Article

OPTIMUS: a multidimensional global optimization package

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

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece;itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece; v.charilogis@uoi.gr
* Correspondence: itsoulos@uoi.gr;

Abstract: A significant number of applications from many research areas can be considered global optimization problems, such as applications in the area of image processing, medical informatics, economic models, etc. This paper presents a programming tool written in ANSI C++, which researchers can use to formulate the problem to be solved and then make use of the local and global optimization methods provided by this tool to efficiently solve such problems. The main features of the suggested software are: a) Coding of the objective problem in a high level language such as ANSI C++ b) Incorporation of many global optimization techniques to tackle the objective problem c)Parameterization of global optimization methods using user-defined parameters.

Keywords: Global optimization; local optimization; stochastic methods; evolutionary techniques; termination rules.

1. Introduction

The location of the global minimum for a continuous and differentiable function $f : S \to R, S \subset R^n$ is formulated as

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

where the set S is defined as:

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

Methods that aim to locate the global minimum finds application in problems from the area of economics [1,2], problems that appear very often in the area of physics [3,4], chemistry [5, 6], common problems from medicine [7,8], job scheduling problems [9,10], water resources planning [11,12], network security problems [13,14], robotics [15,16] etc. Also, global optimization methods were used on some symmetry problems [17–19] as well as on inverse problems [20–22]. In the relevant literature there are a number of global optimization techniques, such as Adaptive Random Search methods [23,24], Controlled Random Search methods [25,26], Simulated Annealing [27–29], Genetic algorithms [30,31], Ant Colony Optimization [32,33], Particle Swarm Optimization [34,35] etc.

Due to the high importance of the global optimization problem, a variety of hybrid optimization techniques have been proposed to handle the global optimization problem, such as methods that combine Particle Swarm Optimization and Genetic algorithms [36,37], combination of genetic algorithms and fuzzy logic classifier [38], incorporation of genetic algorithm and the K-Means algorithm [39], combination of Particle Swarm Optimization method with Ant Colony Optimization [40–42], methods that combine the Simplex method and Inductive search [43] etc. Also, many hybrid techniques combining local and global optimization have been developed [44–46].

Just a few recent application examples include an adaptive genetic algorithm for crystal structure prediction [47], modeling of fusion plasma physics with genetic algorithms [48], usage of genetic algorithms for astroparticle physics studies [49], parameter extraction of

Citation: Tsoulos, I.G.; Charilogis, V. OPTIMUS : a multidimensional global optimization package. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

solar cells using a Particle Swarm Optimization method [50], a new control approach of a a fleet of Unmanned Aerial Vehicles using the method of Particle Swarm Optimization[51] etc.

However, in most cases global optimization methods require a lot of computing resources to implement both in memory and computing time. Because of the large demands that global optimization methods have on computing power, several techniques have been proposed such as asynchronous methods [52–54], parallel approaches of the Multistart optimization method [55,56] and also some methods that take advantage of modern parallel GPU architectures [57–59].

In this paper, a new integrated computing environment for performing global optimization methods for multidimensional functions is presented and analyzed in detail. In this computing environment, the programmer can code the problem to be solved using a high-level programming language such as C++. In addition to the objective function, the programmer can also provide information that the objective problem should have at the start of the optimization process and in addition can formulate a series of actions that will take place after the optimization process is finished. Subsequently, the researcher can formulate a strategy to solve the problem. In this strategy, the researcher can choose from a series of sampling methods, choose a global minimization method established in the relevant literature and possibly some local minimization method to improve the produced result. Similar software environments can be found, such as the BARON software package [60] for non convex optimization problems, the MERLIN optimization software [61] which is accompanied with the Merlin Control Language compiler to guide the optimization course, the DEoptim software [62] which is an R package implementing the differential evolution algorithm, the PDoublePop optimization software [63] that implements a parallel genetic algorithm for global optimization etc.

The rest of this article is structured as follows: in section 2 the proposed software is outlined in detail, in section 3 some experiments are conducted to show the effectiveness of the proposed software and finally in section 4 some conclusions and guidelines for future work are presented.

2. Software

The suggested software is entirely coded in ANSI C++, using the freely available QT programming library, which can be downloaded from https://qt.io (accessed on 8 February 2023). The researcher should code the objective function and a number of other mandatory functions in the C++ programming language. Also, the researcher should provide the dimension of the objective function as well as the bound of the function (equation 1). Subsequently, the user can select a global optimization method to apply to the problem from a wide range of available methods. Also, the user can extend the series of methods by adding any new method that follows the guidelines of the software. In the following subsections, the installation process of the suggested software will be analyzed and a complete example of running an objective problem will be given.

2.1. Installation

At the present time, the software package can only be installed on computers with the Linux operating system, but in the future it will be able to be installed on other systems as well. The instructions to install the package on a computer are as follows:

1. Download and install the QT programming library from https://qt.io. In most Linux distributions this library can be made available from the relevant distribution repositories.
2. Download and unzip the software from https://github.com/itsoulos/OPTIMUS.
3. Set the *OPTIMUSPATH* environment variable pointing at the installation directory of OPTIMUS e.g. OPTIMUSPATH=/home/user/OPTIMUS/, where user is the user name in the Linux operating system. This step is necessary so that the compiler can locate the necessary files during compilation.

4. Set the *LD_LIBRAPY_PATH* to include the OPTIMUS/lib subdirectory e.g. LD_LIBRAPY_PATH=$LD_LIBRAPY_PATH:$OPTIMUSPATH/lib/:
5. Issue the command: cd $OPTIMUSPATH
6. Execute the compilation script: *./compile.sh*

The compilation will take some minutes in most modern computer systems and when the compilation is complete, the *lib* folder will contain the supported global optimization methods in the form of shared libraries, the *PROBLEMS* folder will contain a number of example optimization problems from the relevant literature, and the *bin* folder will contain the main executable of the software named *OptimusApp*. This executable can be used to apply global optimization techniques to objective problems.

