

Article

NeuralMinimizer: A Novel Method for Global Optimization

Ioannis G. Tsoulos ^{1,*}, Alexandros Tzallas ¹ , Evangelos Karvounis ¹  and Dimitrios Tsalikakis ²

¹ Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece

² Department of Engineering Informatics and Telecommunications, University of Western Macedonia, 50100 Kozani, Greece

* Correspondence: itsoulos@uoi.gr

Abstract: The problem of finding the global minimum of multidimensional functions is often applied to a wide range of problems. An innovative method of finding the global minimum of multidimensional functions is presented here. This method first generates an approximation of the objective function using only a few real samples from it. These samples construct the approach using a machine learning model. Next, the required sampling is performed by the approximation function. Furthermore, the approach is improved on each sample by using found local minima as samples for the training set of the machine learning model. In addition, as a termination criterion, the proposed technique uses a widely used criterion from the relevant literature which in fact evaluates it after each execution of the local minimization. The proposed technique was applied to a number of well-known problems from the relevant literature, and the comparative results with respect to modern global minimization techniques are shown to be extremely promising.

Keywords: global optimization; neural networks; stochastic methods

1. Introduction

An innovative method for finding the global minimum of multidimensional functions is presented here. The functions considered are continuous and differentiable and defined as $f : S \rightarrow R, S \subset R^n$. The problem of locating the global optimum is usually formulated as:

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

with S :

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

A variety of problems in the physical world can be represented as global minimum problems, such as problems from physics [1–3], chemistry [4–6], economics [7,8], medicine [9,10], etc. During the past years, many methods, especially the stochastic one, 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. A systematic review of global optimization methods can also be found in the work of Floudas et al. [27]. In addition, during the last few years, a variety of work has been proposed on combinations and modifications to some global optimization methods to more efficiently find the global minimum, such as methods that combine PSO with other methods [28–30], methods aimed to discover all the local minima of functions [31–33], new stopping rules to efficiently terminate the global optimization techniques [34–36], etc. In addition, due to the massive use of parallel processing techniques, several methods have been proposed that take full advantage of parallel processing, such as parallel techniques [37–39], methods that utilize the GPU architectures [40,41], etc.

In addition, during the past years, many metaheuristic algorithms have appeared to tackle global optimization problems such as the Quantum-based avian navigation optimizer



Citation: Tsoulos, I.G.; Tzallas, A.; Karvounis, E.; Tsalikakis, D. NeuralMinimizer: A Novel Method for Global Optimization. *Information* **2023**, *14*, 66. <https://doi.org/10.3390/info14020066>

Academic Editor: Enrique Onieva

Received: 19 December 2022

Revised: 19 January 2023

Accepted: 20 January 2023

Published: 25 January 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

algorithm [42], a Tunicate Swarm Algorithm (TSA) inspired by simulating the lives of Tunicates at sea and how food is obtained [43], Starling murmuration optimizer [44,45], the Diversity-maintained multi-trial vector differential evolution algorithm (DMDE) used in large-scale global optimization [46], an improved moth-flame optimization algorithm with an adaptation mechanism to solve numerical and mechanical engineering problems [47], the dwarf-mongoose optimization algorithm [48], etc.

In this paper, a new multistart method is proposed that uses a machine learning model, which is trained in parallel with the evolution of the optimization process. Although multistart methods are considered the basis for more modern optimization techniques, they have been successfully used in several problems such as the Traveling Salesman Problem (TSP) [49–51], the maximum clique problem [52,53], the vehicle routing problem [54,55], scheduling problems [56,57], etc. In the new technique, a Radial Basis Function (RBF) network [58] is used to construct an approximation of the objective function. This construction is carried out in parallel with the execution of the optimization. A limited number of samples from the objective function and the local minima discovered during the optimization are used to construct the approximation function. During the execution of the method, the samples needed to start local minimizers are taken from the approximation function that is constructed by the neural network. The RBF network was used as an approximation tool as it has been successfully used in a wide range of problems in the field of artificial intelligence [59–62] and its training procedure is very fast, if compared to artificial neural networks, for example. In addition, for a more efficient termination of the method, a termination method proposed by Tsoulos is used [63], but this termination method is applied after each execution of the local minimization procedure. The mentioned method was applied to some test functions provided by the relevant literature and the results are extremely promising as compared with other global optimization techniques.

