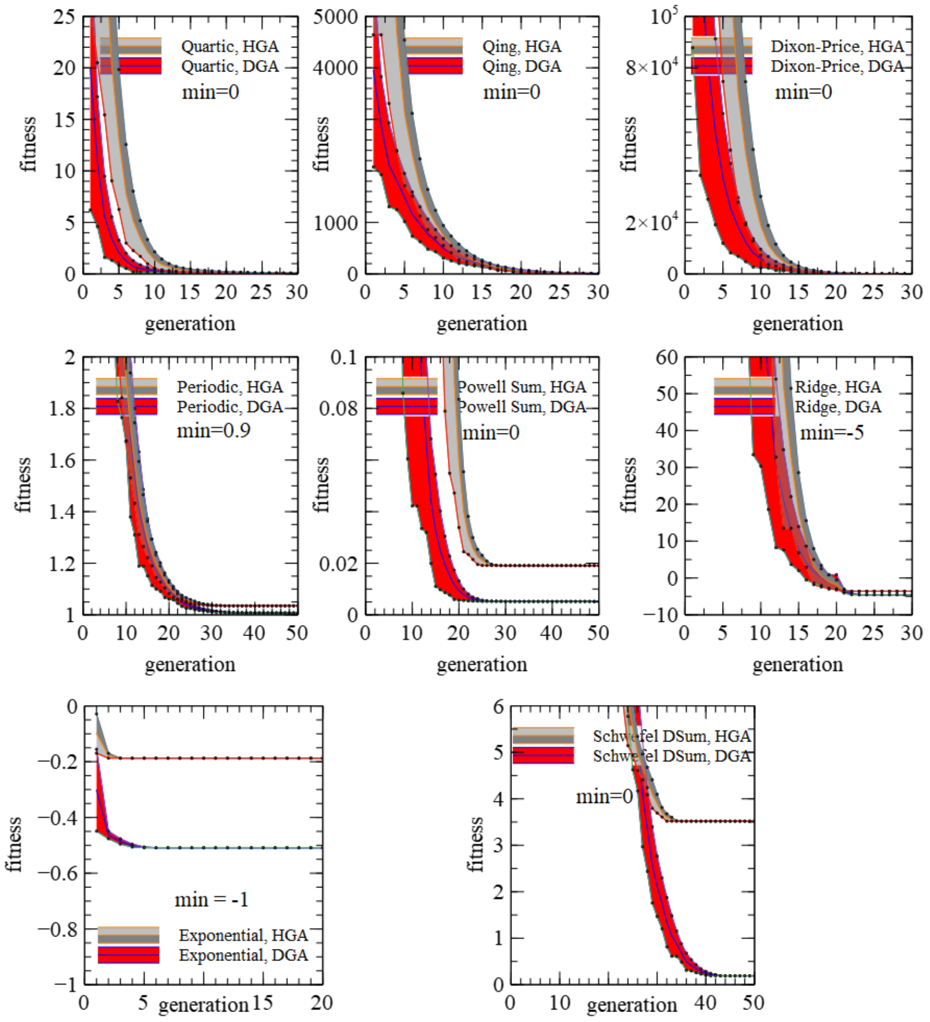

FIGURE 2. The evolution of the population for HGA (gray), respectively DGA (red) in the case of benchmark functions F10 - F17

running times for each run were recorded. Also, these tests were performed for both chromosomal sizes of 25 and 50 genes, the number of new generations computed being 200. The results of this experiment are presented in Table 4 and Table 5 for the chromosomal size of 25 genes and in and in Tables 6 and 7 for the chromosomal size of 50 genes. For each test performed, the factor of improving the quality of the result was calculated based on the minimum value of each test function.

Tables 4 - 7 have the following structure: the first two columns indicate the used test function and its corresponding global minimum, the next three columns provide the average values (Avg.val.) obtained by the HGA in the case of a population size of 1000, respectively 2000, and the corresponding average computational times reported in seconds in order to achieve the corresponding solutions, the next three columns provide the average values obtained by the DGA in the case of a population size of 500, respectively 1000, and the corresponding average computational times reported in seconds in order

TABLE 3. The average intra-generation diversity of the populations

Test Function		Average HGA	Average DGA
Schwefel	F1	589.57	374.48
Ackley	F2	1.321	1.26
Rastrigin	F3	24.069	19.00
Griewank	F4	35.45	13.69
Rosenbrock	F5	13175.69	2324.86
Alpine N1	F6	3.70	2.23
Sphere	F7	12.08	3.71
Step	F8	4626.42	1666.30
Sum-Squares	F9	5.26	1.67
Quartic	F10	2.46	0.62
Qing	F11	1528.68	287.19
Dixon-Price	F12	9689.34	2079.18
Exponential	F13	0.081	0.079
Periodic	F14	0.55	0.52
Powell Sum	F15	24374.85	4.49
Ridge	F16	5009.95	872.94
Schwefel Double-Sum	F17	24.06	19.00