2.2. Implemented global optimization methods

In the following, the global optimization methods present in the proposed software are presented. In most of them, a local optimization method is applied after their end in order to find the global minimum with greater reliability. In the proposed software, each implemented global optimization method has a set of parameters that can determine the global optimization path and the effectiveness of the method. For example, the genetic algorithm contains parameters such as the number of chromosomes or the maximum number of generations allowed. In addition, to make the optimization process easier, each method has been assigned a symbolic name, such as pso for particle swarm optimization. The implemented global optimization methods are:

1. Differential Evolution. The differential evolution method is included in the software as suggested by Storn[64] and denoted as **de**. This global optimization technique has been widely used in areas such as data mining applications [65,66], material design problems [67], feature selection [68], clustering methods [69] etc.
2. Improved Differential Evolution. The modified Differential Evolution method as suggested by Charilogis et al [70] is implemented and denoted as **gende**. This modification of the Differential Evolution method introduces two variations: a new asymptotic stopping rule and a new scheme for a critical parameter of the method.
3. Parallel Differential Evolution. A parallel implementation of the Differential Evolution method as suggested in [71] is considered with the name **ParallelDe**. This parallel technique divides the total work into a number of available parallel computing units, and in each unit an independent Differential Evolution method is executed. The parallelization is performed using the OpenMP programming library [72].
4. Double precision genetic algorithm. A modified genetic algorithm [73] is included in the software and it is denoted as **DoubleGenetic**. Genetic algorithms are typical representatives of evolutionary techniques with many applications such as scheduling problems [74], the vehicle routing problem [75], combinatorial optimization [76], architectural design etc [77].
5. Integer precision genetic algorithm. The method denoted as **IntegerGenetic** is a copy of the **DoubleGenetic** method, but with the usage of integer values as chromosomes. This global optimization method is ideal for problems such as the TSP problem [78,79], path planning [80], Grammatical Evolution applications [81] etc.
6. Improved Controlled Random Search. An improved version of Controlled Random Search as suggested by Charilogis et al [82] is implemented and it is denoted as **gcrs**. This variation modifies the original method by adding a new sampling method, a new stochastic termination rule and a periodical application of a local optimization procedure.
7. Particle Swarm Optimization. A PSO variant denoted as **Pso** is also included in the software. The particle swarm optimization method was applied successfully in a vast number of problems such as parameter extraction of solar cells [83], crystal structure prediction [84], molecular simulations [85] etc.
8. Improved Particle Swarm Optimization. The improved Particle Swarm method as suggested by Charilogis and Tsoulos [86]. The implemented method is denoted as

iPso. The original Particle Swarm Optimization method is enhanced using a new inertia calculation mechanism as well as a novel termination method.

9. Multistart. A simple method that initiates local searches from different initial points is also implemented in the software. Despite its simplicity, the multistart method has been applied on many problems, such as the TSP problem [87], the vehicle routing problem [88], the facility location problem [89], the maximum clique problem [90], the maximum fire risk insured capital problem [91], aerodynamic problems [92] etc
10. Topographical Multi level single linkage. This method is proposed by Ali et al [93] and it is denoted as **Tmlsl** in the implementation.
11. The MinCenter method. In the software presented here, another multistart method has been included, which forms, with the use of the K-Means algorithm for clustering purposes, the regions of attraction for the local minima of the objective problem. This method is denoted as **MinCenter** and it was originally published by Charilogis and Tsoulos [94].
12. NeuralMinimizer. A novel method that incorporates Radial Basis Functions (RBF)[95] to create an estimation of the objective function introduced in [96] is implemented and denoted by the name **NeuralMinimizer**.

2.3. Implemented local optimization methods

All global optimization methods can be enhanced by applying a local minimization method after they are terminated. The parameter used to determine the used local optimization procedure is the $--$localsearch_method parameter. The implemented local optimization methods are the following:

1. The **bfgs** method. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm was implemented using a variant of Powell [97].
2. The **lbfgs** method. The limited memory BFGS method [98] is implemented as an approximation of the BFGS method using a limited amount of computer memory. This local search procedure is ideal for objective functions of higher dimensions.
3. The Gradient descent method. This method is denoted as **gradient** in the software and implements the Gradient Descent local optimization procedure. This local search procedure is used in various problems such as neural network training [99], image registration [100] etc.
4. The **adam** method. The adam local optimizer [101] is implemented also.
5. The Nelder Mead method. The Nelder - Mead simplex procedure for local optimization [102] is also included in the software and it is denoted as **nelderMead**.
6. Hill climbing. The hill climbing local search procedure denoted as **hill** is also implemented. The method has been used in various fields, such as design of photovoltaic power systems [103], load balancing in cloud computing [104] etc.

2.4. Objective problem deployment

The objective problem must be coded in the C++ programming language. The programmer must describe in detail the problem to be solved and must provide the software with detailed information about the dimension of the problem, the value limits of the variables of the problem, the objective function and also the derivative of the function. If the analytical derivative is not available or difficult to calculate then the programmer can program it using finite differences or use some automatic differentiation software, such as the Adept software [105].

2.4.1. Objective function coding

Figure 1 shows an example of objective function. The figure show also the required functions by the proposed software. This code is used for the minimization of the Rastrigin function defined as:

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$$

with $x \in [-1, 1]^2$. The functions shown in the figure 1 have the following meaning:

1. **void** init(QJsonObject data). The function init() is called before the objective function is executed and its purpose is to pass parameters from the execution environment to the objective function. For example, for the Lennard Jones potential [106], the user can pass the number of atoms in the potential using an assignment as follows

```
natoms=data["natoms"].toString().toInt();
```

   If the objective problem does not need a parameter, then this function can be empty.
2. **int** getDimension(). This function returns the dimension of the objective problem.
3. **void** getmargins(vector<Interval> &x). The getmargins() functions returns in the vector x the bounds of the objective problem. The class Interval is a simple class located in the folder *PROBLEMS* of the distribution, that represents double precision intervals eg the value $[2,4]$ is an interval with left bound the value 2 and right bound the value 4.
4. **double** funmin(vector<**double**> &x). This function returns the objective problem $f(x)$ for a given point x. The number of elements of the vector x must be the same in number as the return value of the function getDimension().
5. **void** granal(vector<**double**> &x,vector<**double**> &g). This functions stores in vector g the gradient $\nabla f(x)$ for a given point x.
6. QJsonObject done(vector<**double**> &x). This function is executed after the objective function optimization process is completed. The point x is the global minimum for the function $f(x)$. This function can be used in various cases, such as to generate a graph after the optimization is finished or even in the case of artificial neural networks [107,108] to apply the resulting trained network to the test set of the problem.

