MDPI

Article

Introducing a parallel genetic algorithm for global optimization problems

Vasileios Charilogis[1], Ioannis G. Tsoulos[2,*]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilog@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; itsoulos@uoi.gr
* Correspondence: itsoulos@uoi.gr;
† Current address: Department of Informatics and Telecommunications, University of Ioannina, Greece.
‡ These authors contributed equally to this work.

Abstract: The topic of efficiently finding the global minimum of multidimensional functions finds widespread use in a multitude of problems in the modern world. A multitude of algorithms have been proposed to solve the problems, and Genetic Algorithms and their various variants occupy an excellent position among them. Their popularity stems from their exceptional performance in identifying effective solutions for optimization problems as well as because of their adaptability to various kinds of problems. However, Genetic Algorithms require significant computational resources and time, prompting the need for parallel techniques. Moving in this research direction, a new global optimization method is presented here that exploits the use of parallel computing techniques in Genetic Algorithms. This innovative method employs autonomous parallel computing units, periodically sharing the optimal solutions they discover. Increasing the number of computational threads, coupled with solution exchange techniques, can significantly reduce the number of calls to the objective function, thus saving computational power. Also, a stopping rule is proposed that takes advantage of the parallel computational environment. The proposed method was tested on a wide series of benchmark functions from the relevant literature, and it is compared against other global optimization techniques regarding its efficiency.

Keywords: Parallel techniques; Global optimization; Genetic algorithms; Evolutionary techniques

Citation: Charilogis V.; Tsoulos I.G.; Introducing a parallel genetic algorithm for global optimization problems. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

1. Introduction

Typically the task of locating the global minimum [1] of a function $f : S \to R, S \subset R^n$ is defined as:

$$x^* = \arg\min_{x \in S} f(x). \quad (1)$$

where the set S has as follows:

$$S = [a_1, b_1] \otimes [a_2, b_2] \otimes \ldots [a_n, b_n]$$

The values a_i and b_i are the left and right bounds respectively for the point x_i. A systematic review of the optimization procedure can be found in the work of Fouskakis [2].

The previous defined problem has been tackled using a variety of methods, which have been successfully applied to a wide range of problems in various fields, such as Medicine [3,4], Chemistry [5,6], Physics [7–9], Economics [10,11], etc. Global optimization methods are divided into two main categories: deterministic and stochastic methods [12]. In the first category belong the interval methods [13,14], where the set S is iteratively divided into subregions and those that do not contain the global solution are discarded based on predefined criteria. Many related works have been published in the area of deterministic methods, such as the work of Maranas and Floudas that proposed a deterministic method for chemical problems [15], the TRUST method [16], the method suggested by Evtushenko

and Posypkin[17] etc. In the second category, the search for the global minimum is based on randomness. Also, stochastic optimization methods are used in most cases, since they can be programmed more easily and they do depend on any previous information about the objective problem. Some stochastic optimization methods that have been used by researchers include Ant Colony Optimization [18,19], Controlled Random Search [20–22], Particle Swarm Optimization [23–25], Simulated Annealing [26–28], Differential Evolution [29,30], and genetic algorithms [31–33]. Finally, there is a plethora of research referring to metaheuristic algorithms [34–36], offering new perspectives and solutions to problems in various fields.

The current work proposes a series of modifications in order to effectively parallelize the widely adopted method of Genetic Algorithms for solving the equation 1. Genetic algorithms, initially proposed by John Holland, constitute a fundamental technique in the field of stochastic methods[37]. Inspired by biology, these algorithms simulate the principles of evolution, including genetic mutation, natural selection, and exchange of genetic material [38–40]. The integration of genetic algorithms with machine learning has proven effective in addressing complex problems and validating models. This interaction is highlighted in applications such as the design and optimization of 5G networks, contributing to path loss estimation and improving performance in indoor environments [41]. It is also applied to optimizing the movement of digital robots [42] and conserving energy in industrial robots with two arms [43]. Additionally, genetic algorithms have been employed to find optimal operating conditions for motors [44], optimize the placement of electric vehicle charging stations [45], manage energy [46], and have applications in other fields such as medicine [47,48] and agriculture [49].

Although genetic algorithms have proven to be effective, the optimization process requires significant computational resources and time. This emphasizes the necessity of implementing parallel techniques, as the execution of algorithms is significantly accelerated by the combined use of multiple computational resources. Modern parallel programming techniques include for example the Message Passing Interface (MPI) [50] or the OpenMP library [51]. Parallel programming techniques have also been incorporated in various cases into global optimization, such as the combination of Simulated Annealing and parallel techniques [52], the usage of parallel methods in Particle Swarm Optimization[53], the incorporation of radial basis functions in parallel stochastic optimization [54] etc. One of the main advantages of Genetic Algorithms over other global optimization techniques is that they can be easily parallelized and exploit modern computing units as well as the previously mentioned parallel programming techniques.

In the relevant literature, two major categories of parallel genetic algorithms appear: island genetic algorithms and cellular genetic algorithms [55]. The island model is a Parallel Genetic Algorithm (PGA), that manages several subpopulations on separate islands, executing the genetic algorithm process on each island simultaneously for a different set of solutions. Island models have been utilized in various cases, such as molecular sequence alignment [56], the quadratic assignment problem [57], placement of sensors/actuators in large structures [58] etc. Also, recently Tsoulos et al. proposed an implementation of an island PGA [59]. In the case of the parallel cellular model of genetic algorithms, solutions are organized into a grid. Various diverse operators, such as crossover and mutation, are applied to neighboring regions within the grid. For each solution, a descendant factor is created, replacing its position within the birth region. The model is flexible regarding the structure of the grid, neighborhood strategies, and settings. Implementations may involve multiple processors or graphical processing units, with information exchange possible through physical communication networks. The theory of parallel genetic algorithms has been thoroughly presented by a number of researchers in the relevant literature [60,61]. Also, parallel genetic algorithms have been incorporated in combinatorial optimization [62].

The proposed method is based on the island technique and suggests a number of improvements to the general scheme of parallel Genetic Algorithms. Among these improve-

ments are a series of techniques for propagating optimal solutions among islands that aim to speed up the convergence of the overall algorithm. In addition, on the individual islands of the genetic algorithm, a local minimization technique is periodically applied that has two goals: to discover with the maximum possible accuracy local minima of the objective function but also to speed up the convergence of the overall algorithm, without wasting computing power on values of function that have already been discovered. Furthermore, an efficient termination rule, which is based on asymptotic considerations and it was validated in a series of global optimization methods is also incorporated in the current algorithm. The proposed method was applied to a series of problems appearing in the relevant literature. From the experimental results, it was found that the new method is able to effectively find the global minimum of the functions in a large percentage and moreover the above modifications significantly accelerated the finding of the global minimum as the number of individual islands in the Genetic Algorithm increases.

The remaining of the article follows this structure: In section 2, the genetic algorithm is analyzed, its parallelization is discussed, as well as dissemination techniques (PT or migration methodologies), and the termination criterion. Subsequently, in section 3, the test functions used are presented in detail, along with the experimental results. Finally, in section 4, some conclusions are outlined, and future explorations are formulated.

2. Method description

This section initiates with a detailed description of the base genetic algorithm and continues providing the details of the suggested modifications.

2.1. The Genetic Algorithm

Genetic algorithms are inspired by natural selection and the process of evolution in nature. In their basic form, they start with an initial population of chromosomes, representing possible solutions to a specific problem. Each chromosome is represented as a "gene", and its length is equal to the dimension of the problem. The algorithm processes these solutions through iterative steps, replicating and evolving the population of solutions. In each generation, the selected solutions are crossed and mutated to improve their fit to the problem. As generations progress, the population converges toward solutions with improved fit to the problem. Important factors affecting genetic algorithm performance include population size, selection rate, crossover and mutation probabilities, and strategic replacement of solutions. The choice of these parameters affects the ability of the algorithm to explore the solution space and converge to the optimal result. Subsequently, the operation of the genetic algorithm is presented through the replication and advancement of solution populations step by step[68,69].:

1. **Initialization step**.
   (a) **Set** N_c as the number of chromosomes.
   (b) **Set** N_g the maximum number of allowed generations.
   (c) **Initialize** randomly N_c chromosomes in S. Each chromosome denotes a potential solution to the problem of Equation 1.
   (d) **Set** as p_s the selection rate of the algorithm, with $p_s \leq 1$.
   (e) **Set** as p_m the mutation rate, with $p_m \leq 1$.
   (f) **Set** k=0 as the generation counter.
2. **Fitness calculation step.**
   (a) **For** every chromosome g_i, $i = 1, \ldots, N_c$ **Calculate** the fitness $f_i = f(g_i)$ of chromosome g_i.
3. **Selection step**. The chromosomes are sorted with respect to their fitness values. Denote as N_b the integer part of $(1 - p_s) \times N_c$ chromosomes with the lowest fitness values. These chromosomes will copied to the next generation. The rest of chromosomes will be substituted by offsprings created in the crossover procedure. Each offspring is created from two chromosomes (parents) of the population through the

tournament selection process. The procedure of tournament selection has as follows: A set of $N_t > 1$ randomly selected chromosomes is formed and the individual with the lowest fitness value from this set is selected as parent.

4. **Crossover step**. Two selected solutions (parents) are combined to create new solutions (offspring). During crossover, genes are exchanged between parents, introducing diversity. For each selected pair of parents (z, w), two additional chromosomes, represented by $\tilde{z}$ and $\tilde{w}$, are generated through the following equations.