The proposed method does not sample the actual function but an approximation of it, which is generated incrementally. The creation of the approximation is done by using an RBF neural network, known for its reliability and its ability to efficiently approximate functions. The initial approximation is created from a limited number of points, and then, it will be improved through the local minimizers that will be found during the execution of the method. With the above procedure, the required number of function calls is drastically reduced, since the actual function is not used to produce samples, but an approximation of them. Only samples with low function values are taken from the approximation function, which means that finding the global minimum is likely to be performed faster than other techniques and more efficiently. Furthermore, the generation of the approximation function does not use any prior knowledge about the objective problem.

The rest of this article is organized as follows: in Section 2, the description of the proposed method is provided. In Section 3, the used experimental functions as well as the experimental results and comparisons are listed, and finally, in Section 4, some conclusions and final thoughts are given.

2. Method Description

The proposed technique generates an estimation of the objective function during the optimization using an RBF network. This estimation is initially generated from some samples from the objective function and gradually local minima that will have been discovered during the optimization are added to it. In this way, the estimation of the objective function will be continuously improved to approximate the true function as much as possible. At every iteration, several samples are then taken from the estimated function and sorted in ascending order. Those with the lowest value will be starting points of the local minimization method. The local optimization method used here is a BFGS variant of Powell [64]. This process has the effect of drastically reducing the total number of function calls that are made and, at the same time, the points used as initiators of the local minimization technique approach the global minimum of the objective function. In addition, the proposed method checks the termination rule after the application of every local search method. That way,

if the absolute minimum has already been discovered with some certainty, no more function calls will be wasted finding it.

In the following subsections, the training procedure of RBF networks as well as the proposed method are fully described.

2.1. RBF Preliminaries

An RBF network can be defined as:

$$N(\vec{x}) = \sum_{i=1}^k w_i \phi(\|\vec{x} - \vec{c}_i\|) \quad (2)$$

where

1. The vector $\vec{x}$ is called the input pattern to the equation.
2. The vectors $\vec{c}_i$, $i = 1, \dots, k$ are called the center vectors.
3. The vector $\vec{w}$ stands for the the output weight of the RBF network.

In most cases, the function $\phi(x)$ is a Gaussian function:

$$\phi(x) = \exp\left(-\frac{(x-c)^2}{\sigma^2}\right) \quad (3)$$

The training error for the RBF network on a set of points $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_M, y_M)\}$ is estimated as

$$E(N(\vec{x})) = \sum_{i=1}^M (N(x_i) - y_i)^2 \quad (4)$$

In most approaches, Equation (4) is minimized with respect to the parameters of the RBF network using a two-phase procedure:

1. In the first phase, the K-Means algorithm [65] is used to approximate the k centers and the corresponding variances.
2. In the second phase, the weight vector $\vec{w} = (w_1, w_2, \dots, w_k)$ is calculated by solving a linear system of equations as follows:
 - (a) **Set** $W = w_{kj}$.
 - (b) **Set** $\Phi = \phi_j(x_i)$.
 - (c) **Set** $T = \{t_i = f(x_i), i = 1, \dots, M\}$.
 - (d) The system to be solved is identified as:

$$\Phi^T (T - \Phi W^T) = 0 \quad (5)$$

With the solution:

$$W^T = (\Phi^T \Phi)^{-1} \Phi^T T = \Phi^\dagger T \quad (6)$$

2.2. The Main Algorithm

The main steps of the proposed algorithm are as follows:

1. **Initialization** step.
 - (a) **Set** k the number of weights in the RBF network.
 - (b) **Set** N_S the initial samples that will be taken from the function $f(x)$.
 - (c) **Set** N_T the number of samples that will be used in every iteration as starting points for the local optimization procedure.
 - (d) **Set** N_R the number of samples that will be drawn from the RBF network at every iteration with $N_R > N_T$.
 - (e) **Set** N_G the maximum number of allowed iterations.
 - (f) **Set** $\text{Iter} = 0$, the iteration number.
 - (g) **Set** (x^*, y^*) as the global minimum. Initially, $y^* = \infty$
2. **Creation** Step.
 - (a) **Set** $T = \emptyset$, the training set for the RBF network.
 - (b) **For** $i = 1, \dots, N_S$ **do**
 - i. **Take** a new sample $x_i \in S$.
 - ii. **Calculate** $y_i = f(x_i)$.
 - iii. $T = T \cup (x_i, y_i)$.
 - (c) **End For**
 - (d) **Train** the RBF network on the training set T .
3. **Sampling** Step.
 - (a) **Set** $T_R = \emptyset$.
 - (b) **For** $i = 1, \dots, N_R$ **do**
 - i. **Take** a random sample (x_i, y_i) from the RBF network.
 - ii. **Set** $T_R = T_R \cup (x_i, y_i)$.
 - (c) **End For**
 - (d) **Sort** T_R according to the y values in ascending order.
4. **Optimization** Step.
 - (a) **For** $i = 1, \dots, N_T$ **do**
 - i. **Take** the next sample (x_i, y_i) from T_R .
 - ii. $y_i = \text{LS}(x_i)$, where $\text{LS}(x)$ is a predefined local search method.
 - iii. $T = T \cup (x_i, y_i)$; this step updates the training set of the RBF network.
 - iv. **Train** the RBF network on the set T .
 - v. **If** $y_i \leq y^*$, then $x^* = x_i, y^* = y_i$.
 - vi. **Check** the termination rule as suggested in [63]. If it holds, then report (x^*, y^*) as the located global minimum and terminate.
 - (b) **End For**
5. **Set** $\text{iter} = \text{iter} + 1$.
6. **Goto** to Sampling step.

The steps of the algorithm are illustrated graphically in Figure 1.

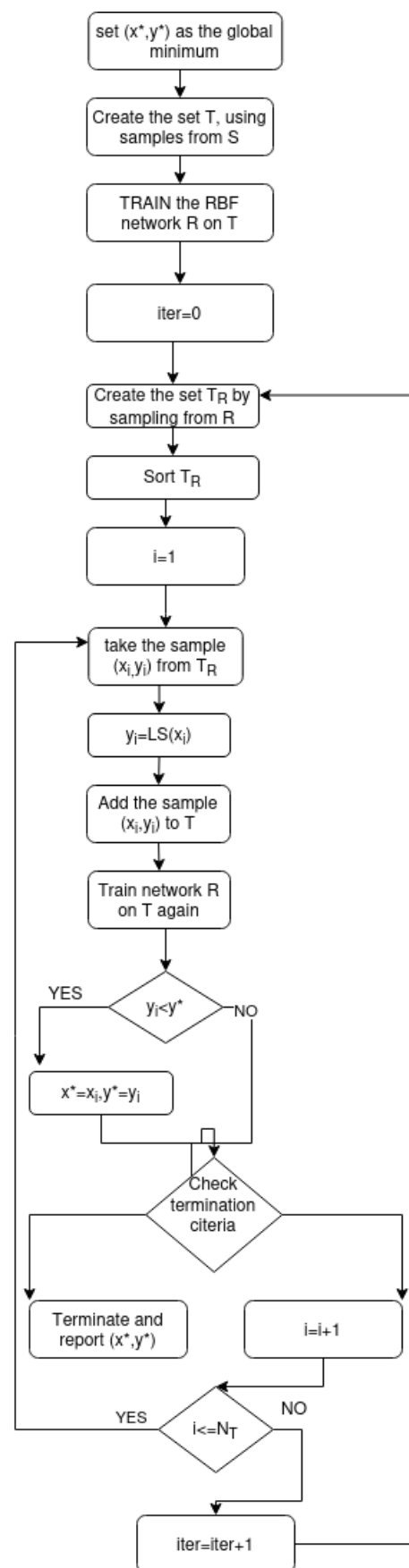


Figure 1. The steps of the proposed algorithm.

3. Experiments

To estimate the efficiency of the new technique, a number of functions from the relevant literature were used [66,67]. These functions are provided in Appendix A. The proposed technique was tested on these test functions and the results produced were compared with those given by a simple genetic algorithm or a PSO method or differential evolution (DE) method. The used genetic algorithm is based on the $GA(c_{r1}, l)$ algorithm from the work of Kaelo and Ali [68]. In order to have fairness in the comparison of the results, for all global optimization techniques, the same local minimization method as that of the proposed method has been used. In addition, the number of chromosomes in the genetic algorithm and the number of particles in the PSO method are identical to the parameter N_T of the proposed procedure. In addition, for the DE method, the number of agents was set to N_T . The values for the parameters used in the conducted experiments are shown in Table 1. For every function and for every global optimizer, 30 independent runs were executed using a different seed for the random generator each time. The proposed method is implemented as the method with the name NeuralMinimizer in the OPTIMUS global optimization environment, which is freely available from <https://github.com/itsoulos/OPTIMUS> (accessed on 25 January 2023). All the experiments were conducted on an AMD Ryzen 5950X with 128 GB of RAM and the Debian Linux operating system.