2.4.2. Objective function compilation

In order to build the objective function the user should create an accompaniment project file as demonstrated in Figure 2. The first line of this project file determines the a shared library will be build by the compilation. The second line lists the number of source files that will be compiled and the last line contains the names of the header files. The software incorporates the utility qmake of the QT library to compile the objective function. The compilation is performed with the following series of commands in the terminal:

1. qmake *file.pro*
2. make

where *file.pro* stands for the name of the project file. The final outcome of this compilation will be the shared library *libfile.so*

2.4.3. Objective function execution

A full working command for the Rastrigin problem using the utility program *OptimusApp* is shown below

```
./OptimusApp --filename=librastrigin.so --opt_method=Pso\ --
    pso_particles=100 --pso_generations=10\
     --localsearch_method=bfgs
```

The parameters for the above command line are as follows:

1. The argument of the option −−filename determines the objective problem in shared library format.
2. The argument of the command line option −−opt_method sets the used global optimization procedure. For this case, the Particle Swarm Optimizer was used.
3. The argument of the option −−pso_particles sets the number of particles of the PSO optimizer.
4. The argument for the command line option −−pso_generations sets the maximum number of generations allowed.

Figure 1. A typical representation of an objective problem, suitable for the OPTIMUS programming tool.

```
# include <math.h>
# include <interval.h>
# include <vector>
# include <stdio.h>
# include <iostream>
# include <QJsonObject>
using namespace std;
extern "C" {
void    init(QJsonObject data) {
}
int     getdimension() {
        return 2;
}
void    getmargins(vector<Interval> &x) {
  for(int i=0;i<x.size();i++)
        x[i]=Interval(-1,1);
}
double  funmin(vector<double> &x) {
        return (x[0]*x[0])+(x[1]*x[1])-cos(18.0*x[0])-cos(18.0*x[1]);
}
void    granal(vector<double> &x,vector<double> &g) {
        g[0]=2.0*x[0]+18.0*sin(18.0*x[0]);
        g[1]=2.0*x[1]+18.0*sin(18.0*x[1]);
}
QJsonObject     done(vector<double> &x) {
return QJsonObject();
}
}
```

Figure 2. The associated project file for the Rastrigin problem.

```
TEMPLATE=lib
SOURCES+=rastrigin.cc interval.cpp
HEADERS += interval.h
```

5. The argument −−localsearch_method sets the used local optimization procedure, that will be applied on the best particle of the PSO procedure when it finishes.

The output of the previous command is shown in figure 3. As it is obvious, the global optimization method is quite close to the global minimum of the function, which is -2. However with the help of the local optimization method applied after its end, this minimum is found with greater numerical accuracy.

The main steps of a typical usage of the software have been also shown in graphical format in Figure 4.

Figure 3. Output for the minimization of the Rastrigin function using the PSO optimizer.

```
Generation     1  value:    -1.7464048
Generation     2  value:    -1.8619942
Generation     3  value:    -1.8852439
Generation     4  value:    -1.9490074
Generation     5  value:    -1.9490074
Generation     6  value:    -1.9490074
Generation     7  value:    -1.9490074
Generation     8  value:    -1.9775267
Generation     9  value:    -1.9972928
Generation    10  value:    -1.9977027
Minimum:         -2.0000000000   Function calls:     1028
```

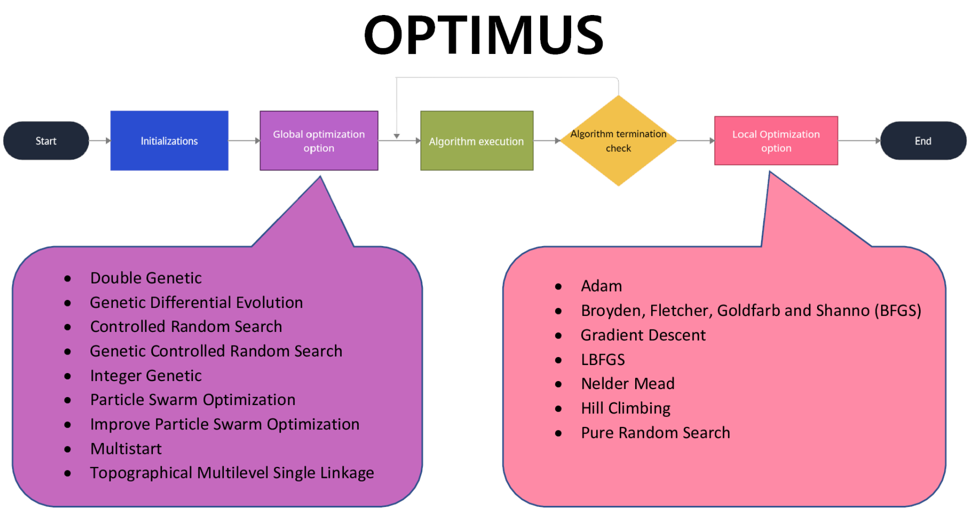

Figure 4. The steps of a typical optimization process using Optimus.

3. Experiments

To assess the ability of the software package to adapt to different problems, a series of experiments were performed under different conditions. In the first series of experiments different global optimization techniques were applied to a series of objective functions that one can locate in the relevant literature. In the second series of experiments, the proposed software was applied to a difficult problem from the field of chemistry, that of finding the minimum potential energy of N interacting atoms of molecules. In the third set of experiments, the scaling of the required number of function calls was evaluated for a parallel technique applied to a difficult problem from the global optimization space, where the problem dimension was constantly increasing. In the last set of experiments, different global optimization techniques were applied to train artificial neural networks.