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) w_i \\ \tilde{w}_i &= a_i w_i + (1 - a_i) z_i \end{aligned} \tag{2}$$

where $i = 1, \ldots, n$. The values a_i are uniformly distributed random numbers, with $a_i \in [-0.5, 1.5]$ [70].

5. **Replacement step**.
   (a) **For** $i = N_b + 1$ to N_c **do**
      i. **Replace** g_i using the next offspring created in the crossover procedure.
   (b) **EndFor**
6. **Mutation step**. Some genes in the offspring are randomly modified. This introduces more diversity into the population and helps identify new solutions.
   (a) **For** every chromosome g_i, $i = 1, \ldots, N_c$ do
      i. **For** each element $j = 1, \ldots, n$ of g_i a uniformly distributed random number $r \in [0, 1]$ is drawn. The element is altered randomly if $r \leq p_m$.
   (b) **EndFor**
7. **Set** $k = k + 1$. If the termination criterion defined in the work of Tsoulos [72], which is outlined in subsection 2.3, is met or $k > N_g$ then goto Local Search step also goto to step 2a.
8. **Local Search step**. For improving the success in finding better solutions, a process of local optimization search takes place. In the present study, the Broyden Fletcher Goldfarb Shanno (BFGS) variant proposed by Powell [71] was employed as the local search procedure. This procedure is applied to the chromosome in the population with the lowest fitness value.

2.2. Parallelization of Genetic Algorithm and Propagation techniques

In the parallel island model of Figure 1, an evolving population is divided into various "islands", each working concurrently to optimize a specific set of solutions. In this figure each island implements a separate genetic algorithm as described in subsection 2.1. The steps of the overall algorithm are also preseneted through a series of steps in Algorithm 1. In contrast to classical parallelization, which handles a central population, the island model features decentralized populations evolving independently. Each island exchanges information with others at specific points in evolution through migration, where solutions move from one island to another, influencing the overall convergence toward the optimal solution. Migration settings determine how often it occurs and which solutions are selected for exchange. Each island can follow a similar search strategy, but for more variety or faster convergence, different approaches can be employed. Islands may have identical or diverse strategies, providing flexibility and efficiency in exploring the solution space. To implement this parallel model, each island is connected to a computational resource. For instance, as depicted in images of Figure 2, the execution of the parallel islands model involves five islands, each managing a distinct set of solutions using five processor units (PU). During the migration process, information related to solutions is exchanged among PU. In the same Figure 2, the four different techniques for spreading the chromosomes with the best functional values are depicted. In Figure 2a, we observe the migration of the best chromosomes from one island to another randomly chosen. In Figure 2b, from a randomly

chosen island to all others, in Figure 2c, from all islands to a randomly chosen one, and finally, in Figure 2d, migration occurs from each island to all the others.

Algorithm 1 The overall algorithm

1. **Set** as N_I the total number of parallel processing units.
2. **Set** as N_R as the number of generations, after which each processing unit will send its best chromosomes to the remaining processing units.
3. **Set** N_P as the number of migrated chromosomes between the parallel processing units.
4. **Set** PT as propagation technique.
5. **Set** $k = 0$ the generation number.
6. **For** $j = 1, .., N_I$ do in parallel
   (a) **Execute** an generation of the GA algorithm described in algorithm 2.1 on processing unit j.
   (b) **If** K **mod** $N_R = 0$,**then**
      i. **Get** the best N_P chromosomes from algorithm j.
      ii. **Propagate** these N_P chromosomes to the rest of processing units using some propagation scheme that will be described subsequently.
   (c) **EndIf**
7. **End For**
8. **Update** $k = k + 1$
9. **Check** the proposed termination rule. If the termination rule is valid, then goto step 9a else goto step 6.
   (a) **Terminate** and report the best value from all processing units. Apply a local search procedure to this located value to enhance the located global minimum.

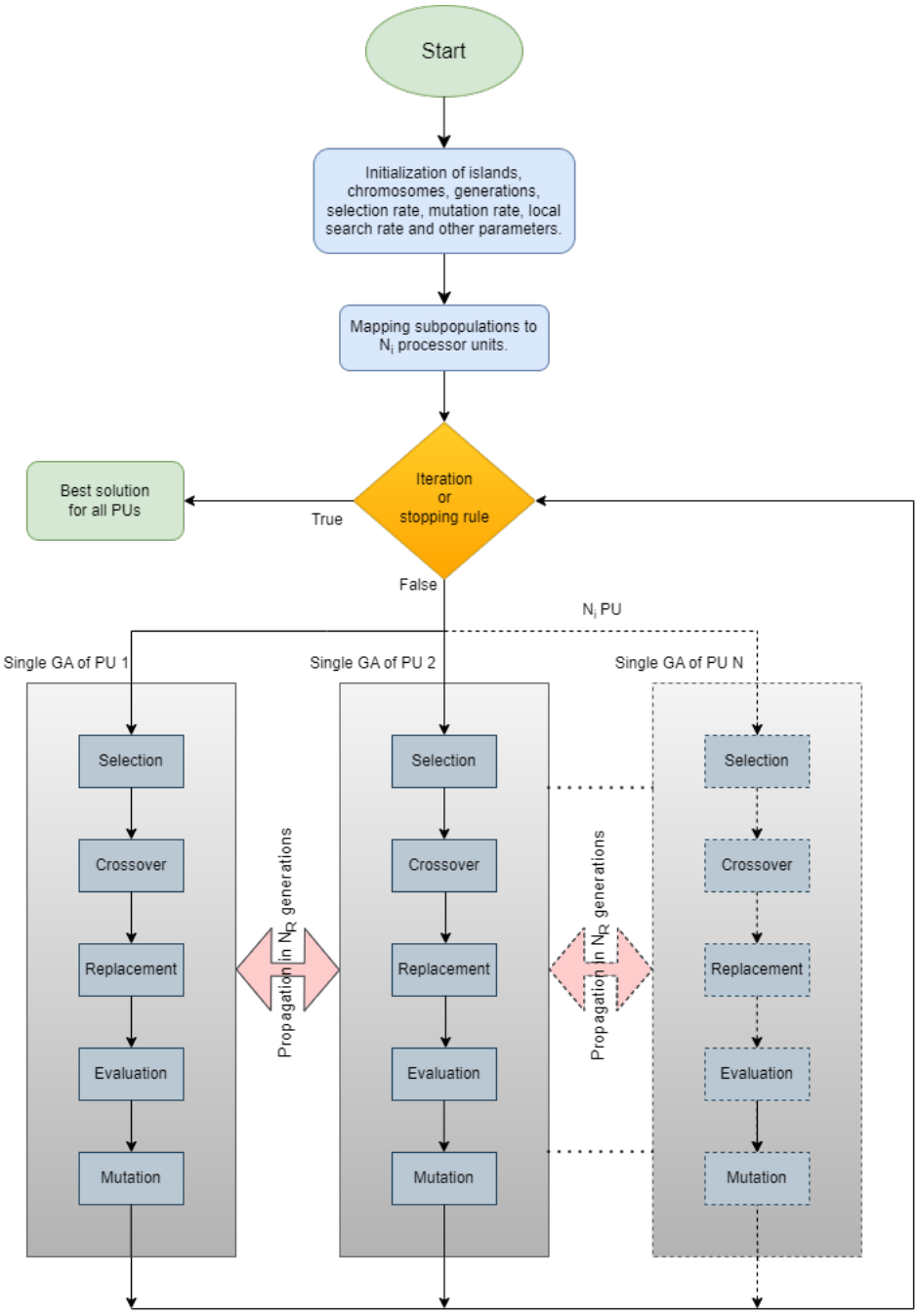
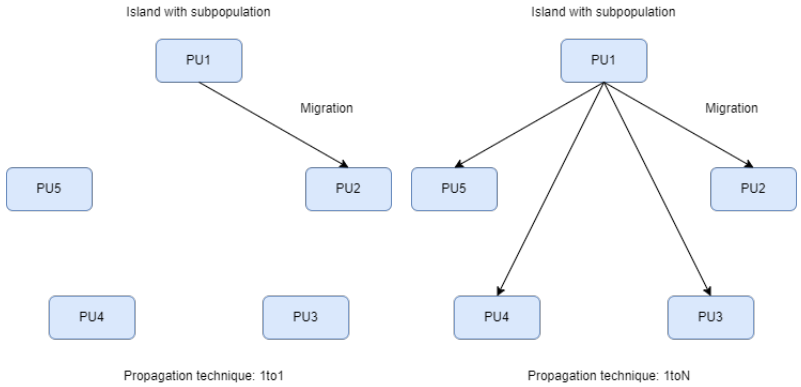


Figure 1. Parallelization of GA

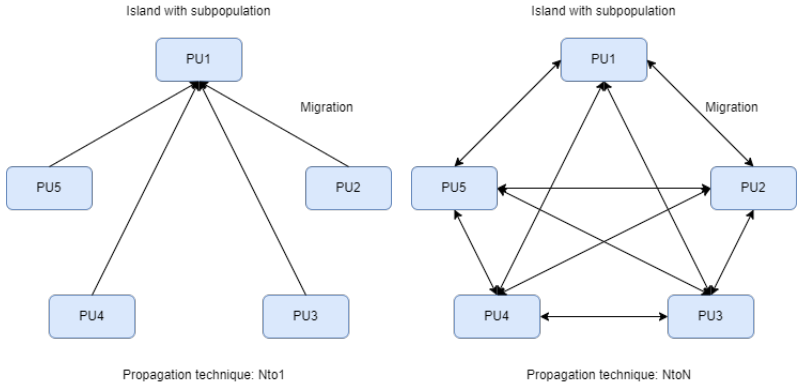
The migration or propagation techniques, as described in this study, are periodically and synchronously performed in N_R iterations on each processing unit. Below are the migration techniques that could be defined:

- 1to1: Optimal solutions migrate from a random island to another random one, replacing the worst solutions (see figure 2a).
- 1toN: Optimal solutions migrate from a random island to all others, replacing the worst solutions (see figure 2b).
- Nto1: All islands send their optimal solutions to a random island, replacing the worst solutions (see figure 2c).
- NtoN: All islands send their optimal solutions to all other islands, replacing the worst solutions (see figure 2d).

