analytics

MDPI

Article

A Parallel Implementation of the Differential Evolution Method

Vasileios Charilogis † and Ioannis G. Tsoulos *,†

Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece
* Correspondence: itsoulos@uoi.gr
† These authors contributed equally to this work.

Abstract: Global optimization is a widely used technique that finds application in many sciences such as physics, economics, medicine, etc., and with many extensions, for example, in the area of machine learning. However, in many cases, global minimization techniques require a high computational time and, for this reason, parallel computational approaches should be used. In this paper, a new parallel global optimization technique based on the differential evolutionary method is proposed. This new technique uses a series of independent parallel computing units that periodically exchange the best solutions they have found. Additionally, a new termination rule is proposed here that exploits parallelism to accelerate process termination in a timely and valid manner. The new method is applied to a number of problems in the established literature and the results are quite promising.

Keywords: global optimization; stopping rules; parallel computing

1. Introduction

The task of locating the global minimum of a continuous and differentiable function $f : S \to R, S \subset R^n$ is defined as

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

The set S is defined as:

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

A variety of practical problems from various research fields can be modeled as global optimization problems, such as problems from physics [1–3], chemistry [4–6], economics [7,8], medicine [9,10], etc. Many methods have been proposed to tackle the problem of Equation (1), such as controlled random search methods [11–13], simulated annealing methods [14–16], differential evolution methods [17,18], particle swarm optimization (PSO) methods [19–21], ant colony optimization [22,23], genetic algorithms [24–26], etc. Reviews of stochastic methods for global optimization problems can be found in the work of Pardalos et al. [27] or in the work of Fouskakis et al. [28].

The current work proposes a parallel implementation of the differential evolution (DE) method that aims to speed up the optimization process of this particular method and also tries to make adequate use of modern computing structures with multicore architectures. The DE method initially generates a population of candidate solutions, which iteratively evolves through the crossover process in order to discover the global minimum of the objective function. The method was applied in various research fields such as electromagnetics [29], energy consumption problems [30], job shop scheduling [31], image segmentation [32], etc. The proposed method partitions the processing into independent structural units, such as threads, and each of them acts independently. Furthermore, the new method proposes a way of communication between the different building blocks of parallel processing and a process termination technique suitably modified for parallel processing.

In the recent literature, several methods have been proposed that take full advantage of parallel processing, such as parallel techniques [33–35], methods that take advantage of

check for updates

Citation: Charilogis, V.; Tsoulos, I.G. A Parallel Implementation of the Differential Evolution Method. *Analytics* **2023**, *2*, 17–30. https://doi.org/10.3390/analytics2010002

Received: 22 December 2022
Revised: 4 January 2023
Accepted: 5 January 2023
Published: 9 January 2023

GPU architectures [36–38], etc. Moreover, Weber et al. [39] proposed a parallel DE method for large-scale optimization problems using new search mechanisms for the individuals of the subpopulations. Chen et al. [40] proposed a parallel DE method for cluster optimization using modified genetic operators. Moreover, Penas et al. [41] suggested an enhanced parallel asynchronous DE algorithm for problems in computational systems biology. Recently, Sui et al. [42] proposed a parallel compact DE method applied to image segmentation.

The proposed technique is a modified version of the parallel island methodology for different evolutionary techniques [43,44]. Therefore, in the proposed technique, the initial population of agents (candidate solutions) is divided into a series of independent populations and each individual population evolves independently in a parallel computing unit, such as a thread. Populations periodically exchange information with each other, such as the lowest functional value to which they have been driven. The proposed technique uses a new differential weight calculation scheme, can use a number of different information exchange methods between the parallel computing units and furthermore proposes a new termination method of the optimization process that can take advantage of the parallelism so that the optimization terminates in time and is valid.

The rest of this article is organized as follows: in Section 2, the original DE method as well as the proposed modifications are outlined, in Section 3, the experimental test functions from the relevant literature and the associated experimental results are listed, and finally, in Section 4, some conclusions and guidelines for future research are provided.

2. Method Description

In this section, the basic DE method is first presented and then, the proposed modifications are analyzed so that it can be executed in parallel.

2.1. The Original DE Method

The original method was originally proposed by Storn [45], and it has been modified in various research papers. For example, the compact differential evolution algorithm [46,47], a self-adaptive DE [48] where the parameters of the method are modified iteratively, fuzzy logic modifications [49], etc. Moreover, a numerical study on some modifications of the DE method can be found in the work of Kaelo et al. [50]. The base DE method has the steps described below in Algorithm 1.