TABLE 4. Statistical results of 30 runs obtained by HGA and DGA in the case of a chromosomal size of 25

Test function	**Global minimum**	**Population size**	**Avg.val. HGA**	**Time**	**Population size**	**Avg.val. DGA**	**Time**	**I [%]**
F1	0	1000	35.109	5.432	500	33.6361	6.192	4.20
F2	0	1000	3.0915	5.129	500	2.772	6.429	10.33
F3	0	1000	4.6565	5.147	500	3.0499	6.164	34.50
F4	0	1000	0.3226	5.348	500	0.3094	6.570	4.09
F5	0	1000	39.1601	4.326	500	29.3468	5.320	25.06
F6	0	1000	0.3666	4.983	500	0.2873	6.266	21.63
F7	0	1000	0.1051	4.486	500	0.0607	5.264	42.25
F8	0	1000	69.00	4.427	500	52.00	5.100	24.64
F9	0	1000	0.0416	4.415	500	0.0368	5.237	11.54
F10	0*	1000	0.0054	4.623	500	0.0041	5.308	24.07
F11	0	1000	6.3858	4.411	500	4.8151	5.237	24.60
F12	0	1000	12.0806	4.486	500	6.5237	5.320	46.00
F13	-1	1000	-0.2008	3.750	500	-0.6551	4.914	56.84
F14	0.9	1000	1.0804	6.519	500	1.0208	7.595	5.52
F15	0	1000	0.0318	5.501	500	0.0235	6.601	26.10
F16	$-\gamma$	1000	-4.0178	4.031	500	-4.2036	4.848	18.92
F17	0	1000	3.4722	6.466	500	0.2532	6.140	92.71

to achieve the corresponding solutions, and the last column display the percentage of performance improvements (I) of the DGA over HGA.

From the results obtained and presented in the Table 4, one can observe that our proposed DGA outperforms the HGA in terms of the obtained average values. The algorithm's performance in the case of DGA is increased by at least 4.02% in the case of

TABLE 5. Statistical results of 30 runs obtained by HGA and DGA in the case of a chromosomal size of 25

Test function	Global minimum	Population size	Avg.val. HGA	Time	Population size	Avg.val. DGA	Time	I [%]
F1	0	2000	15.3376	18.306	1000	13.2034	13.918	13.91
F2	0	2000	2.9548	17.951	1000	1.6469	14.362	44.26
F3	0	2000	1.7148	17.938	1000	1.6431	13.951	4.18
F4	0	2000	0.2898	18.531	1000	0.219	15.033	24.43
F5	0	2000	34.2007	16.895	1000	23.3992	12.261	31.58
F6	0	2000	0.1601	18.637	1000	0.1172	14.176	26.80
F7	0	2000	0.0248	16.427	1000	0.0222	11.967	10.48
F8	0	2000	26.00	16.609	1000	15.00	12.527	42.31
F9	0	2000	0.0231	17.025	1000	0.0110	11.969	52.38
F10	0	2000	0.0045	15.885	1000	0.0026	12.010	42.22
F11	0*	2000	3.8643	16.896	1000	2.6508	11.921	31.40
F12	0	2000	4.0369	17.462	1000	3.5564	12.179	11.90
F13	-1	2000	-0.2220	12.162	1000	-0.5981	10.923	48.34
F14	0.9	2000	1.0039	20.822	1000	1.0005	16.739	0.34
F15	0	2000	0.0165	19:922	1000	0.0141	15.449	14.55
F16	$-\gamma$	2000	-4.0865	14.140	1000	-4.5769	10.948	53.68
F17	0	2000	0.2632	24.931	1000	0.1850	14.004	29.71

Griewank's function, reaching up to 92.71% for the Schwefel's Double-Sum test function, in 11 out of 17 test functions, the percentage of performance improvement is over 24.43%. Regarding the running times, we notice that the computational times of the DGA are comparable to those obtained in the case of running the HGA.

Analyzing the results presented in Table 5, we can observe that the DGA outperforms HGA in terms of the obtained average values. The algorithm's performance in the case of DGA is increased by at least 0.34% in the case of the Periodic function, reaching up to 53.68% for the Ridge test function, in 11 out of 17 test functions, the percentage of performance improvement is over 21.63%. Regarding the running times, we can observe that the computational times of the DGA are smaller as compared to those reported in the case of running the HGA.