If we assume that the migration method "1toN" is executed, then a random island will transfer chromosomes to the other islands, except for itself. However, we kept the label "N" instead of "N-1" because the chromosomes exist on the island that sends them.The number of solutions participating in the migration and replacement process is fully customizable and will be referred to in the experiments below.



(a) Propagation 1to1 **(b)** Propagation 1toN



(c) Propagation Nto1 **(d)** Propagation NtoN

Figure 2. Islands and propagation

2.3. Termination rule

The termination criterion employed in this study was originally introduced in the research conducted by Tsoulos [72] and it is formulated as follows:

- In each generation k, the chromosome g^* with the best functional value $f(g^*)$ is retrieved from the population. If this value does not change for a number of generations, then the algorithm should probably terminate.
- Consider $\sigma^{(k)}$ as the associated variance of the quantity $f(g^*)$ at generation k. The algorithm terminates when:

$$k \geq N_g \text{ or } \sigma^{(k)} \leq \frac{\sigma^{\left(k_{\text{last}}\right)}}{2}$$

where k_{last} is the last generation where a lower value for $f(g^*)$ was discovered.

3. Experiments

A series of benchmark functions, found in the relevant literature is introduced here as well as the conducted experiments and some discussion on the experimental results.

3.1. Test functions

To assess the effectiveness of the proposed method in locating the overall minimum of functions, a set of well - known test functions, found in the relevant literature [63,64] was employed. The functions used here are:

- **Bent Cigar function** defined as:

$$f(x) = x_1^2 + 10^6 \sum_{i=2}^{n} x_i^2$$

with the global minimum $f(x^*) = 0$. For the conducted experiments the value $n = 10$ was used.

- **Bf1** function (Bohachevsky 1), defined as:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

with $x \in [-100, 100]^2$.

- **Bf2** function (Bohachevsky 2), defined as:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$$

with $x \in [-50, 50]^2$.

- **Branin** function, given by $f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ with $-5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15$, with $x \in [-10, 10]^2$.
- **CM** function. The Cosine Mixture function is given by

$$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$$

with $x \in [-1, 1]^n$. The value $n = 4$ was used in the conducted experiments.

- **Discus function** The function is defined as

$$f(x) = 10^6 x_1^2 + \sum_{i=2}^{n} x_i^2$$

with global minimum $f(x^*) = 0$. For the conducted experiments the value $n = 10$ was used.

- **Easom** function The function is given by the equation

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2 - \pi)^2 - (x_1 - \pi)^2\right)$$

with $x \in [-100, 100]^2$.

- **Exponential** function. The function is given by

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \leq x_i \leq 1$$

The global minimum is situated at $x^* = (0, 0, ..., 0)$ with a value -1. In our experiments, we applied this function for $n = 4, 16, 64$, and referred to the respective instances as EXP4, EXP16, EXP64, EXP100 .

- **Griewank2** function. The function is given by

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100, 100]^2$$

The global minimum is located at the $x^* = (0, 0, ..., 0)$ with value 0.

- **Gkls** function. $f(x) = \text{Gkls}(x, n, w)$, is a function with w local minima, described in [65] with $x \in [-1, 1]^n$ and n a positive integer between 2 and 100. The value of the global minimum is -1 and in our experiments we have used $n = 2, 3$ and $w = 50,\ 100$.
- **Hansen** function. $f(x) = \sum_{i=1}^{5} i \cos[(i-1)x_1 + i] \sum_{j=1}^{5} j \cos[(j+1)x_2 + j], x \in [-10, 10]^2$. The global minimum of the function is -176.541793.
- **Hartman 3** function. The function is given by

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$$

with $x \in [0, 1]^3$ and $a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

The value of global minimum is -3.862782.

- **Hartman 6** function.

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$$

with $x \in [0, 1]^6$ and $a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

the value of global minimum is -3.322368.

- **High Conditioned Elliptic** function, defined as

$$f(x) = \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} x_i^2$$

Featuring a global minimum at $f(x^*) = 0$, the experiments were conducted using the value $n = 10$

- **Potential** function. As a test case, the molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard-Jones potential [66] is utilized. The function to be minimized is defined as follows:
-

$$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{3}$$

In the current experiments two different cases were studied: $N = 3,\ 5$

- **Rastrigin** function. The function is given by

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2$$

- **Shekel 7** function.

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$.

- **Shekel 5** function.

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$.

- **Shekel 10** function.

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$.

- **Sinusoidal** function. The function is given by

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \le x_i \le \pi.$$

The global minimum is situated at $x^* = (2.09435, 2.09435, ..., 2.09435)$ with a value $f(x^*) = -3.5$. In the performed experiments, we examined scenarios with $n = 4, 8$ and $z = \frac{\pi}{6}$. The parameter z is employed to offset the position of the global minimum [67].

- **Test2N** function. This function is given by the equation

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5,5].$$

The function has 2^n in the specified range and in our experiments we used $n = 4, 5, 6, 7, 8, 9$.

- **Test30N** function. This function is given by

$$f(x) = \frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i-1)^2\left(1+\sin^2(3\pi x_{i+1})\right)\right)+(x_n-1)^2\left(1+\sin^2(2\pi x_n)\right)$$

with $x \in [-10, 10]$. The function has 30^n local minima in the specified range and we used $n = 3, 4$ in the conducted experiments.

3.2. Experimental results

To evaluate the performance of the parallel genetic algorithm, a series of experiments were carried out. These experiments varied the number of parallel computing units from 1 to 10. The parallelization was achieved using the freely available OpenMP library [51], and the method was implemented in ANSI C++ within the OPTIMUS optimization package, accessible at https://github.com/itsoulos/OPTIMUS. All experiments were conducted on a system equipped with an AMD Ryzen 5950X processor, 128GB of RAM, and running the Debian Linux operating system.

To ensure the reliability and validity of the research, experiments were conducted 30 times and concerned the Tables 2 , 3 and 4, with parameters consistent across all experiments as outlined in Table 1. In Table 2, the number of objective function invocations for each problem and its solving time for various combinations of processing units (PU) and chromosomes are provided. In the columns containing objective function invocation values, values in parentheses represent the percentage of executions where the overall optimum was successfully identified. The absence of this fraction indicates a 100% success rate, meaning that the global minimum was found in every run. Generally, across all problems, there is a decrease in the number of objective function invocations and execution time as the number of parallel computing units increases. In each case, the number of chromosomes remains constant, i.e., 1PUx500chrom, 2PUx250chrom, etc. This is a positive result, indicating that parallelization improves the performance of the genetic algorithm. Figures 3 and 4 are derived from Table 2. Statistical comparison of objective function invocations, solving times, and execution times similarly shows performance improvement and computation time reduction for problems as the number of computing units increases.

Specifically, in Figure 3, the objective function invocations are halved compared to the initial invocations with only two computational units. This reduction in invocations continues significantly as the number of computational units increases. In Figure 4, we observe a similar behavior in the algorithm termination times. In this case, the times are significantly shorter in the parallel process with ten (10) computational units compared to a single computational unit. In the comparisons presented above, there is a reduction in required computational power, as shown in Figure 3, along with a decrease in the time required to find solutions, as depicted in Figure 4. In Table 2, additional interesting details regarding objective function invocations and computational times are presented, such as minimum, maximum, mean, and standard deviation. In conclusion, as the workload is distributed among an increasing number of computational units, there is an improvement in performance. This reinforces the overall methodology.

In Table 3 the chromosome migration with the best functional values occurs in every generation, involving a specific number of chromosomes ten, N P =10 participating in the propagation process. To achieve a more effective implementation of propagation techniques, we proceeded to increase the local optimization rate applied to Table 3 from 0.1% (as presented in Table 2) to 0.5% LSR . However, the procedure of local optimization was maintained at certain levels because an excessive increase would result in an elevated number of calls to the objective function. Conversely, reducing LSR would lead to a decrease in the success rate concerning the identification of optimal chromosomes. In the statistical representation of Figure 5, we observe the superiority of the '1 to N' propagation,

meaning the transfer of ten chromosomes from a random island to all others. Equally, effective appears to be the 'N to N' propagation. As a general rule, if we classify migration methods based on their performance, they will be ranked as follows: '1toN' 2b, 'NtoN' 2d, '1to1'2a , and 'Nto1'2c . The first two strategies, where migration occurs across all islands, demonstrate better performance compared to the other two, where migration only affects one island. The success of '1toN' 2b and 'NtoN' 2d , albeit with a slight difference, appears to be due to the migration of the best chromosomes to all islands. This leads to an improvement in the convergence of the algorithm towards better candidate solutions in a shorter time frame. The actual times are shown in Figure 6. During the conducted experiments, the "1-to-N" and "N-to-N" propagation techniques appear to perform better according to experimental evidence. The common feature of these two techniques is that the optimal solutions are distributed to all computing units, thereby improving the performance of each individual unit and consequently enhancing the overall performance of the general algorithm.