Algorithm 1: The steps of the base DE method

1. **INPUT:**
   - (a) The population size NP ≥ 4. The members of this population are also called agents.
   - (b) The crossover probability CR $\in [0, 1]$.
   - (c) The differential weight $F \in [0, 2]$.
2. **OUTPUT**:
   - (a) The agent x_{best} with the lower function value $f(x_{\text{best}})$.
3. **Initialize** all agents in S.
4. **While** termination criteria are not held **do**
   - (a) **For** $i = 1 \ldots \text{NP}$ **do**
     - i. **Select** as x the agent i.
     - ii. **Select** randomly three agents a, b, c with the property $a \neq b$, $b \neq c$, $c \neq a$.
     - iii. **Select** a random position $R \in \{1, \ldots, n\}$
     - iv. **Create** the vector $y = [y_1, y_2, \ldots, y_n]$ with the following procedure
     - v. **For** $j = 1, \ldots, n$ **do**
       - A. **Set** $r_i \in [0, 1]$ a random number.
       - B. **If** $r_j < \text{CR}$ **or** $j = R$ **then** $y_j = a_j + F \times (b_j - c_j)$ **else** $y_j = x_j$.
     - vi. If $y \in S$ AND $f(y) \leq f(x)$ then $x = y$.
     - vii. **EndFor**
   - (b) **EndFor**
5. **End While**

2.2. Proposed Modifications

In the proposed procedure, the population of agents is segmented into N independent contiguous segments. Each section is called an island in the following text, similar to island genetic algorithms [51,52], which are very popular parallel variants of genetic algorithms. For example, if there are 10 agents and 2 islands, then agents 1–5 are assigned to island 1 and agents 6–10 to island 2. On each island, the process of differential evolution is carried out independently. Of course, there should be some mechanism for communication between the islands as well as some appropriate mechanism for terminating the overall process. The proposed Algorithm 2 is presented next.

Algorithm 2: The steps of the proposed method

1. **INPUT:**
   (a) The parameters NP, CR, F.
   (b) The integer parameter N, which stands for the number of islands.
   (c) The integer parameter N_R, which represents the propagation rate.
   (d) The integer parameter N_I, which represents the number of islands that should terminate in order to terminate the whole process.
2. **OUTPUT**:
   (a) The agent x_{best} with the lower function value $f\left(x_{\text{best}}\right)$.
3. **Initialize** all agents in S.
4. **Set** iter = 1
5. **For** i = 1, . . . , N **do in Parallel**
   (a) **Perform** for every island i the step 4.a of the base DE algorithm of Section 2.1.
6. **EndFor**
7. **If** iter mod N_R = 0, apply the propagation scheme of Section 2.3 to the islands. The default value used in the experiments is the "1 to 1" case.
8. **Set** iter = iter + 1
9. **If** the termination rule of Section 2.4 is not valid, goto 5.
10. **Apply** local search procedure to x_{best}. The local search procedure used in the proposed method is the BFGS variant of Powell [53].

2.3. Propagation Mechanism

During the propagation mechanism, the best values of the islands are spread to the rest by replacing their worst values. In general, there are the following propagation cases:

1. **One to one**. In this case, a random island will send to another randomly selected island its best value.
2. **One to N**. In this case, a random island will send its best value to all other islands.
3. **N to one**. In this case, all islands will inform a randomly selected island about their best value.
4. **N to N**. All islands will inform all the other islands about their best value.

2.4. Termination Rule

In the proposed termination criterion, a simple criterion is checked separately on each island. For every island i the difference

$$\delta_i^{(k)} = \left| f_{i,\text{min}}^{(k)} - f_{i,\text{min}}^{(k-1)} \right|, \tag{2}$$

is measured at each iteration k, where $f_{i,\text{min}}^{(k)}$ is the best located function value for island i at iteration k. If $\delta_i^{(k)} \leq \epsilon$ for at least M consecutive iterations, then most likely the island i

should terminate the population evolution. In the proposed technique, if the quantity of Equation (2) holds for more than N_I islands, then the overall algorithm terminates.

3. Experiments

In the following, the benchmark functions used in the experiments as well as the experimental results are presented.

3.1. Test Functions

To evaluate the ability of the proposed technique to find the total minimum of functions, a series of test functions from the relevant literature [54,55] were used and are presented below.

- **Bent-cigar function.** The function is

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

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

- **Bf1** function. The function Bohachevsky 1 is given by the equation

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

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

- **Bf2** function. The function Bohachevsky 2 is given by the equation

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

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

- **Branin** function. The function is defined 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 $-5 \leq x_1 \leq 10$, $0 \leq x_2 \leq 15$. The value of the global minimum is 0.397887 with $x \in [-10, 10]^2$.

- **CM** function. The cosine mixture function is given by the equation

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

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

- **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 $x \in [-100, 100]^2$ and a global minimum of -1.0.