Taking a closer look at the results displayed in Table 6, we can remark that DGA outperforms HGA in terms of the achieved average values, also in the case of a chromosomal size of 50, a population size of 2000 individuals for HGA, respectively 1000 individuals for DGA. The performance of the algorithm in the case of DGA is increased by at least 2.07% in the case of the Periodic function, reaching up to 72.80% for the Rastrigin test function, and in 11 out of 17 test functions, the percentage of performance improvement is over 25.00%. Regarding the running times, we notice that the computational times of the DGA are comparable to those obtained in the case of running the HGA.

Analyzing the results achieved and displayed in Table 7, we can observe that DGA outperforms HGA in terms of the achieved average values, also in the case of a chromosomal size of 50, a population size of 2000 individuals for HGA, respectively 1000 individuals for DGA. The performance of the algorithm in the case of DGA is increased by at least 1.11% in the case of the Periodic function, reaching up to 67.14% for the Griewank test function, and in 10 out of 17 test functions, the percentage of performance improvement is over 30.32%. Regarding the running times, we can observe that in 12 out of 17 test functions, the computational times of the DGA are smaller as compared to those reported in the case of running the HGA and for the other cases are slightly higher.

TABLE 6. Statistical results of 30 runs obtained by HGA and DGA in the case of a chromosomal size of 50

Test function	Global minimum	Population size	Avg.val. HGA	Time	Population size	Avg.val. DGA	Time	I [%]
F1	0	1000	312.144	14.944	500	184.2350	16.521	40.98
F2	0	1000	6.1028	14.987	500	4.8956	14.962	19.78
F3	0	1000	31.8316	14.822	500	8.6588	16.254	72.80
F4	0	1000	6.6450	14.978	500	4.3256	15.210	34.90
F5	0	1000	286.7859	12.015	500	168.2563	14.323	41.33
F6	0	1000	1.9785	14.179	500	1.2620	15.336	36.21
F7	0	1000	1.1297	11.621	500	1.0234	12.985	9.41
F8	0	1000	382.00	12.026	500	233.2563	12.035	38.94
F9	0	1000	1.4719	11.903	500	0.6254	11.998	57.51
F10	0^*	1000	0.1072	12.029	500	0.0969	12.655	9.61
F11	0	1000	75.8249	11.686	500	73.6635	13.665	2.85
F12	0	1000	684.3180	11.828	500	287.1092	12.194	58.04
F13	-1	1000	-0.0107	11.570	500	-0.0896	11.736	7.98
F14	0.9	1000	1.0881	19.530	500	1.0656	22.652	2.07
F15	0	1000	0.0536	16.211	500	0.0402	21.596	25.00
F16	$-\gamma$	1000	-1.4667	11.109	500	-0.4346	11.994	42.58
F17	0	1000	19.6610	13.135	500	13.4528	14.358	31.58

TABLE 7. Statistical results of 30 runs obtained by HGA and DGA in the case of a chromosomal size of 50

Test function	Global minimum	Population size	Avg.val. HGA	Time	Population size	Avg.val. DGA	Time	I [%]
F1	0	2000	226.0581	40.025	1000	77.1097	48.676	65.89
F2	0	2000	4.8745	58.325	1000	3.3964	48.092	30.32
F3	0	2000	18.1867	61.023	1000	16.0374	47.932	11.82
F4	0	2000	4.0228	58.766	1000	1.3217	49.499	67.14
F5	0	2000	148.3650	62.388	1000	87.9287	39.060	40.73
F6	0	2000	1.3847	59.234	1000	0.8361	46.111	39.62
F7	0	2000	0.8524	60.486	1000	0.3274	38.890	61.59
F8	0	2000	141.652	49.332	1000	88.001	40.721	37.88
F9	0	2000	0.3954	65.114	1000	0.2360	39.357	40.31
F10	0^*	2000	0.0885	40.832	1000	0.0709	39.819	19.89
F11	0	2000	75.0332	38.215	1000	69.3935	39.256	7.52
F12	0	2000	214.9531	47.032	1000	71.0618	41.456	66.94
F13	-1	2000	-0.0102	36.002	1000	-0.1056	37.640	9.64
F14	0.9	2000	1.0032	61.205	1000	0.9921	61.908	1.11
F15	0	2000	0.0269	53.022	1000	0.0224	51.87	16.73
F16	$-\gamma$	2000	-0.6915	41.214	1000	-2.2185	38.358	31.67
F17	0	2000	8.2110	47.322	1000	4.2581	43.862	48.14

A further conclusion of this experiment is that DGA outperforms HGA in terms of the quality of the achieved results even when the size of the population of the DGA is halved, while the computational effort of both genetic algorithms are quite similar.