For conducting experiments among stochastic methods of global optimization, including Particle Swarm Optimization (PSO), Improved PSO (IPSO)[75], Differential Evolution with random selection (DE), Differential Evolution with tournament selection (TDE)[76], Genetic Algorithm (GA), and Parallel Genetic Algorithm (PGA), certain parameters remained constant. Also, the parallel implementation of the GALIB library [78] was used in the comparative experiments. The population size for all methods is 500 particles or agents or chromosomes. In PGA, the population consists of 20PUx25chrom, while all other parameters remain the same as those described in Table 2. Any method employing LSR maintains this parameter at the same value. The double box is a termination rule that is the same for all methods.

The values resulting from experiments in the 4 table are depicted in 7 and fig:8 Figures. The box plots of Figure 7 reveal the superiority of PGA, as objective function calls remain at approximately 10,000 across all problems. Conversely, IPSO, DE, and TDE (especially DE) exhibit a low number of calls in some problems, while in others, they display significant increases. During initialization and optimization, each method has a specific lower limit of calls, which varies from method to method. PGA easily reaches this threshold with very small deviations, as illustrated in the same figure. The Figure 8 presents the total call values for each method. The current work was also compared against the parallel version of Galib, found in the recent literature. Even though Galib seems to have the same success rate in discovering the global minimum of the benchmark functions, it requires significantly more functions calls than the proposed method for the same setup parameters.

Table 1. The following settings were initially used to conduct the experiments

Parameter	value	Explanation
N_c	500x1, 250x2, 100x5, 50x10	Chromosomes
N_g	200	Max generations
N_I	1, 2, 5, 10	Processing units or islands
N_R	no propagation in 2, 1: in every generation in 3	Rate of propagation
N_P	0 in 2, 10 : in 3	Chromosomes for migration
PT	no in Table 2, 1to1 2a, 1toN 2b, Nto1 2c, NtoN 2d	Propagation technique
p_s	10%	Selection rate
p_m	5%	Mutation rate
LSR	0.1% in Table 2, 0.5% in Table 3	Local search rate

Table 2: Statistical analysis comparing execution times (seconds) and function calls across varying numbers of processor units

Problems	$N_i = 1$ $N_c =$ 500 **Calls**	$N_i = 1$ $N_c =$ 500 **Time**	$N_i = 2$ $N_c =$ 250 **Calls**	$N_i = 2$ $N_c =$ 250 **Time**	$N_i = 5$ $N_c =$ 100 **Calls**	$N_i = 5$ $N_c =$ 100 **Time**	$N_i = 10$ $N_c =$ 50 **Calls**	$N_i = 10$ $N_c =$ 50 **Time**
BF1	10578	0.557	10555	0.193	10533	0.126	10511	0.121
BF2	10568	0.554	10545	0.192	10523	0.127	10533	0.119
BRANIN	46793	2.308	31231	0.562	11125	0.134	10533	0.169
CAMEL	26537	1.338	15875	0.29	15833	0.188	10861	0.123
CIGAR10	10502	1.089	10577	0.383	10583	0.222	10541	0.206
CM4	10614	1.054	10583	0.249	10581	0.151	10556	0.139
DISCUS10	10548	1.09	10532	0.382	10500	0.222	10502	0.205
EASOM	100762	4.504	100610	1.66	94541	1.089	22845	0.248
ELP10	10601	1.15	10590	0.436	10574	0.26	10557	0.242
EXP4	16621	1.092	10587	0.249	10560	0.15	10544	0.143
EXP16	10680	1.336	10654	0.53	10643	0.287	10626	0.258
EXP64	10857	2.333	10829	1.235	10814	0.825	10830	0.728
EXP100	10855	3.517	10901	1.763	10868	1.25	10887	1.052
GKLS250	50804	2.825	25832	0.607	11711	0.194	10870(93)	0.198
GKLS350	40707	2.327	23720	0.522	17646	0.26	14130	0.202
GRIEWANK2	10555	0.565	10532	0.197	10517	0.126	10492	0.118
GRIEWANK10	10679	1.079	10629	0.407	10613	0.239	10609	0.22
POTENTIAL3	39607	2.057	34327	0.881	18313	0.34	15471	0.279
PONTENTIAL5	33542	1.653	33737	1.074	12040	0.34	11082	0.291
PONTENTIAL6	28901(3)	1.56	26419(16)	1.018	14265(3)	0.478	11109(10)	0.356
PONTENTIAL10	42644(13)	3.316	37897(23)	2.538	14080(10)	0.937	11319(6)	0.66
HANSEN	46894(90)	2.494	28191(80)	0.575	11085(56)	0.153	11065	0.158
HARTMAN3	22235	1.525	19030	0.379	16463	0.212	12048	0.146
HARTMAN6	18352	1.505	15902	0.429	16726	0.279	12243	0.196
RASTRIGIN	16567	0.855	10543	0.193	10521	0.125	10506	0.116
ROSENBROCK8	10863	0.916	10700	0.333	10698	0.199	10772	0.196
POSENBROCK1	10918	1.371	10946	0.516	10867	0.304	10886	0.271
SHEKEL5	32319(50)	2.069	17913(50)	0.412	11185(36)	0.159	11010(40)	0.15
SHEKEL7	51183(73)	3.277	14981(53)	0.342	11457(60)	0.163	11035(50)	0.154
SHEKEL10	47337(70)	2.977	46927(76)	1.113	16310(56)	0.23	11329(70)	0.152
SINU4	66625(83)	4.344	31511(86)	0.77	13979(73)	0.211	11004(43)	0.161
SINU8	29705	2.57	27613	0.987	24592	0.549	11422	0.236
TEST2N4	25553	1.558	17701	0.397	24763	0.359	13217	0.178
TEST2N5	20297	1.327	18440	0.457	16759	0.265	11483	0.168
TEST2N6	20450	1.311	20837	0.566	18123	0.315	11988	0.194
TEST2N7	26113	1.924	23940	0.723	20825	0.384	11339	0.196

Continued on next page

Table 2: Statistical analysis comparing execution times (seconds) and function calls across varying numbers of processor units (Continued)

Problems	$N_i = 1\ N_c =$ 500 Calls	$N_i = 1\ N_c =$ 500 Time	$N_i = 2\ N_c =$ 250 Calls	$N_i = 2\ N_c =$ 250 Time	$N_i = 5\ N_c =$ 100 Calls	$N_i = 5\ N_c =$ 100 Time	$N_i = 10\ N_c =$ 50 Calls	$N_i = 10\ N_c =$ 50 Time
TEST2N8	18846	1.454	18549	0.585	16700	0.329	11658	0.218
TEST2N9	18154	1.582	18803	0.649	17100	0.368	13299	0.262
TEST30N3	49235	2.46	24129	0.458	14743	0.188	12345	0.152
TEST30N4	29667	1.553	17501	0.358	13367	0.186	11778	0.151
SUM	1105268	74.376	851319	25.61	633126	12.923	465835	9.532
MINIMUM	10502	0.554	10532	0.192	10500	0.125	10492	0.116
MAXIMUM	100762	4.504	100610	2.538	94541	1.25	22845	1.052
AVERAGE	27631.7	1.859	21282.975	0.640	15828.15	0.323	11645.875	0.238
STDEV	19305.784	0.972	15829.020	0.482	13335.509	0.260	2109.230	0.180