Table 1. Experimental settings.

Parameter	Meaning	Value
k	Number of weights	10
N_S	Start samples	50
N_T	Number of samples used as starting points	100
N_R	Number of samples that will be drawn from the RBF network	$10 \times N_T$
N_C	Chromosomes or Particles or agents	100
N_G	Maximum number of iterations	200

The experimental results from the application of the proposed method and the other methods are shown in Table 2. The number in the cells represent average function calls. The number in parentheses indicates the fraction of runs where the global optimum was successfully discovered. Absence of this fraction indicated that the global minimum is discovered for every execution (100% success). At the end of the table, an additional row named AVERAGE has been added to show the total number of function calls and the average success rate in locating the global minimum. In the experimental results, the superiority of the proposed technique over the other two methods in terms of the number of function calls is clear. The proposed technique requires an average of 90% fewer function calls than the other methods. In addition, the proposed technique appears to be more efficient than the other two as it finds, on average, more often the global minimum of most test functions in the experiments. In addition, the statistical comparison between the global optimization methods is shown in Figure 2.

Table 2. Comparison between the proposed method and the Genetic and PSO methods.

Function	Genetic	PSO	DE	Proposed
BF1	7150	9030 (0.87)	5579	1051
BF2	7504	6505 (0.67)	5598	921
BRANIN	6135	6865 (0.93)	5888	460
CAMEL	6564	5162	6403	778
CIGAR10	11,813	18,803	13,313	1896
CM4	10,537	11,124	9018	1877 (0.87)
DISCUS10	20,208	6039	7797	478
EASOM	5281	2037	7917	258
ELP10	20,337	16,731	2863	2263
EXP4	10,537	9155	5944	750
EXP16	20,131	14,061	3653	885
EXP64	20,140	8958	3692	948
GRIEWANK10	20,151 (0.10)	17,497 (0.03)	16,469 (0.03)	2697
POTENTIAL3	18,902	9936	5452	1192
POTENTIAL5	18,477	12,385	3972	2399
HANSEN	10,708	9104	14,016	2370 (0.93)
HARTMAN3	8481	12,971	4677	642
HARTMAN6	17,723 (0.60)	15,174 (0.57)	14,372 (0.90)	883
RASTRIGIN	6744	7639 (0.97)	6148	1408 (0.80)
ROSENBROCK4	20,815 (0.63)	11,526	16,763	1619
ROSENBROCK8	20,597 (0.67)	16,967	16,631	2444
SHEKEL5	14,456 (0.73)	15,082 (0.47)	13,178	2333 (0.87)
SHEKEL7	16,786 (0.83)	14,625 (0.40)	12,050	1844 (0.93)
SHEKEL10	15,586 (0.80)	12,628 (0.53)	13,107	2451
SINU4	11,908	10,659	9048	802
SINU8	20,115	13,912	16,210	1500 (0.97)
TEST2N4	13,943	12,948	10,864	878 (0.93)
TEST2N5	15,814	13,936 (0.90)	15,259	971 (0.77)
TEST2N6	18,987	15,449 (0.70)	12,839	997 (0.70)
TEST2N7	20,035	16,020 (0.50)	8185 (0.97)	1084 (0.30)
TEST30N3	13,029	7239	4839	1061
TEST30N4	12,889	8051	5070	854
Average	472,596 (0.89)	368,218 (0.86)	296,814 (0.96)	42,994 (0.94)

In addition, Table 3 presents the experimental results for the proposed method and for various values of the parameter N_S . As can be seen, the increase in this parameter does not cause a large increase in the total number of function calls, while, at the same time, it improves to some extent the ability of the proposed technique to find the global minimum.