3.1. Test functions

Some of the proposed methods are tested on a series of well - known test problems from the relevant literature. These problems are used by many researchers in the field. The description of the test functions has as follows:

- **Griewank2** function. This objective function is defined as:

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

- **Rastrigin** function. The function is provided 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 $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$

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

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

Table 1. Experimental settings

PARAMETER	VALUE
CHROMOSOMES	200
CROSSOVER RATE	90%
MUTATION RATE	5%
GENERATIONS	200
LOCAL SEARCH METHOD	bfgs

Table 2. Experimental results for some test functions using a series of global optimization methods.

FUNCTION	GENETIC	GENETIC WITH LOCAL
GRIEWANK2	9298(0.97)	10684
RASTRIGIN	8967	11038
SHEKEl5	19403(0.70)	9222
SHEKEL7	16376(0.80)	8836
SHEKEL10	19829(0.77)	8729
TEST2N4	17109	7786
TEST2N5	19464	8264
TEST2N6	24217	8868
TEST2N7	26824	9376
SUM	**161487(0.92)**	**82803**

This objective function has 2^n local minima in the specified range. During the conducted experiments the values $n = 4, 5, 6, 7$ were used.

The experiments were performed using the above objective functions and ran 30 times using a different seed for the random number generator each time. During the execution of the experiments, the genetic algorithm (DoubleGenetic method) was used as a global optimizer in two versions: one without a local optimization method and one with periodic application of the bfgs method at a rate of 5% on the chromosomes in every generation. The execution parameters for the genetic algorithm are listed in Table 1. The experimental results for the two variants of the genetic algorithm are listed in Table 2. The numbers in cells denote average function calls for the 30 independent runs. The numbers in parentheses show the percentage of finding the global minimum in the 30 runs. If this number is absent, it means that the algorithm discovered the global minimum in all 30 executions. In this table, the line SUM represents the sum of the function calls. The experimental results indicate that the usage of a local search method in combination with the genetic algorithm significantly reduces the required number of average function calls and also improves the reliability of the method in finding the global minimum. Of course, periodically applying a local minimization method to some of the chromosomes drastically increases the required execution time, but the large reduction in the total number of calls required is a big advantage of its application.

3.2. The Lennard Jones potential

The molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard-Jones potential [113,114] is used as a test case here. The function to be minimized is given by:

$$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{2}$$

For testing purposes the method **NeuralMinimizer** of the package was applied to the above problem for a variety of number of atoms and the results are shown in Table 3. This method was experimentally compared with two other techniques in the software package, the method DoubleGenetic and the method Pso. In all cases the number of chromosomes (or particles) was set to 100 and the maximum number of allowed iterations was set to

200. As can be seen from the experimental results, the method NeuralMinimizer requires a significantly reduced number of function calls compared to the other two, while its reliability in finding the global minimum for the potential remains high even when the number of atoms participating in the potential increases significantly.

Table 3. Optimizing the Potential problem for different number of atoms.

ATOMS	GENETIC	PSO	NEURALMINIMIZER
3	18902	9936	1192
4	17806	12560	1964
5	18477	12385	2399
6	19069(0.20)	9683	3198
7	16390(0.33)	10533(0.17)	3311(0.97)
8	15924(0.50)	8053(0.50)	3526
9	15041(0.27)	9276(0.17)	4338
10	14817(0.03)	7548(0.17)	5517(0.87)
11	13885(0.03)	6864(0.13)	6588(0.80)
12	14435(0.17)	12182(0.07)	7508(0.83)
13	14457(0.07)	10748(0.03)	6717(0.77)
14	13906(0.07)	14235(0.13)	6201(0.93)
15	12832(0.10)	12980(0.10)	7802(0.90)
AVERAGE	**205941(0.37)**	**137134(0.42)**	**60258(0.93)**

3.3. *Parallel optimization*

The High Conditioned Elliptic function, defined as

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

is used as a test case to measure the scalability of the parallel global optimization technique denoted as ParallelDe. This method was applied to the problem with dimension increasing from 2 to 15 and for a different number of processing threads. The experimental results are shown in diagram form in Figure 5. As one observes from the figure, the number of calls required to find the global minimum decreases as the total processing threads increase, although the problem becomes increasingly difficult with increasing dimension.

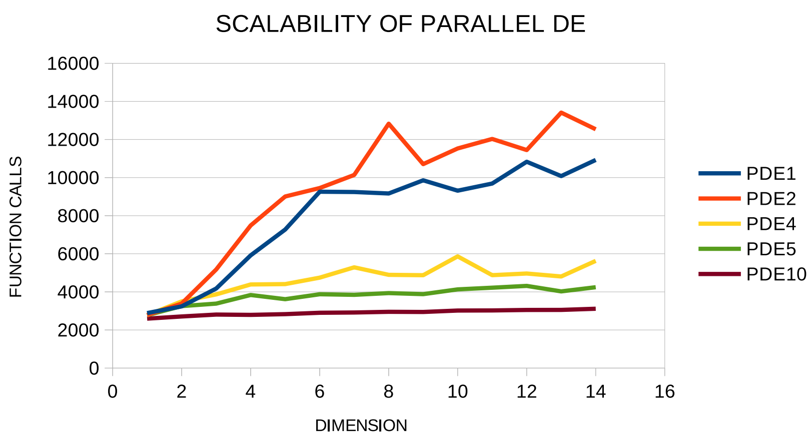

Figure 5. Scalabilty of the ParallelDe method.

3.4. Rbf training

An RBF network is machine learning model and in function form is defined as:

$$y(\overrightarrow{x}) = \sum_{i=1}^{k} w_i \phi(\| \overrightarrow{x} - \overrightarrow{c_i} \|) \tag{3}$$

These models have been used in a wide series of problems, such as solutions of differential equations [115,116], issues of communication [117,118], problems from physics [119,120], chemistry problems [121,122], economics [123–125], network security [126,127] etc. The training error RBF networks is defined as:

$$E(y(x,g)) = \sum_{i=1}^{m} (y(x_i,g) - t_i)^2 \tag{4}$$

In the previous equation, the constant m stands for the number of input patterns and the values t_i represent the expected output for the every input vector x_i. The vector g denotes the parameters of the RBF network. In most cases, the equation 4 is minimized through a two - phase procedure. During the first phase, a subset of the parameters of the network, called centers and variances, are calculated using the well - known K-Means algorithm [128]. During the second phase, a linear system is solved for the calculation of the weight set $w_i,\ i = 1,..,k$.