- **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 located at $x^* = (0, 0, \ldots, 0)$ with a value of -1. In our experiments, we used this function with $n = 4, 16, 64$ and the corresponding functions are denoted by the labels EXP4, EXP16 and EXP64.

- **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 $x^* = (0, 0, \ldots, 0)$ with a value of 0.

- **Gkls** function. $f(x) = \text{Gkls}(x, n, w)$ is a function with w local minima, described in [56] 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 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 the 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 the global minimum is -3.322368.

- **High conditioned elliptic** function, defined as

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

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

- **Potential** function. The molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard–Jones potential [57] was used as a test case here. The function to be minimized is given by:

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

In the 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

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

The global minimum is located at $x^* = (2.09435, 2.09435, \ldots, 2.09435)$ with $f(x^*) = -3.5$. For the conducted experiments, the cases of $n = 4, 8$ and $z = \frac{\pi}{6}$ were studied. The parameter z was used to shift the location of the global minimum [58].

- **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 $n = 4, 5, 6, 7$.

- **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 modified version of the differential evolutionary technique, a series of experiments were performed in which the number of parallel computing units varied from 1 to 10. The freely available OpenMP library [59] was used for parallelization, and the method was coded in ANSI C++ inside the OPTIMUS optimization package available from https://github.com/itsoulos/OPTIMUS (accessed on 4 January 2023). All the experiments were conducted on an AMD Ryzen 5950X with 128 GB of RAM and the Debian Linux operating system. All the experiments were conducted 30 times using different seed for the random generator each time and averages were reported. The values for the parameters used in the DE algorithm are shown in Table 1. The parameter F (differential weight) was calculated as:

$$F = -\frac{1}{2} + 2 \times R, \tag{4}$$

where $R \in [0, 1]$ is a random number, which was used in [60]. This random scheme for the calculation of the parameter F was used successfully to better explore the search space of the objective function. The experimental results for different numbers of threads for the test functions of the previous subsection are shown in Table 2. The number in the cells denote average function calls for every test function. The number in parentheses stands for the fraction of executions where the global optimum was successfully found. The absence of this number indicates that the global minimum was computed for every independent run (100% success). At the end of the table, an additional row named average has been added and shows the total number of function calls for all test functions and the average success rate in locating the global minimum.

As can be seen, as the number of computational threads increased, the required number of function calls needed to locate the global minimum decreased, with no appreciable difference in the overall reliability of the method, which remained extremely high (99–100%). In addition, to show the dynamics of the proposed methodology, it was also used in the training of an artificial neural network [61] for learning a common benchmark problem from machine learning, the wine problem [62,63]. The sums of execution times for 30 independent runs are displayed in Figure 1. As we can see, as the number of network

weights increased from w = 5 to w = 20, the gain from using multiple processing threads increased as the training time decreased noticeably.

Table 1. Experimental settings.

Parameter	Value
NP	200 agents
Propagation	1-to-1 method
N_R	5 iterations
N_I	2 islands
CR	0.9 for the crossover probability
M	15 iterations
ϵ	10^{-4}

Table 2. Comparison of experimental results with a "1 to 1" propagation scheme. The first column represents the name of the objective function and the remaining columns are the average function calls using 1 to 10 processing threads for the proposed method.

Function	Thread 1	Threads 4	Threads 5	Threads 10
BF1	5908	5517	5310	4887
BF2	5415	5008	4888	4577
BRANIN	5467	4767	4535	3895
CIGAR10	1886	1885	1885	1875
CM4	2518	2432	2330	2243 (0.97)
DISCUS10	1818	1816	1814	1807
EASOM	1807	1802	1801	1791
ELP10	44,910	41,731	41,930	29,944
EXP4	1820	1816	1814	1806
EXP16	1838	1835	1834	1830
EXP64	1842	1840	1839	1838
GKLS250	1987	1897	1879	1818
GKLS350	2428	2373	2299	2195
GRIEWANK2	5544	4811	4612	4208
POTENTIAL3	11,121	7868	7260	5598
POTENTIAL5	24,708	15,146	13,793	8620
HANSEN	23,035	13,602	12,178	9242
HARTMAN3	3406	3198	3162	2883
HARTMAN6	7611	6172	5739	4877 (0.97)
RASTRIGIN	5642	4537	4386	3707
ROSENBROCK4	11,859	10,441	10,139	9473
ROSENBROCK8	21,640	19,536	20,560	20,654
SHEKEL5	12,491	10,247	9754	5065 (0.80)
SHEKEL7	10,755	9183	8857	6996
SHEKEL10	10,257	9002	8705	7283
SINU4	6045	5473	5301	4434
SINU8	9764	8132	7748	4523
TEST2N4	8521	7487	7404	6834
TEST2N5	10,218	8916	8715	8050
TEST2N6	11,984	10,240	10,191	9175
TEST2N7	15,674	13,983	13,341	9760
TEST30N3	3720	3379	3349	2994
TEST30N4	3728	3382	3363	3031
Average	**298,267**	**249,454**	**242,715**	**197,913 (0.99)**