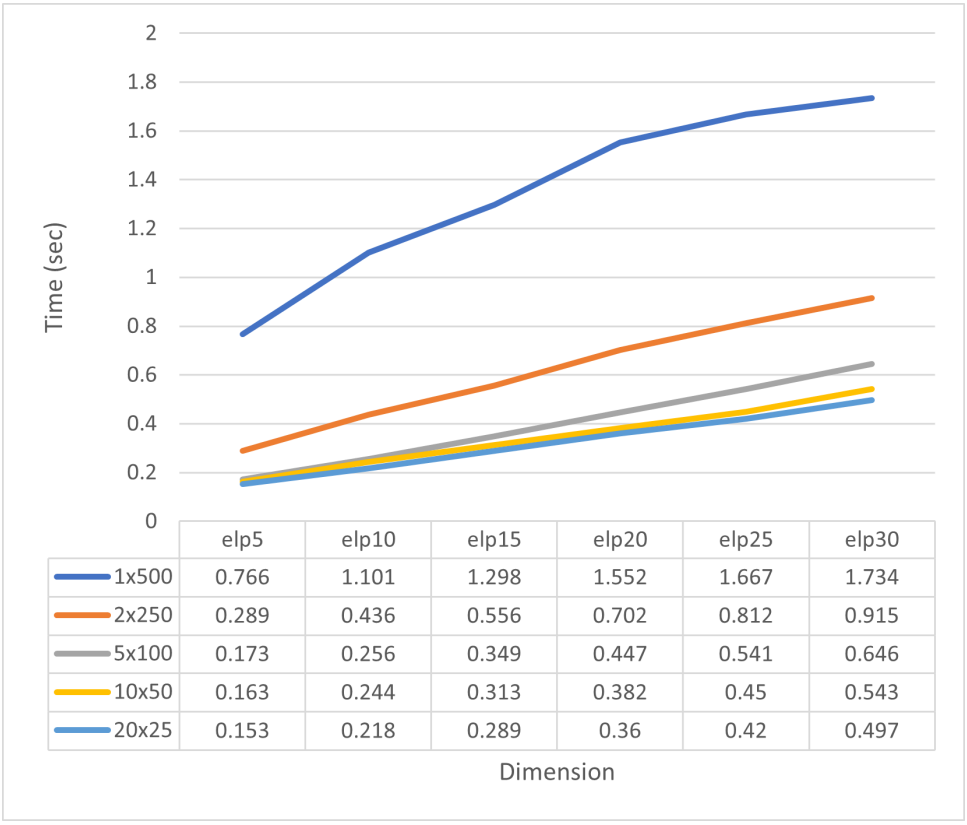
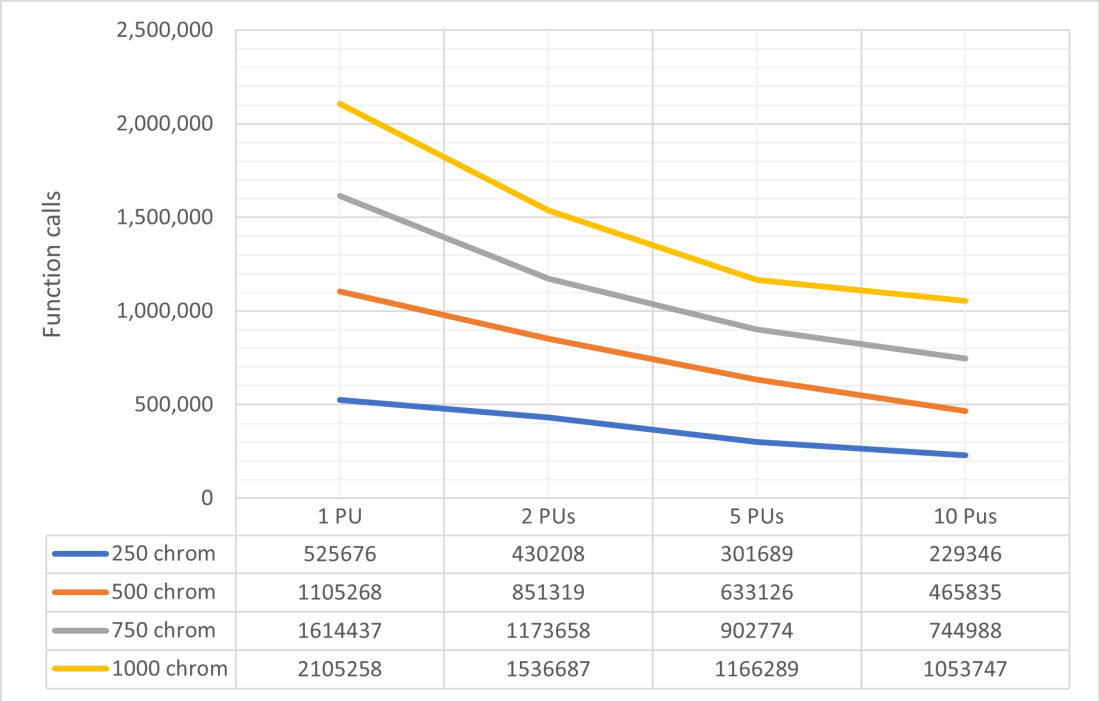


Figure 3. Statistical comparison of function calls with different number of processor units.



	1 PU	2 PUs	5 PUs	10 Pus
250 chrom	525676	430208	301689	229346
500 chrom	1105268	851319	633126	465835
750 chrom	1614437	1173658	902774	744988
1000 chrom	2105258	1536687	1166289	1053747

Figure 4. Statistical comparison of times with different number of processor units.

Table 3. Evaluating function calls and time (seconds) using various propagation techniques for comparison.

Problems	no propagation Calls	no propagation Time	1to1 Calls	1to1 Time	1toN Calls	1toN Time	Nto1 Calls	Nto1 Time	NtoN Calls	NtoN Time
BF1	10809	0.123	10741	0.127	10770	0.126	10746	0.127	10808	0.136
BF2	10725	0.124	10773	0.126	10764	0.13	10783	0.126	10731	0.136
BRANIN	48364	0.56	31470	0.397	18776	0.251	35367	0.448	19224	0.284
CAMEL	29087	0.337	18597	0.23	14429	0.185	24977	0.313	19341	0.286
CIGAR10	10854	0.233	10880	0.216	10915	0.222	10890	0.22	10869	0.235
CM4	10911	0.147	10923	0.15	10941	0.15	10918	0.15	10915	0.163
DISCUS10	10651	0.222	10632	0.213	10651	0.217	10641	0.22	10606	0.231
EASOM	99569	1.094	100163	1.106	100160	1.121	100155	1.139	98336	1.156
ELP10	10832	0.276	10902	0.261	10829	0.266	10811	0.26	10952	0.278
EXP4	10803	0.151	12037	0.167	12695	0.183	11416	0.164	10819	0.158
EXP16	11228	0.272	11259	0.276	11262	0.285	11253	0.28	11260	0.294
EXP64	12127	0.837	12204	0.848	12184	0.85	12151	0.849	12199	0.877
EXP100	12396	1.397	12376	1.4	12372	1.36	12460	1.387	12414	1.42
GKLS250	48672	0.813	55586	0.949	31493	0.564	58638	1.007	27840	0.532
GKLS350	55231	0.815	42100	0.636	28609	0.459	46923	0.72	25341	0.428
GRIEWANK2	10682	0.127	10670	0.125	10697	0.126	10683	0.127	10684	0.134
GRIEWANK10	11144	0.239	11102	0.232	11123	0.239	11171	0.229	11153	0.254
POTENTIAL3	45748	0.832	33598	0.643	17276	0.347	32603	0.631	16870	0.358
PONTENTIAL5	41946	1.156	41112	1.179	19912	0.597	37687	1.089	19622	0.614
PONTENTIAL6	46507	1.639	40518	1.449	21941	0.817	36138	1.315	21528	0.844
PONTENTIAL10	47031	3.4	45166	3.361	40212	3.239	42057	3.183	34750	2.883
HANSEN	63130	0.85	65414	0.918	39649	0.595	67369	0.947	31149	0.507
HARTMAN3	19170	0.248	20339	0.274	16280	0.226	20001	0.265	14587	0.219
HARTMAN6	23725	0.423	16856	0.285	14141	0.233	16955	0.288	13964	0.239
RASTRIGIN	11264	0.147	11256	0.132	10652	0.126	10668	0.128	11290	0.145
ROSENBROCK8	11727	0.204	11892	0.2	11681	0.203	11708	0.199	11882	0.217
POSENBROCK16	12372	0.42	12187	0.304	12394	0.313	12438	0.324	12455	0.324
SHEKEL5	44893	0.645	54184	0.751	34937	0.491	53277	0.755	40859	0.621
SHEKEL7	45722	0.638	55109	0.778	33440	0.472	49029	0.702	46066	0.696
SHEKEL10	58361	0.854	49400	0.721	32691	0.471	52798	0.783	38305	0.608
SINU4	64584	0.972	59414	0.922	36052	0.591	62924	0.972	52937	0.857
SINU8	32572	0.793	25552	0.63	19461	0.462	28744	0.716	18173	0.445
TEST2N4	23430	0.339	20474	0.3	17001	0.261	21468	0.316	18436	0.294
TEST2N5	22662	0.358	20614	0.33	16171	0.262	19697	0.316	16421	0.282
TEST2N6	21663	0.365	18721	0.323	16600	0.289	19556	0.339	14633	0.299
TEST2N7	24401	0.456	18990	0.354	15792	0.3	20967	0.405	13995	0.28
TEST2N8	21017	0.418	18532	0.369	16644	0.339	20139	0.413	13980	0.298
TEST2N9	22684	0.488	18538	0.407	16302	0.353	18929	0.421	14620	0.344
TEST30N3	24524	0.318	22799	0.296	20436	0.297	23186	0.311	19968	0.316
TEST30N4	21090	0.28	25160	0.358	21216	0.319	19444	0.276	16711	0.267
Total	**1164308**	**24.01**	**1088240**	**22.74**	**829551**	**18.33**	**1097765**	**22.86**	**836693**	**18.95**

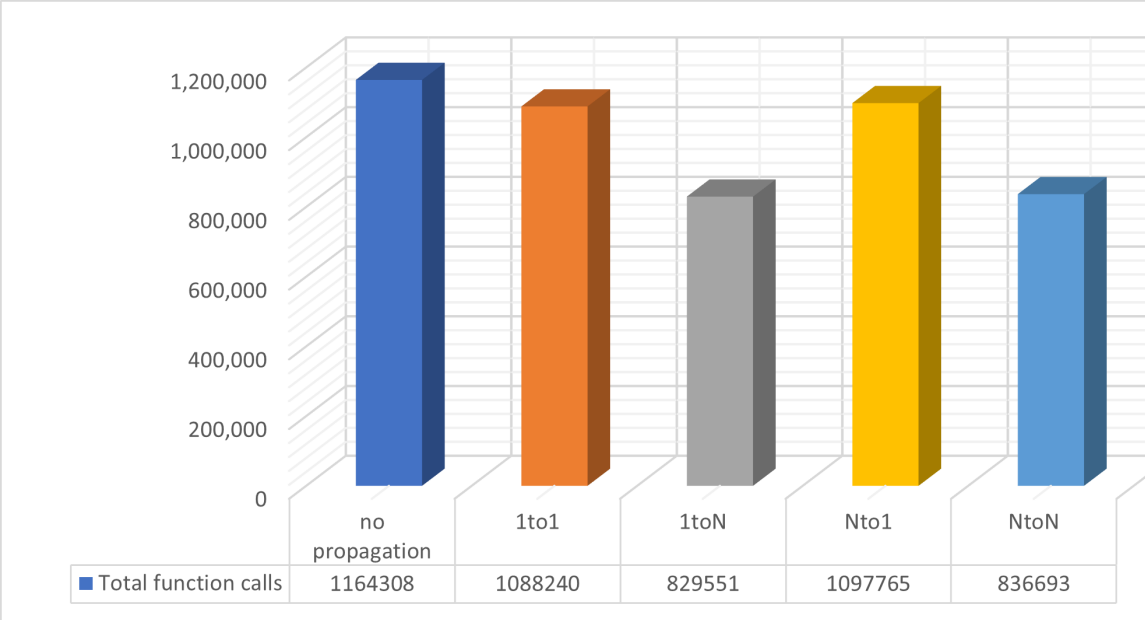


Figure 5. Statistical Comparison of function calls with different number of processor units and different propagation techniques