4.3. **Experiments 2. Haploid vs Diploid GAs Behaviour on Different Crossover Operators.** The purpose of this experiment was the study of the behavior of genetic algorithms, both haploid and diploid, under the conditions of major modifications of the crossing operator. To perform this experiment seven variants of this operator have been used. Each experiment was repeated 30 times with random values in every run and mean of the best values produced by each algorithm have been recorded. Also, standard deviation of the best values were computed. In all experiments, the values of the parameters used by each algorithm were chosen to be the same, so the population size was 1000 individuals, mutation rate was 0.05 and the number of generations for each run was 200. In both investigated GAs: haploid and diploid, the selection was made by the roulette wheel technique and the size of each chromosome was 25 genes.

Tables 8 - 10 have the following structure: the first two columns indicate the used test function and its corresponding global minimum, the next column displays the used crossover operator, for each test function we investigated seven different operators, the next two columns provide the average values obtained by the HGA, respectively DGA, and the last column displays the percentage of performance improvements (I) of the DGA over the HGA.

Analyzing the displayed results in Tables 8 -10, the following observations emerge:

(1) We observe a significant improvement of the average results obtained by using different crossover operators, in both investigated GAs: HGA and DGA, in some situations even reaching the global minimum value of the test function.
(2) In 12 out of 17 test functions (F1-F4, F6-F10, F12, F14 , F15), using a particular crossover technique combined with the use of diploidy can lead to results whose average values are very close to the global minimum of the test function.
(3) In 9 out of 17 test functions (F2-F4, F6, F7, F9, F10, F15, F17) the BLX crossing technique combined with the use of diploidy lead to an improvement of the average values of the DGA over HGA of more than 90%.
(4) For all the considered test functions and crossover operators, the DGA provided better average values in comparison to those achieved by the HGA, except four cases when both DGA and HGA obtained the global minimum: in the case of Rastrigin's Function with the crossover operator $BLX - 0.3$ and in the case of the Step function with the crossover operators: $BLX - 0, BLX - 0.3$ and $BLX - 0.5$.

Overall, the comparison between the investigated GA's: diploid genetic algorithms and haploid genetic algorithms can be summarized as follows:

(1) In terms of quality of the achieved results, the DGA outperforms HGA in the case of all considered test functions. The main reason is that the DGA covers the search space better and has a slower convergence, which makes the algorithm avoid local optima. The improvements of DGA over HGA vary from 8.82% to 94.67%.
(2) The search times for DGA are longer than the corresponding ones in the case of HGA due to the slower convergence. However, when comparing the quality of the results and the execution time, there is a balance in favour of quality. HGA's are faster than DGA's with a range from 4% to about 90%.
(3) We also investigated the performance of HGA and DGA with respect to various crossover operators. This is because we wanted to make sure that the performance of DGA is independent of the genetic operators, among which the crossover is the most important. We observed that the crossover operator plays an important role in the performance of both HGA and DGA and again the DGA outperformed the HGA in terms of the achieved average values.

TABLE 8. Statistical results of 30 runs obtained by HGA and DGA using different crossover methods in the case of the benchmark functions F1 - F6