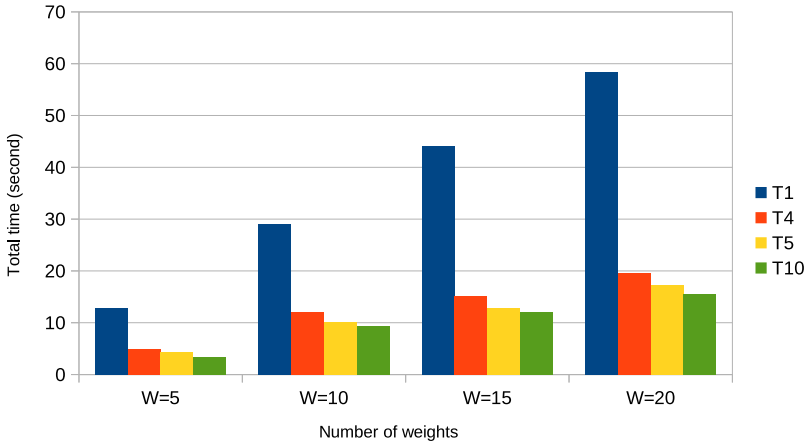

Figure 1. Time comparison when the proposed method applied to Neural Network training.

In addition, to discover whether there was differentiation using the different propagation techniques, additional experiments were performed using 10 processing threads. In each processing thread, as before, the population of each island was 20 agents. The results from these experiments are shown in Table 3. From the experimental results, it appears that most of the time, there were no significant changes in the total number of function calls except in the case of an "N to N" propagation. There was a significant reduction in function calls, but also a drop in reliability of the techniques from 99% to 91%. This may be because, due to the exchange of the best costs between all the islands, the total population was locked into local minima.

Table 3. Experiments for the proposed method using different options for the propagation method. The number of processing threads was set to 10. Numbers in cells represent average function calls for every test function.

Function	1 to 1	1 to N	N to 1	N to N
BF1	4887	4259	4209	2792
BF2	4577	4021	3917	2691
BRANIN	3895	3378	3307	2382
CIGAR10	1875	1874	1871	1873
CM4	2243 (0.97)	2173	2136 (0.97)	2030
DISCUS10	1807	1810	1808	1809
EASOM	1791	1790	1791	1789
ELP10	29,944	42,025	22,876	19,117
EXP4	1806	1807	1811	1806
EXP16	1830	1829	1828	1824
EXP64	1838	1838	1838	1836
GKLS250	1818	1812	1810	1802
GKLS350	2195	2163	2109	2011 (0.97)
GRIEWANK2	4208	3620	3514	2445 (0.80)
POTENTIAL3	5598	4445	4353	2521
POTENTIAL5	8620	7475	7025	3374
HANSEN	9242	6075	6181	3135
HARTMAN3	2883	2664	2593	2207
HARTMAN6	4877 (0.97)	4327 (0.83)	4362 (0.80)	2834 (0.57)
RASTRIGIN	3707	3213	2870	2220 (0.90)
ROSENBROCK4	9473	8294	7883	7084
ROSENBROCK8	20,654	24,470	15,919	19,272
SHEKEL5	5065 (0.80)	7556	5386 (0.93)	4456 (0.70)

Table 3. *Cont.*

Function	1 to 1	1 to N	N to 1	N to N
SHEKEL7	6996	7207 (0.90)	6488 (0.93)	4493 (0.80)
SHEKEL10	7283	6812 (0.93)	6440	3916 (0.73)
SINU4	4434	4204	4020	2796 (0.97)
SINU8	4523	5386	4605	3341 (0.90)
TEST2N4	6834	5777	5625	3609 (0.97)
TEST2N5	8050	6695 (0.97)	6647	4179 (0.73)
TEST2N6	9175	7770 (0.93)	7660	4522 (0.53)
TEST2N7	9760	9259 (0.77)	9081	5200 (0.57)
TEST30N3	2994	2814	2653	2210
TEST30N4	3031	2797	2700	2107
Average	**197,913 (0.99)**	**201,649 (0.98)**	**167,316(0.98)**	**129,683 (0.91)**

Furthermore, the proposed method was compared against the original differential evolution method and two variants from the relevant literature, mentioned as DERL and DELB [50]. The results from this comparison are shown in Table 4. As is evident, the proposed technique significantly outperformed the other modifications of the different evolutionary method. This was largely due to the different differential weight calculation technique but also to the proposed termination method. The used differential weight calculation technique largely succeeded in providing a better search of the search space, while the new termination method terminated the optimization method in time. Moreover, this new termination technique was modified to perform well in parallel computing environments as well.