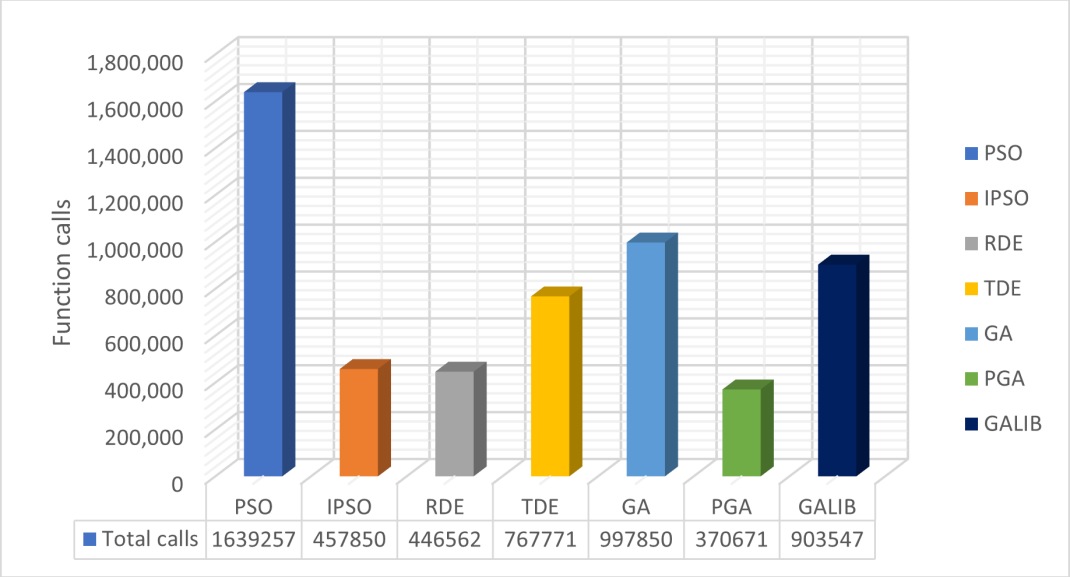


Figure 6. Comparison of times (seconds) with different number of processor units and different propagation techniques

Table 4. Comparison of function calls using different stochastic optimization methods

PROBLEMS	PSO	IPSO	RDE	TDE	GA	GALIB	PGA
BF1	50398	11478	7943(86)	5535	10578	11641	10501
BF2	50397	11292	8472(76)	5539	10568	11321	10510
BRANIN	44800	10849	5513	5514	46793	34487	10838
CAMEL	48242	11051	5555	5514	26537	17321	11087
CIGAR10	50581	12331	5586	100573	10502	11567(50)	10566
CM4	48559	11767	5550	5538	10614	11118(70)	10548
DISCUS10	50523	14328	18187	100518	10548	10988	10503
EASOM	21786	10938	29256	24691	100762	79689	10797
ELP10	49837	4323	11933	100584	10601	11673	10559
EXP4	48523	11041	46752	19467	16621	16045	10503
EXP16	50518	10973	5537	69494	10680	10500	10595
GKLS250	43925	10869	41016	11430	50804	31298	10893(76)
GKLS350	48202	10750	56220	16831	40707	29897(96)	11555(96)
GRIEWANK2	44021	13514	5538	5533	10555	14419(67)	10498
GRIEWANK10	50557(3)	12258(86)	5612(13)	85742(3)	10679	10800	10576
POTENTIAL3	49213	12124	5530	5523	39607	33452	11039
PONTENTIAL5	50548	16027	5587	5569	33542	31285	11134
PONTENTIAL6	50558(3)	24414(66)	5607(6)	5588(3)	28901(3)	28444(10)	11143(10)
PONTENTIAL10	50641(6)	31434	5670(3)	5661(6)	42644(13)	38883(20)	11290(20)
HANSEN	47296	13131	5522	5521	46894(90)	45440	11055
HARTMAN3	47778	10961	5525	5522	22235	19434	11097
HARTMAN6	50088(33)	11085(86)	5536(83)	5536	18352	18444(60)	11273
RASTRIGIN	47433	11594	5542	5524	16567	16286(96)	10506
ROSENBROCK8	50549	13487	72088	100503	10863	11419	10645
POSENBROCK16	50584	12659	21517	10645	10918	11681	10957
SHEKEL5	49944(33)	13058(93)	5532(86)	5524(93)	32319(50)	29287	10883(43)
SHEKEL7	50062(53)	12134(96)	5533(96)	5523	51183(73)	47245(77)	10926(53)
SHEKEL10	50124(63)	14176	5535(90)	5523	47337(70)	45911(77)	11207(80)
SINU4	49239	11349	5527	5510	66625(83)	66383	11063(76)
SINU8	50224	11295	5537(80)	5520	29705	29234	11378
TEST2N4	50112(93)	13173	5529	5519	25553	19913	11049
TEST2N9	50517(13)	17510(60)	5546(6)	5535(56)	18154	15376	11145
TEST30N3	44301	19638	5515	5511	49235	49234	11051
TEST30N4	49177	20839	5514	5511	29667	33428	11301
TOTAL	**1639257**	**457850**	**446562**	**767771**	**997850**	**903547**	**370671**

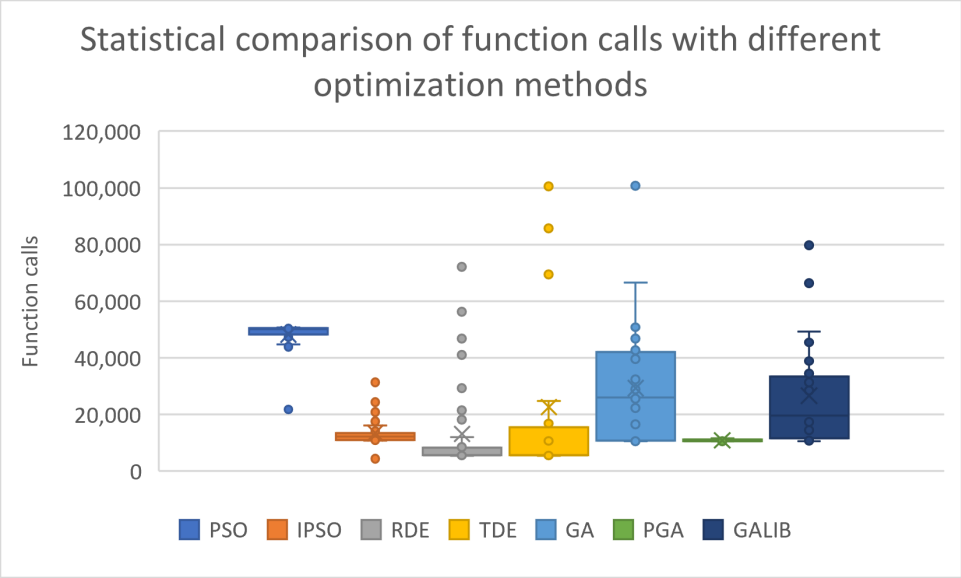


Figure 7. Statistical comparison of function calls using different stochastic optimization methods