Test fnction	Global minimum	Crossover operator	Avg.val. HGA	Avg.val. DGA	I [%]
F1	0	SX	577.3638	468.3567	18.88
		TPX	83.1695	67.2589	19.13
		UX	42.8536	31.6597	26.12
		AX	8185.9515	8138.2286	0.58
		BLX-0	9865.0504	8499.1507	13.85
		BLX-0.3	10339.19303	7324.8558	29.15
		BLX-0.5	9698.1175	6827.8118	29.60
		MMAX	5.0902	2.1623	57.52
		LX	1.2856	0.1063	91.73
F2	0	SX	5.7785	2.7724	52.02
		TPX	4.2745	2.2370	47.67
		UX	1.8792	0.9154	51.29
		AX	1.5649	0.0716	95.42
		BLX-0	0.0046	0.0016	65.22
		BLX-0.3	4.249E-4	7.972E-6	98.12
		BLX-0.5	3.474E-4	1.435E-6	99.59
		MMAX	0.3733	0.0255	93.17
		LX	1.1022	0.0012	99.89
F3	0	SX	23.7417	6.5783	72.29
		TPX	10.6452	5.0069	52.97
		UX	1.6074	1.5639	2.71
		AX	36.9054	1.9461	94.73
		BLX-0	0.2701	0.0067	97.52
		BLX-0.3	0.00	0.00	-
		BLX-0.5	1.037E-4	4.074E-8	99.96
		MMAX	0.4461	0.0127	97.15
		LX	5.5465	6.695E-4	99.99
F4	0	SX	3.2667	0.7273	77.74
		TPX	1.0358	0.4438	57.15
		UX	0.0879	0.0865	1.59
		AX	0.0678	0.0052	92.33
		BLX-0	1.131E-6	2.07E-7	81.70
		BLX-0.3	1.602E-11	1.122E-14	99.93
		BLX-0.5	2.743E-3	1.105E-7	99.60
		MMAX	0.0324	2.257E-4	99.30
		LX	0.0277	1.123E-6	99.99
F5	0	SX	166.2661	58.3750	64.89
		TPX	49.3213	41.0161	16.84
		UX	37.3251	33.4032	10.51
		AX	32.6086	29.0396	10.94
		BLX-0	28.4515	28.1295	1.13
		BLX-0.3	27.1132	27.0055	0.40
		BLX-0.5	26.8146	26.1092	2.63
		MMAX	29.9702	28.2408	5.77
		LX	31.2400	28.0932	10.07
F6	0	SX	1.0062	0.3617	64.05
		TPX	0.4734	0.2259	52.28
		UX	0.1560	0.0808	48.21
		AX	31.8549	23.4847	26.28
		BLX-0	7.2958	2.7135	62.81
		BLX-0.3	5.525E-7	7.701E-9	98.61
		BLX-0.5	4.142E-4	1.089E-5	97.37
		MMAX	0.0468	0.0318	32.05
		LX	8.7948	0.0606	99.31

TABLE 9. Statistical results of 30 runs obtained by HGA and DGA using different crossover methods in the case of the benchmark functions F7 - F12

Test fnction	Global minimum	Crossover operator	Avg.val. HGA	Avg.val. DGA	I [%]
F7	0	SX	0.4118	0.0892	78.34
		TPX	0.2595	0.0482	81.43
		UX	0.0484	0.0215	55.58
		AX	0.0139	2.909E-4	97.91
		BLX-0	4.128E-7	2.395E-8	94.20
		BLX-0.3	3.552E-13	3.455E-15	99.03
		BLX-0.5	9.314E-9	3.737E-11	99.60
		MMAX	0.0142	1.725E-5	99.88
		LX	0.0091	3.447E-7	100
F8	0	SX	241	61	74.69
		TPX	142	25	82.39
		UX	16	11	31.25
		AX	6	1	83.33
		BLX-0	0	0	-
		BLX-0.3	0	0	-
		BLX-0.5	0	0	-
		MMAX	2	0	100
		LX	1	0	100
F9	0	SX	0.4748	0.1244	73.80
		TPX	0.2532	0.0470	81.44
		UX	0.0252	0.0102	59.52
		AX	0.0069	5.402E-4	92.17
		BLX-0	1.563E-4	5.524E-6	96.47
		BLX-0.3	2.329E-12	2.491E-15	99.89
		BLX-0.5	5.137E-8	9.877E-10	98.08
		MMAX	0.0019	2.999E-5	98.42
		LX	0.0052	1.435E-6	99.97
F10	0*	SX	0.0486	0.0180	62.96
		TPX	0.0363	0.0125	65.56
		UX	0.0614	0.0301	50.98
		AX	0.0167	0.0045	73.05
		BLX-0	0.0172	6.829E-4	96.03
		BLX-0.3	0.0166	6.627E-5	99.60
		BLX-0.5	0.0397	0.0027	93.12
		MMAX	0.0084	9.544E-5	98.86
		LX	0.0208	3.732E-3	82.06
F11	0	SX	117.866	27.6948	76.50
		TPX	16.5189	6.8158	58.74
		UX	8.5718	6.9793	18.58
		AX	1736.887	1346.093	22.50
		BLX-0	1503.256	910.252	39.45
		BLX-0.3	1.0032	0.8962	10.67
		BLX-0.5	1.0008	0.9631	3.77
		MMAX	40.8264	12.6860	68.93
		LX	425.9237	106.3197	75.04
F12	0	SX	210.9759	10.8047	94.88
		TPX	75.1921	6.1684	91.80
		UX	2.8886	1.712	40.73
		AX	1.8930	0.6471	95.82
		BLX-0	0.5172	0.5016	3.02
		BLX-0.3	0.5006	0.5000	0.12
		BLX-0.5	0.5003	0.5000	0.06
		MMAX	1.0869	0.5076	53.30
		LX	1.3621	0.5115	62.45