Table 4. Comparison of the proposed method against other variants of the differential evolution technique.

Function	Proposed	Original DE	DERL	DELB
BF1	4887	5516	2881	5319
BF2	4577	5555	2895	5405
BRANIN	3895	5656	2857	4830
CIGAR10	1875	88,396	66,161	58,460
CM4	2243 (0.97)	9107	3856	6014
DISCUS10	1807	87,657	55,722	49,014
EASOM	1791	7879	7225	14,934
ELP10	29,944	33,371	9345	39,890
EXP4	1806	6027	2638	4142
EXP16	1830	26,194	25,117	11,740
EXP64	1838	26,497	27,831	18,346
GKLS250	1818	3800	1983	3706
GKLS350	2195	6206	2901	5027
GRIEWANK2	4208	6365	3325	6165
POTENTIAL3	5598	82,933	111,496	44,592
POTENTIAL5	8620	24,118	61,694	46,557
HANSEN	9242	18,470	7123	12212
HARTMAN3	2883	4655	2205	4124
HARTMAN6	4877 (0.97)	15,488	5343	7215 (0.93)
RASTRIGIN	3707	6362	3102	5704
ROSENBROCK4	9473	16,857	6679	10,411
ROSENBROCK8	20,654	56,445	17,198	22,939
SHEKEL5	5065 (0.80)	13,079	5224 (0.90)	8167
SHEKEL7	6996	12,409	4994 (0.97)	8093
SHEKEL10	7283	13,238	5240	8822

Table 4. *Cont.*

Function	Proposed	Original DE	DERL	DELB
SINU4	4434	8977	3828	6052
SINU8	4523	28,871	9318	10,157
TEST2N4	6834	10,764	4529	7331
TEST2N5	8050	15,568	5917	8969
TEST2N6	9175	21,185	7613	10,648
TEST2N7	9760	28,411	9492	12,252
TEST30N3	2994	4965	2758	4693
TEST30N4	3031	5123	2688	5153
Average	**197,913 (0.99)**	**706,144**	**491,178 (0.99)**	**477,083 (0.99)**

Moreover, a statistical comparison was performed for the proposed method and different numbers of processing threads, and the results are outlined in Figure 2. A statistical comparison was also included for the results of the proposed method against the other variations of the DE method and the corresponding plot is shown in Figure 3.

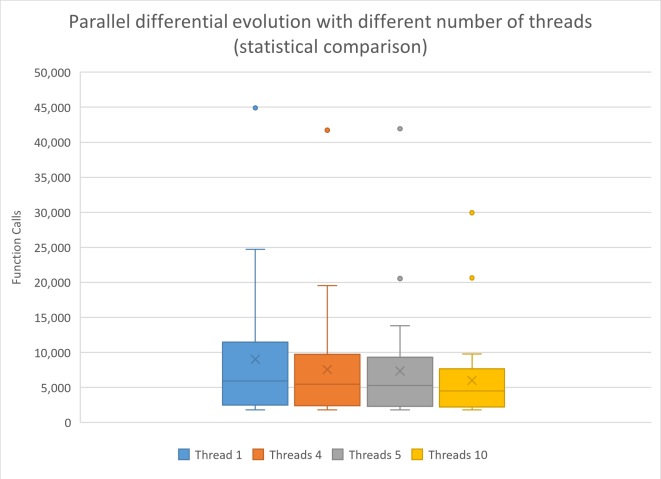

Figure 2. Statistical comparison for the proposed method and different numbers of threads.

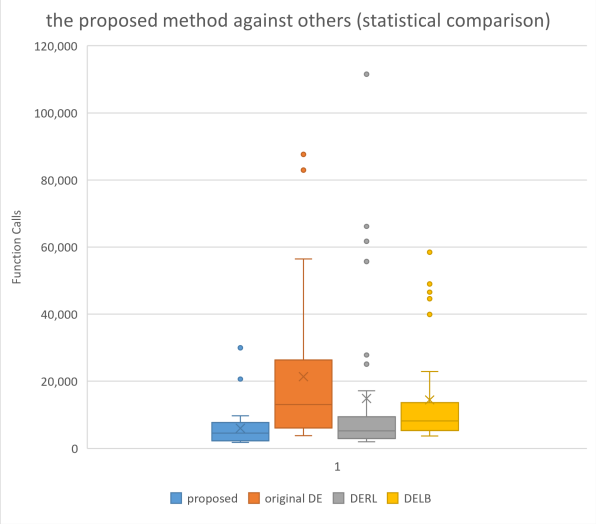

Figure 3. Statistical comparison for the results of the proposed method against different variations of the DE method.