Here, in this series of experiments two methods of the OPTIMUS package, the genetic algorithm and the particle swarm optimization, was used to minimize the equation 4. The RBF network was applied on a series of classification problems, described in various papers [129,130]. The comparative results are shown in Table 4. The RBF network was implemented in ANSI C++ using the freely available Armadillo library [131]. The experiments were validated using the 10-fold validation technique. Furthermore, all the experiments were conducted 30 times. In each execution a different seed for the random number generator was used. The average classification error on test set is reported in the table with the experimental results and the number of weights k was set to $k = 10$. The table has the following organization:

1. The column DATASET defines the name of the experimental dataset.
2. The column KRBF stands for classic training method for RBF networks using the previous mentioned method of two phases.
3. The column GENETIC stands for the application of a genetic algorithm with 200 chromosomes.
4. The column PSO represents the results for the application of the particle swarm optimization method with 200 particles.
5. An additional line (with the symbolic name AVERAGE) has been added at the end of the table, in which the average classification error for each method is displayed.

As can be deduced from the experimental results, the global optimization methods of the proposed software package were applied with great success to such a difficult and complex problem as that of training a neural network. Furthermore, both genetic algorithm and particle swarm optimization significantly outperform the traditional technique in terms of average test error.

4. Conclusions

In this work, an environment for executing global optimization problems was presented. In this environment, the user can code the objective problem using some predefined functions and then has the possibility to choose one among several global optimization methods to solve the mentioned problem. In addition, it is given the possibility to choose to use some local optimization method to enhance the reliability of the produced results. This programming environment is freely available and easy to extend to accommodate

Table 4. Classification error for different datasets.

DATASET	KRBF	GENETIC	PSO
Alcohol	46.63%	25.08%	29.23%
Appendicitis	12.23%	16.00%	14.93%
Australian	34.89%	24.53%	23.63%
Balance	33.42%	14.98%	15.08%
Dermatology	62.34%	36.41%	35.39%
Glass	50.16%	49.76%	53.83%
Hayes Roth	64.36%	37.18%	37.87%
Heart	31.20%	18.60%	17.38%
HouseVotes	6.13%	3.77%	3.91%
Ionosphere	16.22%	10.49%	11.96%
Liverdisorder	30.84%	28.60%	29.25%
Mammographic	21.38%	17.34%	17.61%
Parkinsons	17.42%	16.63%	16.91%
Pima	25.78%	24.32%	23.87%
Popfailures	7.04%	5.51%	5.84%
Saheart	32.19%	29.33%	28.80%
Sonar	27.85%	20.78%	21.42%
Spiral	44.87%	17.18%	33.67%
Tae	60.07%	53.75%	52.69%
Wdbc	7.27%	5.45%	5.11%
Wine	31.41%	9.37%	8.02%
Z_F_S	13.16%	3.89%	3.74%
ZOO	21.93%	9.53%	11.10%
AVERAGE	**30.38%**	**20.80%**	**21.79%**

more global optimization techniques. It is subject to continuous improvements and some of those planned for the near future are:

1. Possibility to port the Optimus tool to other operating systems such as FreeBSD, Windows etc.
2. Use of modern parallel techniques to speed up the generated results and implementation of efficient termination techniques. At the present time, the ParallelDe has been implemented using parallel techniques and it is expected that parallel implementations will be created for other global minimization techniques as well. In addition, new termination techniques specifically designed for parallel techniques should be devised and implemented.
3. Implementing a GUI interface to control the optimization process.
4. Ability to code the objective function in other programming languages such as Python, Ada, Fortran etc.
5. Creating a scripting language to efficiently guide the optimization of objective functions.

Author Contributions: I.G.T. and V.C. conceived of the idea, the methodology implemented the software. I.G.T. conducted the experiments, employing datasets, 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: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Institutional Review Board 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.

Sample Availability: Not applicable.

References

1. 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.
2. 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.
3. Q. Duan, S. Sorooshian, V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, Water Resources Research **28**, pp. 1015-1031 , 1992.
4. P. Charbonneau, Genetic Algorithms in Astronomy and Astrophysics, Astrophysical Journal Supplement **101**, p. 309, 1995
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. P.M. Pardalos, D. Shalloway, G. Xue, Optimization methods for computing global minima of nonconvex potential energy functions, Journal of Global Optimization **4**, pp. 117-133, 1994.
7. Eva K. Lee, Large-Scale Optimization-Based Classification Models in Medicine and Biology, Annals of Biomedical Engineering **35**, pp 1095-1109, 2007.
8. Y. Cherruault, Global optimization in biology and medicine, Mathematical and Computer Modelling **20**, pp. 119-132, 1994.
9. Y. Gao, H. Rong, J.Z. Huang, Adaptive grid job scheduling with genetic algorithms, Future Generation Computer Systems **21**, pp. 151-161, 2005.
10. D.Y. Sha, H.H. Lin, A multi-objective PSO for job-shop scheduling problems, Expert Systems with Applications **37**, pp. 1065-1070, 2010.
11. X. Cai, D.C. McKinney, L.S. Lasdon, Solving nonlinear water management models using a combined genetic algorithm and linear programming approach, Advances in Water Resources **24**, pp. 667-676, 2001.
12. S.G. Gino Sophia, V. Ceronmani Sharmila, S. Suchitra et al, Water management using genetic algorithm-based machine learning, Soft Comput **24**, pp. 17153–17165, 2020.
13. Z. Bankovic, D. Stepanovic, S. Bojanic, O. Nieto - Taladriz, Improving network security using genetic algorithm approach, Computers & Electrical Engineering **33**, pp. 438-451, 2007.
14. S. Paul, I. Dutt, S.N. Choudhri, Design and implementation of network security using genetic algorithm. Int J Res Eng Technol **2**, pp. 172-177, 2013.
15. A. Tuncer, M. Yildrim, Dynamic path planning of mobile robots with improved genetic algorithm, Computers & Electrical Engineering **38**, pp. 1564-1572, 2012.
16. N. Kherici, Y.M. Ben Ali, Using PSO for a walk of a biped robot, Journal of Computational Science **5**, pp. 743-749, 2014.
17. B. Freisleben and P. Merz, A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems, In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 616-621, 1996.
18. R. Grbić, E.K. Nyarko and R. Scitovski, A modification of the DIRECT method for Lipschitz global optimization for a symmetric function, J Glob Optim **57**, pp. 1193–1212, 2013.
19. R. Scitovski, A new global optimization method for a symmetric Lipschitz continuous function and the application to searching for a globally optimal partition of a one-dimensional set, J Glob Optim **68**, pp. 713–727, 2017.
20. Barbara Kaltenbacher and William Rundell, The inverse problem of reconstructing reaction–diffusion systems, Invese Problems **36**, 2020.
21. N. Levashova, A. Gorbachev, R. Argun, D. Lukyanenko, The Problem of the Non-Uniqueness of the Solution to the Inverse Problem of Recovering the Symmetric States of a Bistable Medium with Data on the Position of an Autowave Front., Symmetry **13**, 2021.
22. Larisa Beilina, Michael V. Klibanov, A Globally Convergent Numerical Method for a Coefficient Inverse Problem, SIAM Journal on Scientific Computing **31**,pp. 478-509, 2008.
23. M. Brunato, R. Battiti, RASH: A Self-adaptive Random Search Method. In: Cotta, C., Sevaux, M., Sörensen, K. (eds) Adaptive and Multilevel Metaheuristics. Studies in Computational Intelligence, vol 136. Springer, Berlin, Heidelberg, 2008.
24. S. Andradóttir, A.A. Prudius, A.A., Adaptive random search for continuous simulation optimization. Naval Research Logistics **57**, pp. 583-604, 2010.
25. W.L. Price, Global optimization by controlled random search, J Optim Theory Appl **40**, pp. 333–348, 1983.
26. P. Kaelo, M.M. Ali, Some Variants of the Controlled Random Search Algorithm for Global Optimization. J Optim Theory Appl **130**, pp. 253–264 (2006).

27. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.
28. K.M.El-Naggar, M.R. AlRashidi, M.F. AlHajri, A.K. Al-Othman, Simulated Annealing algorithm for photovoltaic parameters identification, Solar Energy **86**, pp. 266-274, 2012.
29. L.M. Rasdi Rere, M.I. Fanany, A.M. Arymurthy, Simulated Annealing Algorithm for Deep Learning, Procedia Computer Science **72**, pp. 137-144, 2015.
30. J. Mc Call, Genetic algorithms for modelling and optimisation, Journal of Computational and Applied Mathematics **184**, pp. 205-222, 2005.
31. C.K.H. Lee, A review of applications of genetic algorithms in operations management, Elsevier Engineering Applications of Artificial Intelligence **76**, pp. 1-12, 2018.
32. B. Chandra Mohan, R. Baskaran, A survey: Ant Colony Optimization based recent research and implementation on several engineering domain, Expert Systems with Applications **39**, pp. 4618-4627, 2012.
33. T. Liao, T. Stützle, M.A. Montes de Oca, M. Dorigo, A unified ant colony optimization algorithm for continuous optimization, European Journal of Operational Research **234**, pp. 597-609, 2014.
34. D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview. Soft Comput **22**, pp. 387–408, 2018.
35. N.K. Jain, U. Nangia, J. Jain, A Review of Particle Swarm Optimization. J. Inst. Eng. India Ser. B **99**, pp. 407–411, 2018.
36. D.H. Kim, A. Abraham, J.H. Cho, A hybrid genetic algorithm and bacterial foraging approach for global optimization, Information Sciences **177**, pp. 3918-3937, 2007.
37. Y.T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing **8**, pp. 849-857, 2008.
38. G.T. Reddy, M.P.K. Reddy, K. Lakshmanna et al, Hybrid genetic algorithm and a fuzzy logic classifier for heart disease diagnosis, Evol. Intel. **13**, pp. 185–196, 2020.
39. M.D. Anisur Rahman, M.D. Zahidul Islam, A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowledge-Based Systems **71**, pp. 345-365, 2014.
40. T. Niknam, An efficient hybrid evolutionary algorithm based on PSO and ACO for distribution feeder reconfiguration, European Transactions on Electrical Power **20**, pp. 575-590, 2010.
41. M.K. Patel, M.R. Kabat, C.R. Tripathy, A hybrid ACO/PSO based algorithm for QoS multicast routing problem, Ain Shams Engineering Journal **5**, pp. 113-120, 2014.
42. A.K. Dubey, A. Kumar, R. Agrawal, An efficient ACO-PSO-based framework for data classification and preprocessing in big data, Evol. Intel. **14**, pp. 909–922, 2021.
43. Offord C., Bajzer Ž. (2001) A Hybrid Global Optimization Algorithm Involving Simplex and Inductive Search. In: Alexandrov V.N., Dongarra J.J., Juliano B.A., Renner R.S., Tan C.J.K. (eds) Computational Science - ICCS 2001. ICCS 2001. Lecture Notes in Computer Science, vol 2074. Springer, Berlin, Heidelberg.
44. S. Li, M. Tan, I. W. Tsang, J. T. -Y. Kwok, A Hybrid PSO-BFGS Strategy for Global Optimization of Multimodal Functions, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **41**, pp. 1003-1014, 2011.
45. H. Badem, A. Basturk, A. Caliskan, M.E. Yuksel, A new hybrid optimization method combining artificial bee colony and limited-memory BFGS algorithms for efficient numerical optimization, Applied Soft Computing **70**, pp. 826-844, 2018.
46. A.A. Nagra, F. Han, Q.H. Ling, An improved hybrid self-inertia weight adaptive particle swarm optimization algorithm with local search, Engineering Optimization **51**, pp. 1115-1132, 2018.
47. S.Q. Wu, M. Ji, C.Z. Wang, M.C. Nguyen, X. Zhao, K. Umemoto, R. M. Wentzcovitch, K. M. Ho, An adaptive genetic algorithm for crystal structure prediction, Journal of Physics: Condensed Matter **26**, 035402, 2013.
48. M. Honda, Application of genetic algorithms to modelings of fusion plasma physics, Computer Physics Communications **231**, pp. 94-106, 2018.
49. X.L. Luo, J. Feng, H.H. Zhang, A genetic algorithm for astroparticle physics studies, Computer Physics Communications **250**, 106818, 2020.
50. M. Ye, X. Wang, Y. Xu, Parameter extraction of solar cells using particle swarm optimization, Journal of Applied Physics **105**, 094502. 2009.
51. A. Belkadi, L. Ciarletta, D. Theilliol, Particle swarm optimization method for the control of a fleet of Unmanned Aerial Vehicles, Journal of Physics: Conference Series, Volume 659, 12th European Workshop on Advanced Control and Diagnosis (ACD 2015) 19–20 November 2015, Pilsen, Czech Republic.
52. M. Depolli, R. Trobec, B. Filipič, Asynchronous Master-Slave Parallelization of Differential Evolution for Multi-Objective Optimization, Evolutionary Computation **21**, pp. 261-291, 2013.
53. A. P. Engelbrecht, Asynchronous particle swarm optimization with discrete crossover, In: 2014 IEEE Symposium on Swarm Intelligence, Orlando, FL, USA, 2014, pp. 1-8.
54. F. Bourennani, Cooperative asynchronous parallel particle swarm optimization for large dimensional problems, International Journal of Applied Metaheuristic Computing (IJAMC) **10.3**, pp. 19-38, 2019.
55. J. Larson and S.M. Wild, Asynchronously parallel optimization solver for finding multiple minima, Mathematical Programming Computation **10**, pp. 303-332, 2018.