TABLE 10. Statistical results of 30 runs obtained by HGA and DGA using different crossover methods in the case of the benchmark functions F13 - F17

Test fnction	**Global minimum**	**Crossover operator**	**Avg.val. HGA**	**Avg.val. DGA**	**I [%]**
F13	-1	SX	-0.1116	-0.4535	38.48
		TPX	-0.2039	-0.4645	32.73
		UX	-0.0535	-0.3349	29.73
		AX	-0.3441	-0.4516	16.39
		BLX-0	-0.1918	-0.6595	57.87
		BLX-0.3	-0.2957	-0.7096	58.77
		BLX-0.5	-0.2142	-0.6988	61.67
		MMAX	-0.0942	-0.5875	54.46
		LX	-0.1228	-0.6984	65.62
F14	0.9	SX	1.0806	1.0462	3.18
		TPX	1.0318	1.0031	2.78
		UX	1.0024	1.0007	0.17
		AX	4.3330	1.7297	60.08
		BLX-0	0.9098	0.9008	0.99
		BLX-0.3	0.9199	0.9	2.16
		BLX-0.5	1.0172	0.9985	1.84
		MMAX	1.3559	0.9019	33.48
		LX	1.2365	1.0322	16.52
F15	0	SX	0.1912	0.0093	95.14
		TPX	0.0469	0.0078	83.37
		UX	0.0062	3.234E-4	94.78
		AX	1.339E-6	1.041E-8	99.22
		BLX-0	6.521E-11	3.767E-12	94.22
		BLX-0.3	3.353E-27	7.689E-29	97.71
		BLX-0.5	1.073E-16	1.418E-18	98.68
		MMAX	7.291E-9	7.413E-11	98.68
		LX	1.815E-10	1.559E-11	91.41
F16	$-\gamma$	SX	1.1903	-3.6484	78.17
		TPX	-4.0438	-4.5941	57.55
		UX	-4.0319	-4.8226	81.68
		AX	-0.5362	-3.8226	73.62
		BLX-0	-1.5249	-1.6146	2.58
		BLX-0.3	-1.3541	-1.4326	2.15
		BLX-0.5	-1.6511	-1.7223	2.13
		MMAX	-1.0381	-1.7415	17.75
		LX	-0.1564	-0.683	10.87
F17	0	SX	7.2356	2.6254	63.72
		TPX	4.1252	0.2569	93.77
		UX	1.3265	0.2422	81.74
		AX	1.0233	0.2588	74.71
		BLX-0	0.9523	0.0624	93.45
		BLX-0.3	0.1245	0.0124	90.04
		BLX-0.5	0.2322	0.1022	55.99
		MMAX	0.8526	0.0414	95.14
		LX	0.4125	0.1203	70.84

5. CONCLUSIONS AND FURTHER RESEARCH

The article makes a comprehensive comparison between haploid genetic algorithms (HGA) and diploid genetic algorithms (DGA). We investigated both representations from all perspective, to make sure that no specific configuration aspect influences the results.

The research follows a strict methodology and attempts to investigate: (1) the quality of the achieved results; (2) the necessary computational times in order to achieve the solutions; (3) the impact of the genetic operators on the quality of the results.

We concluded that the quality of the results achieved by running DGA is definitely better than those obtained in the case of conventional GAs (HGA). On the other hand HGA is faster than DGA. However, comparing the quality of the results and the search time, the balance is in favour of DGA, meaning that, although the number of chromosomes is double, the execution time is is less than double, comparing with HGA.

We have proven that DGA is always better than HGA, independent of the genetic operators. The reason for the better performance of DGA is given by a more thorough search of the space. The best individuals do not have enough power to draw the population in local optima, which are avoided this way.

We have proven that DGA is always better than HGA in terms of the quality of the achieved results, independent of the genetic operators or the problem in stake. The reason for achieving a better performance is based on a better exploration of the space and thus by a more thorough search of it. This yields from the considered phenotype of the individual, which was chosen as the average of the two chromosomes, assuring an even influence of the two chromosomes on the survival chances of the individual.

The diploid population is more compact, its amplitude is lower and the number of chromosomes is double, which leads to a exploration of the solution space more thoroughly. The best individuals do not have enough power to draw the population to local optima, which are avoided this way.

Based on the fascinating results achieved by the diploid genetic algorithms and presented in this paper, in the near future we plan to investigate the following research directions:

(1) the use of DGAs for solving different complex optimization problems;
(2) to explore and compare poly-ploid genetic algorithms against DGAs and HGAs;
(3) to investigate different ways of defining the phenotype of the DGA.