4. Conclusions

A new global optimization technique was presented in this manuscript, which can be performed in parallel computing environments. This method was based on the well-known differential evolutionary technique and partitioned the initial population of agents, so as to create independent populations executed on parallel computing units. The parallel units periodically exchanged the best values for the objective function with each other, and from the experiments carried out it was found that the most robust information exchange technique was the so-called "1 to 1", where a randomly selected subpopulation exchanges information with another randomly selected subpopulation. Furthermore, a new termination method was proposed which could take full advantage of the parallel computing environment. With this termination rule, the decision to terminate the method could be efficiently made even by a small portion of the independent computing units.

From the experimental results, it appeared that the proposed technique could successfully find the global minimum in a series of problems from the relevant literature, and in fact, as the number of parallel processing units increased, the required number of function calls decreased. Furthermore, after experiments on a difficult problem such as the training of artificial neural networks, it was shown that the time required for the optimization process decreased dramatically with the increase of threads.

However, much can still be done to improve the methodology, such as finding a better way of communication between parallel processing units or even formulating more efficient termination criteria that exploit parallel computing environments. In addition, the proposed technique could be applied to other global optimization techniques such as genetic algorithms or particle swarm optimization.

Author Contributions: I.G.T. and V.C., conceived of the idea and methodology and supervised the technical part regarding the software. I.G.T., conducted the experiments, employed test functions and provided the comparative experiments. V.C., performed the statistical analysis and prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The experiments of this research work were performed at the high-performance computing system established at Knowledge and Intelligent Computing Laboratory, Department of Informatics and Telecommunications, University of Ioannina, acquired with the project "Educational Laboratory equipment of TEI of Epirus" with MIS 5007094 funded by the Operational Programme "Epirus" 2014–2020, by ERDF and national funds.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The experiments of this research work were performed at the high-performance computing system established at Knowledge and Intelligent Computing Laboratory, Department of Informatics and Telecommunications, University of Ioannina, acquired with the project "Educational Laboratory equipment of TEI of Epirus" with MIS 5007094 funded by the Operational Programme "Epirus" 2014–2020, by ERDF and national funds.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Honda, M. Application of genetic algorithms to modelings of fusion plasma physics. *Comput. Phys. Commun.* **2018**, *231*, 94–106. [CrossRef]
2. Luo, X.L.; Feng, J.; Zhang, H.H. A genetic algorithm for astroparticle physics studies. *Comput. Phys. Commun.* **2020**, *250*, 106818. [CrossRef]
3. Aljohani, T.M.; Ebrahim, A.F.; Mohammed, O. Single and Multiobjective Optimal Reactive Power Dispatch Based on Hybrid Artificial Physics–Particle Swarm Optimization. *Energies* **2019**, *12*, 2333. [CrossRef]
4. Pardalos, P.M.; Shalloway, D.; Xue, G. Optimization methods for computing global minima of nonconvex potential energy functions. *J. Glob. Optim.* **1994**, *4*, 117–133. [CrossRef]