56. H.P.J. Bolton, J.F. Schutte, A.A. Groenwold, Multiple Parallel Local Searches in Global Optimization. In: Dongarra J., Kacsuk P., Podhorszki N. (eds) Recent Advances in Parallel Virtual Machine and Message Passing Interface. EuroPVM/MPI 2000. Lecture Notes in Computer Science, vol 1908. Springer, Berlin, Heidelberg, 2000.
57. Y. Zhou and Y. Tan, GPU-based parallel particle swarm optimization, In: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 1493-1500.
58. L. Dawson and I. Stewart, Improving Ant Colony Optimization performance on the GPU using CUDA, In: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 1901-1908.
59. Barkalov, K., Gergel, V. Parallel global optimization on GPU. J Glob Optim **66**, pp. 3–20, 2016.
60. N.V. Sahinidis, BARON: A general purpose global optimization software package, J Glob Optim **8**, pp. 201–205, 1996.
61. D.G. Papageorgiou, I.N. Demetropoulos, I.E. Lagaris, Computer Physics Communications **159**, pp. 70-71, 2004.
62. K. Mullen, D. Ardia, D.L. Gil, D. Windover, J. Cline, DEoptim: An R Package for Global Optimization by Differential Evolution, Journal of Statistical Software **40**, pp. 1-26, 2011.
63. 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.
64. R. Storn, On the usage of differential evolution for function optimization, In: Proceedings of North American Fuzzy Information Processing, pp. 519-523, 1996.
65. I. Triguero, S. Garcia, F. Herrera, Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification, Pattern Recognition **44**, pp. 901-916, 2011.
66. Y.H. Li, J.Q. Wang, X.J. Wang, Y.L. Zhao, X.H. Lu, D.L. Liu, Community Detection Based on Differential Evolution Using Social Spider Optimization, Symmetry **9**, 2017.
67. W. Yang, E.M. Dilanga Siriwardane, R. Dong, Y. Li, J. Hu, Crystal structure prediction of materials with high symmetry using differential evolution, J. Phys.: Condens. Matter **33** 455902, 2021.
68. C.Y. Lee, C.H. Hung, Feature Ranking and Differential Evolution for Feature Selection in Brushless DC Motor Fault Diagnosis , Symmetry **13**, 2021.
69. S. Saha, R. Das, Exploring differential evolution and particle swarm optimization to develop some symmetry-based automatic clustering techniques: application to gene clustering, Neural Comput & Applic **30**, pp. 735–757, 2018.
70. V. Charilogis, I.G. Tsoulos, A. Tzallas, E. Karvounis, Modifications for the Differential Evolution Algorithm, Symmetry **14**, 447, 2022.
71. V. Charilogis, I.G. Tsoulos, A Parallel Implementation of the Differential Evolution Method, Analytics **2**, pp. 17-30, 2023.
72. R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.
73. I.G. Tsoulos, Modifications of real code genetic algorithm for global optimization, Applied Mathematics and Computation **203**, pp. 598-607, 2008.
74. J.F.Gonçalves, J.J.M. Mendes, M.G.C. Resende, A genetic algorithm for the resource constrained multi-project scheduling problem, European Journal of Operational Research **189**, pp. 1171-1190, 2008.
75. W.Ho, G.T.S. Ho, P. Ji, H.C.W. Lau, A hybrid genetic algorithm for the multi-depot vehicle routing problem, Engineering Applications of Artificial Intelligence **21**, pp. 548-557, 2008.
76. J.F. Gonçalves, M.G.C. Resende, Biased random-key genetic algorithms for combinatorial optimization. J Heuristics **17**, pp. 487–525, 2011.
77. M. Turrin, P. Buelow, R. Stouffs, Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms, Advanced Engineering Informatics **25**, pp. 656-675, 2011.
78. J. Kaabi, Y. Harrath, Permutation rules and genetic algorithm to solve the traveling salesman problem, Arab Journal of Basic and Applied Sciences **26**, pp. 283-291, 2019.
79. Q.M. Ha, Y. Deville, Q.D. Pham et al., A hybrid genetic algorithm for the traveling salesman problem with drone, J Heuristics **26**, pp. 219–247, 2020.
80. F. Ahmed, K. Deb, Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms, Soft Comput **17**, pp. 1283–1299, 2013.
81. M. O'Neill, C. Ryan, Grammatical Evolution, IEEE Trans. Evolutionary Computation **5**, pp. 349-358, 2001.
82. V. Charilogis, I.G. Tsoulos, A. Tzallas, N. Anastasopoulos, An Improved Controlled Random Search Method, Symmetry **13**, 1981, 2021.
83. M. Ye, X. Wang, Y. Xu, Parameter extraction of solar cells using particle swarm optimization, Journal of Applied Physics **105**, 094502, 2009.
84. Y. Wang, J. Lv, L. Zhu, Y. Ma, Crystal structure prediction via particle-swarm optimization, Phys. Rev. B **82**, 094116, 2010.
85. M. Weiel, M. Götz, A. Klein et al, Dynamic particle swarm optimization of biomolecular simulation parameters with flexible objective functions. Nat Mach Intell **3**, pp. 727–734, 2021.
86. V. Charilogis, I.G. Tsoulos, Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions, Information **13**, 217, 2022.
87. Li W., A Parallel Multi-start Search Algorithm for Dynamic Traveling Salesman Problem. In: Pardalos P.M., Rebennack S. (eds) Experimental Algorithms. SEA 2011. Lecture Notes in Computer Science, vol 6630. Springer, Berlin, Heidelberg, 2011.