References

[1] Ali, M. M.; Khompatraporn, C.; Zabinsky, Z. B. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. *J Glob Optim.* **31** (2015), 635–672.

[2] Bhasin, H.; Mehta, S. *On the applicability of diploid genetic algorithms*. AI & Society, **31** (2016), 265–274.

[3] Bhasin, H.; Behal, G.; Aggarwal, N.; Saini, R. K.; Choudhary, S. On the applicability of diploid genetic algorithms in dynamic environments. *Soft Computing* **20** (2016), 3403–3410.

[4] Bull, L. Haploid-diploid evolutionary algorithms: The Baldwin effect and recombination nature's way. in *Proc. of Swarm Intelligence and Evolutionary Computation Symposium* (2017).

[5] Cobb, H. G.; Grefenstette, J. J. Genetic algorithms for tracking changing environments. in *Proc. 5th Int. Conf. Genetic Algorithms* 1993, 523–530.

[6] Collingwood, E.; Corne, D.; Ross, P. Useful diversity via multiploidy. in *Proc. of IEEE International Conference on Evolutionary Computation* (1996), 810–813.

[7] Cosma, O.; Pop, P. C.; Pop Sitar, C. A two-level based genetic algorithm for solving the soft-clustered vehicle routing problem. *Carpathian J. Math.* **38** (2022), no. 1, 117–128.

[8] Cosma, O.; Pop, P. C.; Zelina, I. A novel genetic algorithm for solving the clustered shortest-path tree problem. *Carpathian J. Math.* **36** (2020), no. 1, 401–414.

[9] Cosma, O.; Pop, P. C.; Sabo, C. An Efficient Hybrid Genetic Approach for Solving the Two-Stage Supply Chain Network Design Problem with Fixed Costs. *Mathematics* **8** (2020), no. 5, 712.

[10] Darwin, C. *On the Origin of Species by Means of Natural Selection. Or the Preservation of Favoured Races in the Struggle for Life* 1859.

[11] De Jong, K. A. *An analysis of the behavior of a class of genetic adaptative systems*. Ph.D. dissertation, University of Michigan, USA, 1975.

[12] Deb, K.; Goldberg, D. E. An investigation of niche and species formation in genetic function optimization. in *Proc. of the 3rd International Conference on Genetic Algorithms* (1989), 42–50.

[13] Dieterich, J. M.; Hartke, B. Empirical review of standard benchmark functions using evolutionary global optimization. *Applied Mathematics* **3** (2012), 1552–1564.

[14] Dulebenets, M. A. A diploid evolutionary algorithm for sustainable truck scheduling at a cross-docking facility. *Sustainability* **10** (2018), 1333.

[15] Dulebenets, M. A. An Adaptive Polyploid Memetic Algorithm for scheduling trucks at a cross-docking terminal. *Information Sciences* **565** (2021), 390–421.

[16] Eshelman, L. J.; Schaffer, J. D. Real-coded genetic algorithms and interval-schemata. *Foundations of genetic algorithms* **2** (1993), 187–202.

[17] Galan, S. F.; Mengshoel, O. J.; Pinter, R. A Novel Mating Approach for Genetic Algorithms. *Evolutionary Computation* **21** (2013), no. 2, 197–229.

[18] Garcia-Pedrajas, N.; Martínez, C. H.; Ortiz-Boyer, D. CIXL2: A Crossover Operator for Evolutionary Algorithms Based on Population Features. *J Artif Intell Res.* **24** (2005), 1–48.

[19] Goldberg, D. E.; Smith, R. E. Nonstationary function optimization using genetic algorithms with dominance and diploidy. in *Proc. of the Second International Conference on Genetic Algorithms* (1987), 59–68.

[20] Griewank, A. O. Generalized Descent for Global Optimization. *J Optim Theory Appl.* **34** (1981), no. 1, 11–39.

[21] Herrera, F.; Lozano, M.; Sanchez, A. M. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems* **18** (2003), 309–338.

[22] Holland, J. H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[23] Jasuja, A. Feature Selection Using Diploid Genetic Algorithm. *Annals of Data Science* **7** (2020), no. 1, 33–43.

[24] Jamil, M.; Yang, X. S. A literature survey of benchmark functions for global optimization problems. *Int. J. Math. Model. Numer. Optim.* **4** (2013), no. 2, 150–194.

[25] Kremer, I.; Yishay, M.; Motty, P. Implementing the "wisdom of the crowd". *J. Polit. Econ.* **122** (2014), no. 5, 988–1012.