5. Liwo, A.; Lee, J.; Ripoll, D.R.; Pillardy, J.; Scheraga, H.A. Protein structure prediction by global optimization of a potential energy function. *Biophysics* **1999**, *96*, 5482–5485. [CrossRef]
6. An, J.; He, G.; Qin, F.; Li, R.; Huang, Z. A new framework of global sensitivity analysis for the chemical kinetic model using PSO-BPNN. *Comput. Chem. Eng.* **2018**, *112*, 154–164. [CrossRef]
7. Gaing, Z.-L. Particle swarm optimization to solving the economic dispatch considering the generator constraints. *IEEE Trans. Power Syst.* **2003**, *18*, 1187–1195. [CrossRef]
8. Basu, M. A simulated annealing-based goal-attainment method for economic emission load dispatch of fixed head hydrothermal power systems. *Int. J. Electr. Power Energy Syst.* **2005**, *27*, 147–153. [CrossRef]
9. Cherruault, Y. Global optimization in biology and medicine. *Math. Comput. Model.* **1994**, *20*, 119–132. [CrossRef]
10. Lee, E.K. Large-Scale Optimization-Based Classification Models in Medicine and Biology. *Ann. Biomed. Eng.* **2007**, *35*, 1095–1109. [CrossRef]
11. Price, W.L. Global optimization by controlled random search. *J. Optim. Theory Appl.* **1983**, *40*, 333–348. [CrossRef]
12. Křivý, I.; Tvrdík, J. The controlled random search algorithm in optimizing regression models. *Comput. Stat. Data Anal.* **1995**, *20*, 229–234. [CrossRef]
13. Ali, M.M.; Törn, A.; Viitanen, S. A Numerical Comparison of Some Modified Controlled Random Search Algorithms. *J. Glob.* **1997**, *11*, 377–385.
14. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
15. Ingber, L. Very fast simulated re-annealing. *Math. Comput. Model.* **1989**, *12*, 967–973. [CrossRef]
16. Eglese, R.W. Simulated annealing: A tool for operational research. *Simulated Annealing Tool Oper. Res.* **1990**, *46*, 271–281. [CrossRef]
17. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
18. Liu, J.; Lampinen, J. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Comput.* **2005**, *9*, 448–462. [CrossRef]
19. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
20. Poli, R.; Kennedy, J.K.; Blackwell, T. Particle swarm optimization An Overview. *Swarm Intell.* **2007**, *1*, 33–57. [CrossRef]
21. Trelea, I.C. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Inf. Process. Lett.* **2003**, *85*, 317–325. [CrossRef]
22. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
23. Socha, K.; Dorigo, M. Ant colony optimization for continuous domains. *Eur. J. Oper. Res.* **2008**, *185*, 1155–1173. [CrossRef]
24. Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*, *Addison*; Wesley Publishing Company: Reading, MA, USA, 1989.
25. Michaelewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*; Springer: Berlin/Heidelberg, Germany, 1996.
26. Grady, S.A.; Hussaini, M.Y.; Abdullah, M.M. Placement of wind turbines using genetic algorithms. *Renew. Energy* **2005**, *30*, 259–270. [CrossRef]
27. Pardalos, P.M.; Romeijn, H.E.; Tuy, H. Recent developments and trends in global optimization. *J. Comput. Appl. Math.* **2000**, *124*, 209–228. [CrossRef]
28. Fouskakis, D.; Draper, D. Stochastic Optimization: A Review. *Int. Stat. Rev.* **2002**, *70*, 315–349. [CrossRef]
29. Rocca, P.; Oliveri, G.; Massa, A. Differential Evolution as Applied to Electromagnetics. *IEEE Antennas Propag. Mag.* **2011**, *53*, 38–49. [CrossRef]
30. Lee, W.S.; Chen, Y.T.; Kao, Y. Optimal chiller loading by differential evolution algorithm for reducing energy consumption. *Energy Build.* **2011**, *43*, 599–604. [CrossRef]
31. Yuan, Y.; Xu, H. Flexible job shop scheduling using hybrid differential evolution algorithms. *Comput. Ind.* **2013**, *65*, 246–260. [CrossRef]
32. Xu, L.; Jia, H.; Lang, C.; Peng, X.; Sun, K. A Novel Method for Multilevel Color Image Segmentation Based on Dragonfly Algorithm and Differential Evolution. *IEEE Access* **2019**, *7*, 19502–19538. [CrossRef]
33. Schutte, J.F.; Reinbolt, J.A.; Fregly, B.J.; Haftka, R.; George, A.D. Parallel global optimization with the particle swarm algorithm. *Int. J. Numer. Methods Eng.* **2004**, *61*, 2296–2315. [CrossRef]
34. Larson, J.; Wild, S.M. Asynchronously parallel optimization solver for finding multiple minima. *Math. Comput.* **2018**, *10*, 303–332. [CrossRef]
35. Tsoulos, I.G.; Tzallas, A.; Tsalikakis, D. PDoublePop: An implementation of parallel genetic algorithm for function optimization. *Comput. Phys. Commun.* **2016**, *209*, 183–189. [CrossRef]
36. Kamil, R.; Reiji, S. An Efficient GPU Implementation of a Multi-Start TSP Solver for Large Problem Instances. In Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation, Philadelphia, PA, USA, 7–11 July 2012; pp. 1441–1442.
37. Van Luong, T.; Melab, N.; Talbi, E.G. GPU-Based Multi-start Local Search Algorithms. In *Learning and Intelligent Optimization*; Coello, C.A.C., Ed.; LION 2011. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6683. [CrossRef]
38. Barkalov, K.; Gergel, V. Parallel global optimization on GPU. *J. Glob. Optim.* **2016**, *66*, 3–20. [CrossRef]