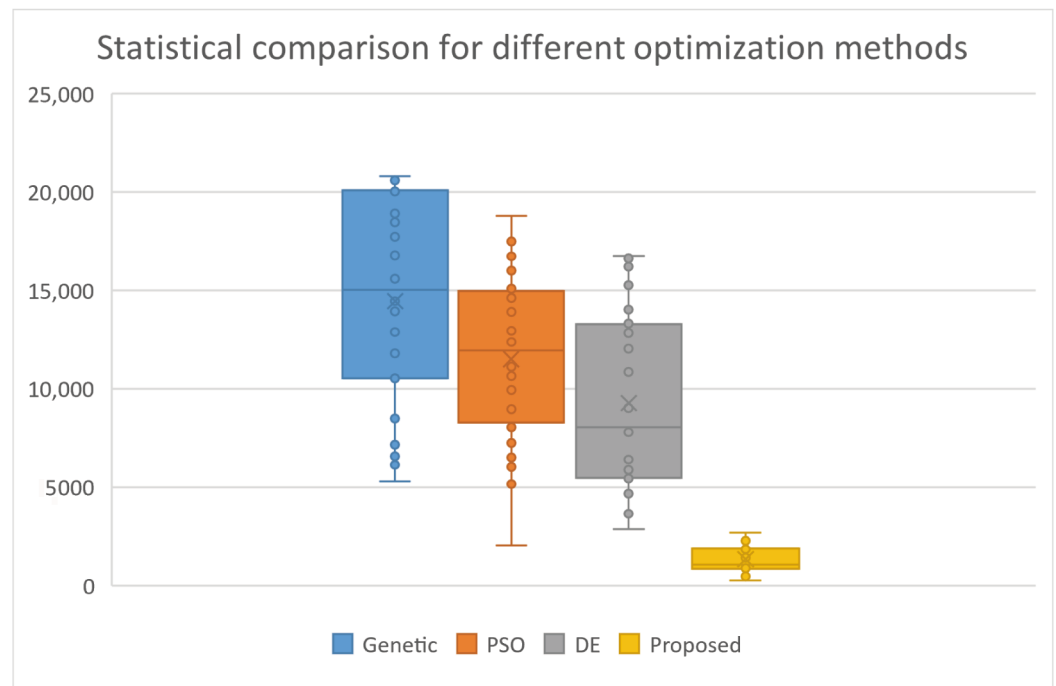


Figure 2. Statistical comparison between the global optimization methods.

Table 3. Experimental results for the proposed method and for different values of the critical parameter N_S (50, 100, 200). Numbers in the cells represent averages of 30 runs.

Function	$N_S = 50$	$N_S = 100$	$N_S = 200$
BF1	1051	1116	1224
BF2	921	949	1058
BRANIN	460	506	599
CAMEL	778	676	739
CIGAR10	1896	1934	2042
CM4	1877 (0.87)	1859 (0.93)	1877 (0.90)
DISCUS10	478	531	634
EASOM	258	307	450
ELP10	2263	2339	3130
EXP4	750	778	884
EXP16	885	932	1030
EXP64	948	998	1091
GRIEWANK10	2697	2647	2801
POTENTIAL3	1192	1228	1305
POTENTIAL5	2399	2417	2544
HANSEN	2370 (0.93)	2602 (0.93)	2578 (0.97)
HARTMAN3	642	696	798
HARTMAN6	883	940	1038
RASTRIGIN	1408 (0.80)	989 (0.83)	1041
ROSENBROCK4	1619	1674	1751
ROSENBROCK8	2444	2499	2583
SHEKEL5	2333 (0.87)	1267	1878 (0.97)

Table 3. *Cont.*

Function	$N_S = 50$	$N_S = 100$	$N_S = 200$
SHEKEL7	1844 (0.93)	1517 (0.93)	1685 (0.97)
SHEKEL10	2451	2695	1498
SINU4	802	821	901
SINU8	1500 (0.97)	1216	1247
TEST2N4	878 (0.93)	934	850 (0.97)
TEST2N5	971 (0.77)	941 (0.80)	993
TEST2N6	997 (0.70)	1087 (0.77)	1098
TEST2N7	1084 (0.30)	1160 (0.53)	1313 (0.57)
TEST30N3	1061	998	1320
TEST30N4	854	830	1108
Average	42,994 (0.94)	42,083 (0.96)	45,088 (0.97)

The efficiency of the method is also shown in Table 4, where the proposed method is compared against the genetic algorithm and particle swarm optimization for a range of number of atoms of the Potential problem. As can be seen in the table, the proposed method requires a significantly smaller number of function calls compared with the other techniques and its reliability in finding the global minimum remains high even when the number of atoms in the potential increases significantly.