[26] Liekens, A.; Eikelder, H.; Hilbers, P. Modeling and simulating diploid simple genetic algorithms. in *Proc. of Foundations of Genetic Algorithms VII* (2003), 151–168.

[27] Matei, O.; Pop, P. C.; Sas, I; Chira, C. An improved immigration memetic algorithm for solving the heterogeneous fixed fleet vehicle routing problem. *Neurocomputing* 150A (2015), 58–66.

[28] Matei, O. *Evolutionary computation: principles and practices*. Risoprint, 2008.

[29] Mitchell, M. *An introduction to genetic algorithms*. MIT Press, USA, 1998.

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

[31] Ng, K. P.; Wong, K. C. A New Diploid Scheme and Dominance Change Mechanism for Non-Stationary Function Optimization. in *Proc. of the 6th International Conference on Genetic Algorithms* (1995), 159–166.

[32] Pencheva, T.; Atanassov, K.; Shannon, A. Modelling of a roulette wheel selection operator in genetic algorithms using generalized nets. *Int. J. Bioautomation* **13** (2009), 257–264.

[33] Petrovan, A.; Matei, O.; Pop, P. C. Haploid Versus Diploid Genetic Algorithms. A Comparative Study. in *Proc. of International Conference on Hybrid Artificial Intelligence Systems* (2019), 193–205.

[34] Petrovan, A.; Matei, O.; Erdei, Z. A Behavioural Study of the Crossover Operator in Diploid Genetic Algorithms, in *Proc. of 15th International Conference on Soft Computing Models in Industrial and Environmental Applications* (2020), 79–88.

[35] Pop, P. C.; Matei, O.; Sabo, C. A hybrid diploid genetic based algorithm for solving the generalized traveling salesman problem, in *Proc. of Hybrid Artificial Intelligence Systems* 10334 (2017), 149–160.

[36] Pop, P. C.; Matei, O.; Sabo, C.; Petrovan, A. A two-level solution approach for solving the generalized minimum spanning tree problem. *European J. Oper. Res.* **265** (2018), no. 2, 478–487.

[37] Pop, P. C.; Matei, O.; Pintea, C. A Two-level Diploid Genetic Based Algorithm for Solving the Family Traveling Salesman Problem, in *Proceedings of the Genetic and Evolutionary Computation Conference* (2018), 340–346.

[38] Rezaeipanah, A.; Matoori, S. S.; Ahmadi, G. A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. *Applied Intelligence* **51** (2021), 467–492.

[39] Sareni, B.; Krähenbühl, L. Fitness Sharing and Niching Methods Revisited. *IEEE Transactions on Evolutionary Computation* **2** (1998), no. 3, 97–106.

[40] Sahin, C. B.; Dinler, O. B.; Abualigah, L. Prediction of software vulnerability based deep symbiotic genetic algorithms: Phenotyping of dominant-features. *Applied Intelligence* **51** (2021), 8271–8287.

[41] Shi, S.; Xiong, H. A hybrid immune genetic algorithm with tabu search for minimizing the tool switch times in CNC milling batch-processing. *Applied Intelligence* (2021) https://doi.org/10.1007/s10489-021-02869-3.

[42] Schlierkamp-Voosen, D.; Mühlenbein, H. Strategy adaptation by competing subpopulations. in *Proc. of International Conference on Parallel Problem Solving from Nature* (1994), 199–208.

[43] Schwefel, H.-P. *Numerical Optimization for Computer Models* John Wiley & Sons, New York, USA, 1981.

[44] Thorgaard, G. H. Ploidy manipulation and performance. *Aquaculture* **57** (1986), no. (1-4), 57–64.

[45] Yang, S. On the design of diploid genetic algorithms for problem optimization in dynamic environments. in *Proceedings of IEEE International Conference on Evolutionary Computation* (2006) 1362–1369.
[46] Yang, S.; Yao, X. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing* **9** (2005), 815–834.
[47] Yukiko, Y.; Nobue, A. A diploid genetic algorithm for preserving population diversity — Pseudo-Meiosis GA. in *Proc. of Parallel Problem Solving from Nature*. Lecture Notes in Computer Science **866** (1994), 36–45.

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
NORTH UNIVERSITY CENTER OF BAIA MARE, ROMANIA
Email address: adrian.petrovan@ieec.utcluj.ro
Email address: oliviu.matei@ieec.utcluj.ro

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
NORTH UNIVERSITY CENTRE AT BAIA MARE, ROMANIA
Email address: petrica.pop@mi.utcluj.ro