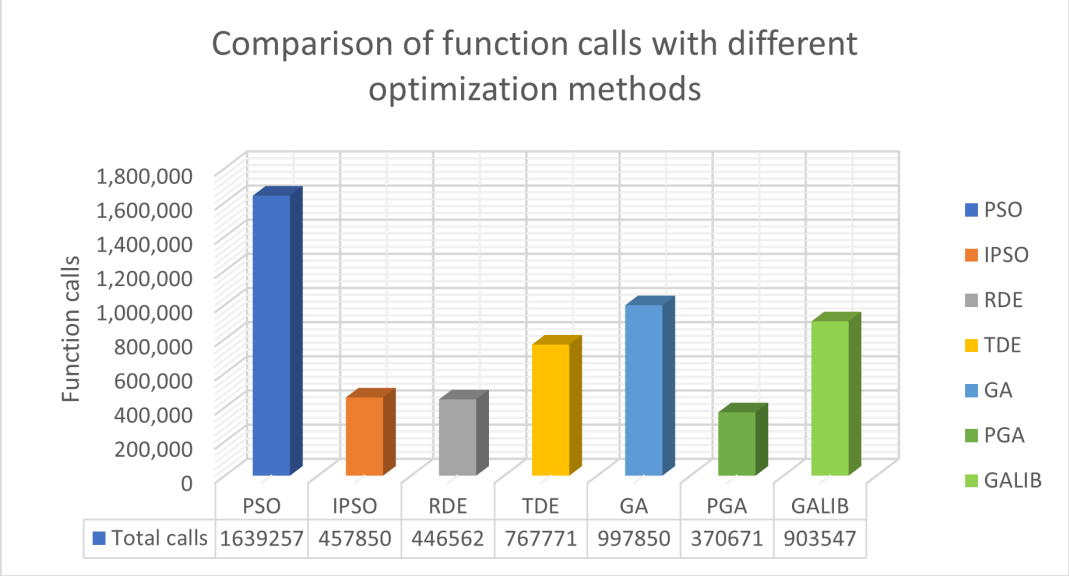
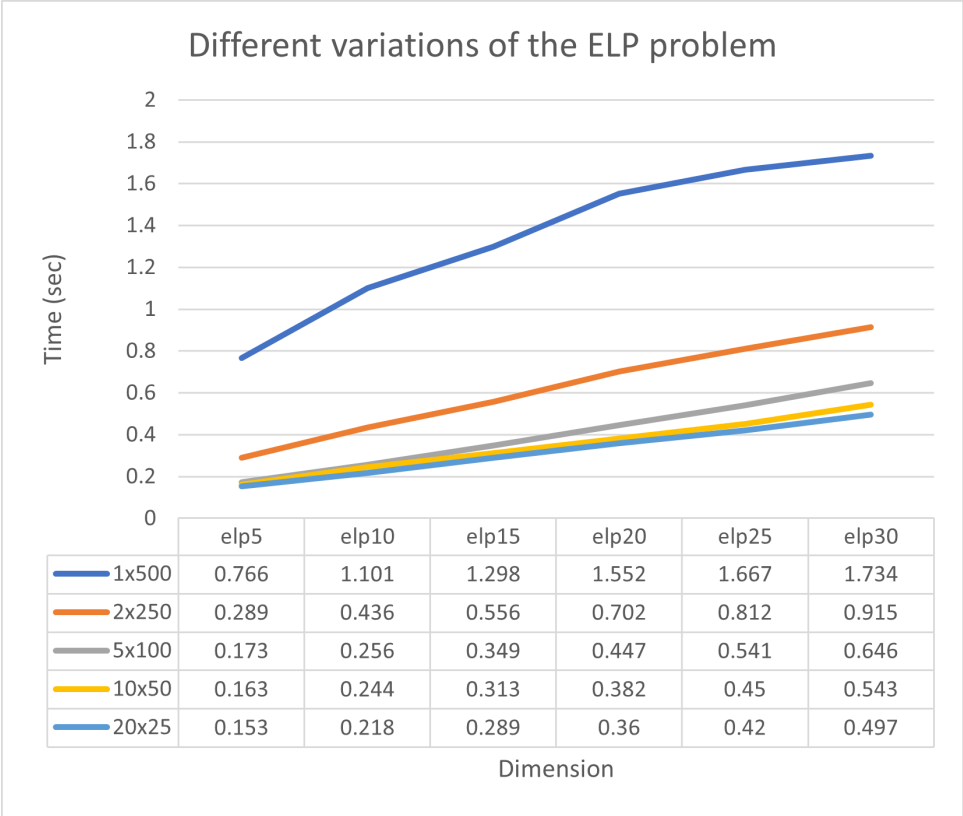


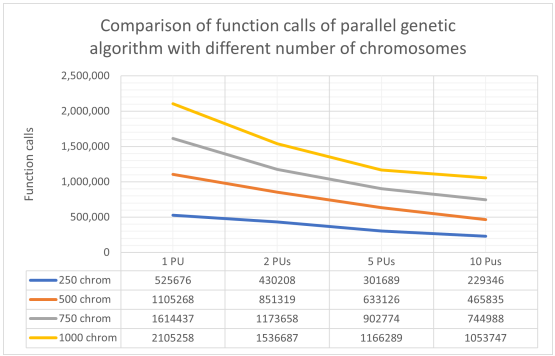
Figure 8. Comparison of total function calls using different stochastic optimization methods



	elp5	elp10	elp15	elp20	elp25	elp30
1x500	0.766	1.101	1.298	1.552	1.667	1.734
2x250	0.289	0.436	0.556	0.702	0.812	0.915
5x100	0.173	0.256	0.349	0.447	0.541	0.646
10x50	0.163	0.244	0.313	0.382	0.45	0.543
20x25	0.153	0.218	0.289	0.36	0.42	0.497

Figure 9. Different variations of the ELP problem

In Figure 9, it is observed that with the collaboration of sublisting units, the process of finding minima is significantly accelerated. Additionally, a new experiment was conducted where the number of chromosomes was varied from 250 to 1000 and the number of processing units was changed from 1 to 10. The total number of function calls for each case is shown graphically in Figure 10. The method maintains the same behavior for any number of chromosomes. This means that the set of required calls is significantly reduced by adding new parallel processing units. Of course, as expected, the total number of calls required increases as the number of available chromosomes increases.



	1 PU	2 PUs	5 PUs	10 Pus
250 chrom	525676	430208	301689	229346
500 chrom	1105268	851319	633126	465835
750 chrom	1614437	1173658	902774	744988
1000 chrom	2105258	1536687	1166289	1053747

Figure 10. Comparison of function calls with different number of chromosomes.

4. Conclusions

According to the relevant literature, despite the high success rate they exhibit in finding good functional values, genetic algorithms require significant computational power, leading to longer processing times. This manuscript introduces a parallel technique for global optimization, where a genetic algorithm is employed to solve the problem. Specifically, the initial population of chromosomes is divided into various subpopulations that run

on different computational units. During the optimization process, the islands operate independently but periodically exchange chromosomes with good functional values. The number of chromosomes participating in migration is limited by the crossover and mutation rates. Additionally, periodic local optimization is performed on each computational unit, which, in turn, should not require excessive computational power (function calls).

Experimental results revealed that even parallelization with just two computational units significantly reduces both the number of function calls and processing time, proving to be quite effective even with more computational units. Furthermore, it was observed that the most effective information exchange technique was the so-called '1toN,' with a slight difference from the 'NtoN,' where a randomly selected subpopulation sends information to all other subpopulations. Moreover, the 'NtoN' technique, where all subpopulations send information to all other subpopulations, seems to perform equally well.

Similar dissemination techniques have been applied to other stochastic methods, such as the Differential Evolution (DE) method by Charilogis and Tsoulos [73] and the Particle Swarm Optimization (PSO) method by Charilogis and Tsoulos [74]. In the case of Differential Evolution, the proposed dissemination technique is '1to1' 2a and not '1toN' 2b as suggested in this study. However, in the case of PSO and GA, the recommended dissemination technique is the same.

The parallelization of various methodologies of genetic algorithms or even different stochastic techniques for global optimization can be explored with the aim of improving the methodology. However, in such heterogeneous environments, more efficient termination criteria are required, or even their combined use.

Author Contributions: I.G. Tsoulos conceptualized the idea and methodology, supervised the technical aspects related to the software, and contributed to manuscript preparation. V. Charilogis conducted the experiments using various datasets, performed statistical analysis, and collaborated with all authors in manuscript preparation. All authors have reviewed and endorsed the conclusive version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research has been financed by the European Union : Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan , under the call RESEARCH – CREATE – INNOVATE, project name "iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195).

Conflicts of Interest: The authors have no conflicts of interest to declare.

References

1. A. Törn, A. Žilinskas, Global optimization (Vol. 350, pp. 1-255). Berlin: Springer-Verlag, 1989.
2. D. Fouskakis, D. Draper, Stochastic optimization: a review. International Statistical Review **70**, pp. 315-349, 2002.
3. Y. Cherruault, Global optimization in biology and medicine, Mathematical and Computer Modelling **20**, pp. 119-132, 1994.
4. Eva K. Lee, Large-Scale Optimization-Based Classification Models in Medicine and Biology, Annals of Biomedical Engineering **35**, pp 1095-1109, 2007.
5. A. Liwo, J. Lee, D.R. Ripoll, J. Pillardy, H. A. Scheraga, Protein structure prediction by global optimization of a potential energy function, Biophysics **96**, pp. 5482-5485, 1999.
6. W.H. Shin, J.K. Kim, D.S. Kim, C. Seok, GalaxyDock2: Protein–ligand docking using beta-complex and global optimization, J. Comput. Chem. **34**, pp. 2647– 2656, 2013.
7. Q. Duan, S. Sorooshian, V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, Water Resources Research **28**, pp. 1015-1031 , 1992.
8. L. Yang, D. Robin, F. Sannibale, C. Steier, W. Wan, Global optimization of an accelerator lattice using multiobjective genetic algorithms, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **609**, pp. 50-57, 2009.