Table 4. Optimizing the Potential problem for different number of atoms.

Atoms	Genetic	PSO	Proposed
3	18,902	9936	1192
4	17,806	12,560	1964
5	18,477	12,385	2399
6	19,069 (0.20)	9683	3198
7	16,390 (0.33)	10,533 (0.17)	3311 (0.97)
8	15,924 (0.50)	8053 (0.50)	3526
9	15,041 (0.27)	9276 (0.17)	4338
10	14,817 (0.03)	7548 (0.17)	5517 (0.87)
11	13,885 (0.03)	6864 (0.13)	6588 (0.80)
12	14,435 (0.17)	12,182 (0.07)	7508 (0.83)
13	14,457 (0.07)	10,748 (0.03)	6717 (0.77)
14	13,906 (0.07)	14,235 (0.13)	6201 (0.93)
15	12,832 (0.10)	12,980 (0.10)	7802 (0.90)
Average	205,941 (0.37)	137,134 (0.42)	60,258 (0.93)

4. Conclusions

An innovative technique for finding the global minimum of multidimensional functions was presented in this work. This new technique is based on the multistart procedure, but also generates an estimation of the objective function through a machine learning model. The machine learning model constructs an estimation of the objective function using a small number of samples from the true function but also with the contribution of local minima discovered during the execution of the method. In this way, the estimation of the objective function is continuously improved and the sampling to perform local minimization is done

from the estimated function rather than the actual one. This procedure combined with checking the termination criterion after each execution of the local minimization method led the proposed method to have excellent results both in terms of the speed of finding the global minimum and its efficiency. In addition, the method shows significant stability in its performance even in the presence of large changes of its parameters as presented in the experimental results section.

In the future, the use of the RBF network to construct an approximation of the objective function can be applied to more modern optimization techniques such as genetic algorithms. It would also be interesting to create a parallel implementation of the proposed method, in order to significantly speed up its execution and to be able to be used efficiently in optimization problems of higher dimensions.

Author Contributions: I.G.T., A.T., E.K. and D.T. conceived of the idea and methodology. I.G.T. and A.T. conducted the experiments, employing several test functions and provided the comparative experiments. E.K. and D.T. performed the statistical analysis and all other authors prepared the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

Appendix A

- **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 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.

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

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2$$

The global minimum has the value of $f(x^*) = -1.0316$.

- **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 global minimum -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 values $n = 4, 16, 64$ were used here and the corresponding function names are EXP4, EXP16, EXP64.

- **Griewank10 function,** defined as:

$$f(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

with $n = 10$.

- **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

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

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

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

The value of the global minimum is -3.862782 .

- **Hartman 6 function.**

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

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

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

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 $n = 10$ for the conducted experiments.

- **Potential** function used to represent the lowest energy for the molecular conformation of N atoms via the Lennard–Jones potential [69]. The function is defined as:

$$V_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^6 \right] \quad (\text{A1})$$

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.

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

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

- **Shekel 5** function.

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

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

- **Shekel 10** function.

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

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

- **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 $x^* = (2.09435, 2.09435, \dots, 2.09435)$ with $f(x^*) = -3.5$. For the conducted experiments, the cases of $n = 4, 8$, and $z = \frac{\pi}{6}$ were studied.

- **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 local minima in the specified range and, in our experiments, we used $n = 4, 5, 6, 7$.

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

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

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.