39. Weber, M.; Neri, F.; Tirronen, V. Shuffle or update parallel differential evolution for large-scale optimization. *Soft Comput.* **2011**, *15*, 2089–2107. [CrossRef]
40. Chen, Z.; Jiang, X.; Li, J.; Li, S.; Wang, L. PDECO: Parallel differential evolution for clusters optimization. *J. Comput. Chem.* **2013**, *34*, 1046–1059. [CrossRef]
41. Penas, D.R.; Banga, J.R.; Gonzalez, P.; Doallo, R. Enhanced parallel Differential Evolution algorithm for problems in computational systems biology. *Appl. Soft Comput.* **2015**, *33*, 86–99. [CrossRef]
42. Sui, X.; Chu, S.C.; Pan, J.S.; Luo, H. Parallel Compact Differential Evolution for Optimization Applied to Image Segmentation. *Appl. Sci.* **2020**, *10*, 2195. [CrossRef]
43. Skakovski, A.; Jędrzejowicz, P. An island-based differential evolution algorithm with the multi-size populations. *Expert Syst. Appl.* **2019**, *126*, 308–320. [CrossRef]
44. Skakovski, A.; Jędrzejowicz, P. A Multisize no Migration Island-Based Differential Evolution Algorithm with Removal of Ineffective Islands. *IEEE Access* **2022**, *10*, 34539–34549. [CrossRef]
45. Storn, R. On the usage of differential evolution for function optimization. In Proceedings of the North American Fuzzy Information Processing, Berkeley, CA, USA, 19–22 June 1996; pp. 519–523.
46. F. Neri, E. Mininno, Memetic Compact Differential Evolution for Cartesian Robot Control. *IEEE Comput. Intell.* **2010**, *5*, 54–65. [CrossRef]
47. Mininno, E.; Neri, F.; Cupertino, F.; Naso, D. Compact Differential Evolution. *IEEE Trans. Evol.* **2011**, *15*, 32–54. [CrossRef]
48. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 398–417. [CrossRef]
49. Hachicha, N.; Jarboui, B.; Siarry, P. A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics. *Inf. Sci.* **2011**, *181*, 79–91. [CrossRef]
50. Kaelo, P.; Ali, M.M. A numerical study of some modified differential evolution algorithms. *Eur. J. Oper.* **2006**, *169*, 1176–1184. [CrossRef]
51. Corcoran, A.L.; Wainwright, R.L. A parallel island model genetic algorithm for the multiprocessor scheduling problem. In Proceedings of the 1994 ACM Symposium on Applied Computing, SAC '94, Phoenix, AZ, USA, 6–8 March 1994; pp. 483–487.
52. Whitley, D.; Rana, S.; Heckendorn, R.B. Island model genetic algorithms and linearly separable problems. In *Evolutionary Computing*; Series Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1305, pp. 109–125.
53. Powell, M.J.D. A Tolerant Algorithm for Linearly Constrained Optimization Calculations. *Math. Program.* **1989**, *45*, 547–566. [CrossRef]
54. Ali, M.M. Charoenchai Khompatraporn, Zelda B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. *J. Glob. Optim.* **2005**, *31*, 635–672. [CrossRef]
55. Floudas, C.A.; Pardalos, P.M.; Adjiman, C.; Esposoto, W.; Gümüs, Z.; Harding, S.; Klepeis, J.; Meyer, C.; Schweiger, C. *Handbook of Test Problems in Local and Global Optimization*; Kluwer Academic Publishers: Dordrecht, The Netherlands, 1999.
56. Gaviano, M.; Ksasov, D.E.; Lera, D.; Sergeyev, Y.D. Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans. Math. Softw.* **2003**, *29*, 469–480. [CrossRef]
57. Lennard-Jones, J.E. On the Determination of Molecular Fields. *Proc. R. Soc. Lond. A* **1924**, *106*, 463–477.
58. Zabinsky, Z.B.; Graesser, D.L.; Tuttle, M.E.; Kim, G.I. Global optimization of composite laminates using improving hit and run. In *Recent Advances in Global Optimization*; ACM Digital Library: New York, NY, USA, 1992; pp. 343–368.
59. Chandra, R.; Dagum, L.; Kohr, D.; Maydan, D.; McDonald, J.; Menon, R. *Parallel Programming in OpenMP*; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 2001.
60. Charilogis, V.; Tsoulos, I.G.; Tzallas, A.; Karvounis, E. Modifications for the Differential Evolution Algorithm. *Symmetry* **2022**, *14*, 447. [CrossRef]
61. Bishop, C.M. Neural networks and their applications. *Rev. Sci. Instrum.* **1994**, *65*, 1803–1832. [CrossRef]
62. Raymer, M.; Doom, T.E.; Kuhn, L.A.; Punch, W.F. Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. *IEEE Trans. Syst. Man Cybern Part B* **2003**, *33*, 802–813. [CrossRef]
63. Zhong, P.; Fukushima, M. Regularized nonsmooth Newton method for multi-class support vector machines. *Optim. Methods Softw.* **2007**, *22*, 225–236. [CrossRef]