9. E. Iuliano, Global optimization of benchmark aerodynamic cases using physics-based surrogate models, Aerospace Science and Technology **67**, pp.273-286, 2017.
10. C. D. Maranas, I. P. Androulakis, C. A. Floudas, A. J. Berger, J. M. Mulvey, Solving long-term financial planning problems via global optimization, Journal of Economic Dynamics and Control **21**, pp. 1405-1425, 1997.
11. Zwe-Lee Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on **18** Power Systems, pp. 1187-1195, 2003.
12. L. Liberti, S. Kucherenko, Comparison of deterministic and stochastic approaches to global optimization. International Transactions in Operational Research **12**, pp. 263-285, 2005.
13. M.A. Wolfe, Interval methods for global optimization, Applied Mathematics and Computation **75**, pp. 179-206, 1996.
14. T. Csendes and D. Ratz, Subdivision Direction Selection in Interval Methods for Global Optimization, SIAM J. Numer. Anal. **34**, pp. 922–938, 1997.
15. C.D Maranas, C.A. Floudas, A deterministic global optimization approach for molecular structure determination, J. Chem. Phys. **100**, 1247, 1994.
16. J. Barhen, V. Protopopescu, D. Reister, TRUST: A Deterministic Algorithm for Global Optimization, Science **276**, pp. 1094-1097, 1997.
17. Y. Evtushenko, M.A. Posypkin, deterministic approach to global box-constrained optimization, Optim Lett **7**, pp. 819–829, 2013.
18. M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, IEEE Computational Intelligence Magazine **1**, pp. 28-39, 2006.
19. K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research 185, pp. 1155-1173, 2008.
20. W. L. Price, Global optimization by controlled random search, Journal of Optimization Theory and Applications **40**, pp. 333-348, 1983.
21. Ivan Křivý, Josef Tvrdík, The controlled random search algorithm in optimizing regression models, Computational Statistics & Data Analysis **20**, pp. 229-234, 1995.
22. M.M. Ali, A. Törn, and S. Viitanen, A Numerical Comparison of Some Modified Controlled Random Search Algorithms, Journal of Global Optimization **11**,pp. 377–385, 1997.
23. J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.
24. Ioan Cristian Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters **85**, pp. 317-325, 2003.
25. Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.
26. S. Kirkpatrick, CD Gelatt, , MP Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.
27. L. Ingber, Very fast simulated re-annealing, Mathematical and Computer Modelling **12**, pp. 967-973, 1989.
28. R.W. Eglese, Simulated annealing: A tool for operational research, Simulated annealing: A tool for operational research **46**, pp. 271-281, 1990.
29. R. Storn, K. Price, Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization **11**, pp. 341-359, 1997.
30. J. Liu, J. Lampinen, A Fuzzy Adaptive Differential Evolution Algorithm. Soft Comput **9**, pp.448–462, 2005.
31. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
32. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
33. S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.
34. I. BoussaïD, J. Lepagnot, P. Siarry, P., A survey on optimization metaheuristics. Information sciences **237**, pp. 82-117, 2013.
35. T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms. Computers & Industrial Engineering **137**, 106040, 2019.
36. K. Hussain, M.N.M. Salleh, S. Cheng, Y. Shi, Metaheuristic research: a comprehensive survey.Artificial Intelligence Review **52**, pp. 2191-2233, 2019.
37. Holland, J.H. Genetic algorithms. Sci. Am. 1992, 267, 66–73.
38. Stender, J. Parallel Genetic Algorithms: Theory & Applications; IOS Press: Amsterdam, The Netherlands, 1993.
39. Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning; Addison-Wesley Publishing Company: Reading, MA, USA, 1989.
40. Michaelewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs; Springer: Berlin/Heidelberg, Germany, 1996.
41. :Santana, Y.H.; Alonso, R.M.; Nieto, G.G.; Martens, L.; Joseph, W.; Plets, D. Indoor genetic algorithm-based 5G network planning using a machine learning model for path loss estimation. Appl. Sci. 2022, 12, 3923.
42. : Liu, X.; Jiang, D.; Tao, B.; Jiang, G.; Sun, Y.; Kong, J.; Chen, B. Genetic algorithm-based trajectory optimization for digital twin robots. Front. Bioeng. Biotechnol. 2022, 9, 793782. [7]:
43. Nonoyama, K.; Liu, Z.; Fujiwara, T.; Alam, M.M.; Nishi, T. Energy-efficient robot configuration and motion planning using genetic algorithm and particle swarm optimization. Energies 2022, 15, 2074.

44. Liu, K.; Deng, B.; Shen, Q.; Yang, J.; Li, Y. Optimization based on genetic algorithms on energy conservation potential of a high speed SI engine fueled with butanol–gasoline blends. Energy Rep. 2022, 8, 69–80.
45. Zhou, G.; Zhu, Z.; Luo, S. Location optimization of electric vehicle charging stations: Based on cost model and genetic algorithm. Energy 2022, 247, 123437
46. Min, D.; Song, Z.; Chen, H.; Wang, T.; Zhang, T. Genetic algorithm optimized neural network based fuel cell hybrid electric vehicle energy management strategy under start-stop condition. Appl. Energy 2022, 306, 118036
47. Doewes, R.I.; Nair, R.; Sharma, T. Diagnosis of COVID-19 through blood sample using ensemble genetic algorithms and machine learning classifier. World J. Eng. 2022, 19, 175–182.
48. Choudhury, S.; Rana, M.; Chakraborty, A.; Majumder, S.; Roy, S.; RoyChowdhury, A.; Datta, S. Design of patient specific basal dental implant using Finite Element method and Artificial Neural Network technique. J. Eng. Med. 2022, 236, 1375–1387.
49. Chen, Q.; Hu, X. Design of intelligent control system for agricultural greenhouses based on adaptive improved genetic algorithm for multi-energy supply system. Energy Rep. 2022, 8, 12126–12138.
50. R.L. Graham, T.S. Woodall, J.M. Squyres, Open MPI: A flexible high performance MPI. In Parallel Processing and Applied Mathematics: 6th International Conference, PPAM 2005, Poznań, Poland, September 11-14, 2005, Revised Selected Papers 6 (pp. 228-239). Springer Berlin Heidelberg, 2006.
51. E. Ayguadé, N. Copty, A. Duran, J. Hoeflinger, Y. Lin, F. Massaioli, G. Zhang, The design of OpenMP tasks. IEEE Transactions on Parallel and Distributed systems **20**, pp. 404-418, 2008.
52. E. Onbaşoğlu, L. Özdamar, Parallel simulated annealing algorithms in global optimization. Journal of global optimization **19**, pp. 27-50, 2001.
53. J.F. Schutte, J.A. Reinbolt, B.J. Fregly, R.T. Haftka, A.D. George, Parallel global optimization with the particle swarm algorithm. International journal for numerical methods in engineering **61**, pp. 2296-2315, 2004.
54. Regis, R. G., & Shoemaker, C. A. (2009). Parallel stochastic global optimization using radial basis functions. INFORMS Journal on Computing, 21(3), 411-426.
55. Tomohiro Harada and Enrique Alba. 2020. Parallel Genetic Algorithms: A Useful Survey. ACM Comput. Surv. 53, 4, Article 86 (August 2020), 39 pages. https://doi.org/10.1145/3400031.
56. L.A. Anbarasu, P. Narayanasamy, V. Sundararajan, Multiple molecular sequence alignment by island parallel genetic algorithm. Current Science, pp. 858-863, 2000.
57. U. Tosun, T. Dokeroglu, C. Cosar, A robust island parallel genetic algorithm for the quadratic assignment problem. International Journal of Production Research **51**, pp. 4117-4133, 2013.
58. A. Nandy, D. Chakraborty, M.S. Shah, Optimal sensors/actuators placement in smart structure using island model parallel genetic algorithm. International Journal of Computational Methods **16**, 1840018, 2019.
59. I.G. Tsoulos, A. Tzallas, D. Tsalikakis, PDoublePop: An implementation of parallel genetic algorithm for function optimization. Computer Physics Communications **209**, pp. 183-189, 2016.
60. R. Shonkwiler, Parallel genetic algorithms. In ICGA (pp. 199-205), 1993.
61. E. Cantú-Paz, A survey of parallel genetic algorithms, Calculateurs paralleles, reseaux et systems repartis **10**, pp. 141-171, 1998.
62. H. Mühlenbein, Parallel genetic algorithms in combinatorial optimization. In Computer science and operations research, pp. 441-453, Pergamon, 1992.
63. C.A. Floudas, P.M. Pardalos, C. Adjiman, W. Esposoto, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, C. Schweiger, Handbook of Test Problems in Local and Global Optimization, Kluwer Academic Publishers, Dordrecht, 1999.
64. M. Montaz Ali, Charoenchai Khompatraporn, Zelda B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization **31**, pp 635-672, 2005.
65. M. Gaviano, D.E. Ksasov, D. Lera, Y.D. Sergeyev, Software for generation of classes of test functions with known local and global minima for global optimization, ACM Trans. Math. Softw. **29**, pp. 469-480, 2003.
66. J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.
67. Z.B. Zabinsky, D.L. Graesser, M.E. Tuttle, G.I. Kim, Global optimization of composite laminates using improving hit and run, In: Recent advances in global optimization, pp. 343-368, 1992.
68. Yu, X., Gen, M. (2010). Introduction to Evolutionary Algorithms. Springer London Dordrecht Heidelberg New York. ISBN 978-1-84996-128-8 e-ISBN 978-1-84996-129-5 DOI 10.1007/978-1-84996-129-5.
69. Lawrence D. (1991). Handbook Of Genetic Algorithms. Publisher: Thomson Publishing Group (First Edition)
70. P. Kaelo, M.M. Ali, Integrated crossover rules in real coded genetic algorithms, European Journal of Operational Research **176**, pp. 60-76, 2007.
71. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
72. Tsoulos, I.G. Modifications of real code genetic algorithm for global optimization. Appl. Math. Comput. 2008, 203, 598–607.
73. Charilogis, V.; Tsoulos, I.G. A Parallel Implementation of the Differential Evolution Method. Analytics 2023, 2, 17–30.
74. Charilogis, V.; Tsoulos, I.G.; Tzallas, A. (2023). An Improved Parallel Particle Swarm Optimization. SN Computer Science (2023) 4:766
75. Charilogis, V., Tsoulos, I. (2022). Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions. Information 2022, 13(5), 217; Doi:10.3390/info13050217.

76. Charilogis, V., Tsoulos, I., Tzallas, A., Karvounis, E. (2022). Modifications for the Differential Evolution Algorithm. Symmetry 2022,14(3),447; Doi: 10.3390/sym14030447
77. S. Heiles, R. L. Johnston, Global optimization of clusters using electronic structure methods, Int. J. Quantum Chem. **113**, pp. 2091–2109, 2013.
78. M. Wall, GAlib: A C++ library of genetic algorithm components. Mechanical Engineering Department, Massachusetts Institute of Technology. 1996 Aug;87:54.