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, A.H. 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. Optim.* **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. *Eur. J. 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, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948. [\[CrossRef\]](#)
20. Poli, R.; Kennedy, J.K.; Blackwell, T. Particle swarm optimization algorithm: 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. Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*; Springer-Verlag: Berlin, 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. Floudas, C.A.; Gounaris, C.E. A review of recent advances in global optimization. *J. Glob. Optim.* **2009**, *45*, 3–38. [\[CrossRef\]](#)
28. Da, Y.; Xiurun, G. An improved PSO-based ANN with simulated annealing technique. *Neurocomputing* **2005**, *63*, 527–533. [\[CrossRef\]](#)
29. Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **2010**, *10*, 629–640. [\[CrossRef\]](#)
30. Pan, X.; Xue, L.; Lu, Y.; Sun, N. Hybrid particle swarm optimization with simulated annealing. *Multimed. Tools Appl.* **2019**, *78*, 29921–29936. [\[CrossRef\]](#)
31. Ali, M.M.; Storey, C. Topographical multilevel single linkage. *J. Glob. Optim.* **1994**, *5*, 349–358. [\[CrossRef\]](#)
32. Salhi, S.; Queen, N.M. A hybrid algorithm for identifying global and local minima when optimizing functions with many minima. *Eur. J. Oper. Res.* **2004**, *155*, 51–67. [\[CrossRef\]](#)
33. Tsoulos, I.G.; Lagaris, I.E. MinFinder: Locating all the local minima of a function. *Comput. Phys. Commun.* **2006**, *174*, 166–179. [\[CrossRef\]](#)
34. Betro, B.; Schoen, F. Optimal and sub-optimal stopping rules for the multistart algorithm in global optimization. *Math. Program.* **1992**, *57*, 445–458. [\[CrossRef\]](#)
35. Hart, W.E. Sequential stopping rules for random optimization methods with applications to multistart local search. *Siam J. Optim.* **1998**, *9*, 270–290. [\[CrossRef\]](#)
36. Lagaris, I.E.; Tsoulos, I.G. Stopping Rules for Box-Constrained Stochastic Global Optimization. *Appl. Math. Comput.* **2008**, *197*, 622–632. [\[CrossRef\]](#)
37. Schutte, J.F.; Reinbolt, J.A.; Fregly, B.J.; Haftka, R.T.; George, A.D. Parallel global optimization with the particle swarm algorithm. *Int. J. Numer. Methods Eng.* **2004**, *61*, 2296–2315. [\[CrossRef\]](#)
38. Larson, J.; Wild, S.M. Asynchronously parallel optimization solver for finding multiple minima. *Math. Comput.* **2018**, *10*, 303–332. [\[CrossRef\]](#)
39. Tsoulos, I.G.; Tzallas, A.; Tsalikakis, D. PDoublePop: An implementation of parallel genetic algorithm for function optimization. *Comput. Phys. Commun.* **2016**, *209*, 183–189. [\[CrossRef\]](#)
40. 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.
41. 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.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6683.
42. Hoda, Z.; Nadimi-Shahraki, M.H.; Gandomi, A.H. QANA: Quantum-based avian navigation optimizer algorithm. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104314.
43. Gharehchopogh, F.S. An Improved Tunicate Swarm Algorithm with Best-random Mutation Strategy for Global Optimization Problems. *J. Bionic Eng.* **2022**, *19*, 1177–1202. [\[CrossRef\]](#)
44. Hoda, Z.; Nadimi-Shahraki, M.H.; Gandomi, A.H. Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. *Comput. Methods Appl. Mech. Eng.* **2022**, *392*, 114616.
45. Nadimi-Shahraki, M.H.; Asghari, Varzaneh, Z.; Zamani, H.; Mirjalili, S. Binary Starling Murmuration Optimizer Algorithm to Select Effective Features from Medical Data. *Appl. Sci.* **2023**, *13*, 564. [\[CrossRef\]](#)
46. Nadimi-Shahraki, M.H.; Hoda, Z. DMDE: Diversity-maintained multi-trial vector differential evolution algorithm for non-decomposition large-scale global optimization. *Expert Syst. Appl.* **2022**, *198*, 116895. [\[CrossRef\]](#)

47. Nadimi-Shahraki, M.H.; Fatahi, A.; Zamani, H.; Mirjalili, S.; Abualigah, L. An improved moth-flame optimization algorithm with adaptation mechanism to solve numerical and mechanical engineering problems. *Entropy* **2021**, *23*, 1637. [\[CrossRef\]](#) [\[PubMed\]](#)
48. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Dwarf mongoose optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2022**, *391*, 114570. [\[CrossRef\]](#)
49. Li, W. A Parallel Multi-start Search Algorithm for Dynamic Traveling Salesman Problem. In *Experimental Algorithms*; Pardalos, P.M., Rebennack, S., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6630.
50. Martí, R.; Resende, M.G.C.; Ribeiro, C.C. Multi-start methods for combinatorial optimization. *Eur. J. Oper. Res.* **2013**, *226*, 1–8. [\[CrossRef\]](#)
51. Pandiri, V.; Singh, A. Two multi-start heuristics for the k-traveling salesman problem. *Opsearch* **2020**, *57*, 1164–1204. [\[CrossRef\]](#)
52. Wu, Q.; Hao, J.K. An adaptive multistart tabu search approach to solve the maximum clique problem. *J. Comb. Optim.* **2013**, *26*, 86–108. [\[CrossRef\]](#)
53. Djeddi, Y.; Haddadene, H.A.; Belacel, N. An extension of adaptive multi-start tabu search for the maximum quasi-clique problem. *Comput. Ind. Eng.* **2019**, *132*, 280–292. [\[CrossRef\]](#)
54. Bräysy, O.; Hasle, G.; Dullaert, W. A multi-start local search algorithm for the vehicle routing problem with time windows. *Eur. J. Oper. Res.* **2004**, *159*, 586–605. [\[CrossRef\]](#)
55. Michallet, J.; Prins, C.; Amodeo, L.; Yalaoui, F.; Vitry, G. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Comput. Oper. Res.* **2014**, *41*, 196–207. [\[CrossRef\]](#)
56. Peng, K.; Pan, Q.K.; Gao, L.; Li, X.; Das, S.; Zhang, B. A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm Evol. Comput.* **2019**, *45*, 92–112. [\[CrossRef\]](#)
57. Mao, J.Y.; Pan, Q.K.; Miao, Z.H.; Gao, L. An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. *Expert Syst. Appl.* **2021**, *169*, 114495. [\[CrossRef\]](#)
58. Park, J.; Sandberg, I.W. Approximation and Radial-Basis-Function Networks. *Neural Comput.* **1993**, *5*, 305–316. [\[CrossRef\]](#)
59. Yoo, S.H.; Oh, S.K.; Pedrycz, W. Optimized face recognition algorithm using radial basis function neural networks and its practical applications. *Neural Netw.* **2015**, *69*, 111–125. [\[CrossRef\]](#)
60. Huang, G.B.; Saratchandran, P.; Sundararajan, N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Trans. Neural Netw.* **2005**, *16*, 57–67. [\[CrossRef\]](#)
61. Majdisova, Z.; Skala, V. Radial basis function approximations: Comparison and applications. *Appl. Math. Model.* **2017**, *51*, 728–743. [\[CrossRef\]](#)
62. Kuo, B.C.; Ho, H.H.; Li, C.H.; Hung, C.C.; Taur, J.S. A Kernel-Based Feature Selection Method for SVM With RBF Kernel for Hyperspectral Image Classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2014**, *7*, 317–326. [\[CrossRef\]](#)
63. Tsoulos, I.G. Modifications of real code genetic algorithm for global optimization. *Appl. Math. Comput.* **2008**, *203*, 598–607. [\[CrossRef\]](#)
64. Powell, M.J.D. A Tolerant Algorithm for Linearly Constrained Optimization Calculations. *Math. Program.* **1989**, *45*, 547–566. [\[CrossRef\]](#)
65. MacQueen, J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Davis Davis, CA, USA, 27 December 1965–7 January 1966; Volume 1, pp. 281–297.
66. Ali, M.M.; Khompatraporn, C.; Zabinsky, Z.B. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. *J. Glob. Optim.* **2005**, *31*, 635–672. [\[CrossRef\]](#)
67. Floudas, C.A.; Pardalos, P.M.; Adjiman, C.; Esposito, W.R.; Gümüs, Z.H.; Harding, S.; Klepeis, J.; Meyer, C.; Schweiger, C. *Handbook of Test Problems in Local and Global Optimization*; Kluwer Academic Publishers: Dordrecht, Germany, 1999.
68. Kaelo, P.; Ali, M.M. Integrated crossover rules in real coded genetic algorithms. *Eur. J. Oper. Res.* **2007**, *176*, 60–76. [\[CrossRef\]](#)
69. Lennard-Jones, J.E. On the Determination of Molecular Fields. *Proc. R. Soc. Lond. A* **1924**, *106*, 463–477.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.