88. Olli Bräysy, Geir Hasle, Wout Dullaert, A multi-start local search algorithm for the vehicle routing problem with time windows, European Journal of Operational Research **159**, pp. 586-605, 2004.
89. Mauricio G.C. Resende, Renato F. Werneck,A hybrid multistart heuristic for the uncapacitated facility location problem, European Journal of Operational Research **174**, pp. 54-68, 2006.
90. E. Marchiori, Genetic, Iterated and Multistart Local Search for the Maximum Clique Problem. In: Cagnoni S., Gottlieb J., Hart E., Middendorf M., Raidl G.R. (eds) Applications of Evolutionary Computing. EvoWorkshops 2002. Lecture Notes in Computer Science, vol 2279. Springer, Berlin, Heidelberg.
91. Gomes M.I., Afonso L.B., Chibeles-Martins N., Fradinho J.M. (2018) Multi-start Local Search Procedure for the Maximum Fire Risk Insured Capital Problem. In: Lee J., Rinaldi G., Mahjoub A. (eds) Combinatorial Optimization. ISCO 2018. Lecture Notes in Computer Science, vol 10856. Springer, Cham. https://doi.org/10.1007/978-3-319-96151-4_19
92. Streuber, Gregg M. and Zingg, David. W., Evaluating the Risk of Local Optima in Aerodynamic Shape Optimization, AIAA Journal 59, pp. 75-87, 2012.
93. M.M. Ali, C. Storey, Topographical multilevel single linkage, J. Global Optimization **5**, pp. 349–358,1994
94. V. Charilogis, I.G. Tsoulos, MinCentre: using clustering in global optimisation, International Journal of Computational Intelligence Studies **11**, pp. 24-35, 2022.
95. J. Park, I.W. Sandberg, Approximation and Radial-Basis-Function Networks, Neural Computation **5**, pp. 305-316, 1993.
96. I.G. Tsoulos, A. Tzallas, E. Karvounis, D. Tsalikakis, NeuralMinimizer, a novel method for global optimization that incorporates machine learning, Information **14**, 2, 2023.
97. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
98. D.C. Liu, J. Nocedal, On the Limited Memory Method for Large Scale Optimization, Mathematical Programming B **45**, pp. 503-528, 1989.
99. S.I. Amari, Backpropagation and stochastic gradient descent method, Neurocomputing **5**, pp. 185-196, 1993.
100. S. Klein, J.P.W. Pluim, M. Staring, Adaptive Stochastic Gradient Descent Optimisation for Image Registration, Int J Comput Vis **81**, pp. 227–239, 2009.
101. D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, ICLR (Poster), 2015.
102. D.M. Olsson,L.S. Nelson, The Nelder-Mead Simplex Procedure for Function Minimization, Technometrics **17**, pp. 45-51, 1975.
103. W. Xiao, W. G. Dunford, A modified adaptive hill climbing MPPT method for photovoltaic power systems, In: 2004 IEEE 35th Annual Power Electronics Specialists Conference (IEEE Cat. No.04CH37551) pp. 1957-1963 Vol.3, 2004.
104. B. Mondal, K. Dasgupta, P. Dutta, Load Balancing in Cloud Computing using Stochastic Hill Climbing-A Soft Computing Approach, Procedia Technology **4**, pp. 783-789, 2012.
105. R.J. Hogan, Fast reverse-mode automatic differentiation using expression templates in C++. ACM Trans. Math. Softw. **40**, pp. 1-26, 2014.
106. J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.
107. C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
108. G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.
109. M.M. Ali and P. Kaelo, Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578-593, 2008.
110. H. Koyuncu, R. Ceylan, A PSO based approach: Scout particle swarm algorithm for continuous global optimization problems, Journal of Computational Design and Engineering **6**, pp. 129–142, 2019.
111. Patrick Siarry, Gérard Berthiau, François Durdin, Jacques Haussy, ACM Transactions on Mathematical Software **23**, pp 209–228, 1997.
112. I.G. Tsoulos, I.E. Lagaris, GenMin: An enhanced genetic algorithm for global optimization, Computer Physics Communications **178,** pp. 843-851, 2008.
113. J.A. Northby, Structure and binding of Lennard-Jones clusters: $13 \leq n \leq 147$, J. Chem. Phys. **87**, pp. 6166–6178, 1987.
114. G.L. Xue, R.S. Maier, J.B. Rosen, Improvements on the Northby Algorithm for molecular conformation: Better solutions, J. Global. Optim. 4, pp. 425–440, 1994.
115. Nam Mai-Duy, Thanh Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Networks 14, pp. 185-199, 2001.
116. N. Mai-Duy, Solving high order ordinary differential equations with radial basis function networks. Int. J. Numer. Meth. Engng. **62**, pp. 824-852, 2005.
117. C. Laoudias, P. Kemppi and C. G. Panayiotou, Localization Using Radial Basis Function Networks and Signal Strength Fingerprints in WLAN, GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference, Honolulu, HI, 2009, pp. 1-6, 2009.
118. M. Azarbad, S. Hakimi, A. Ebrahimzadeh, Automatic recognition of digital communication signal, International journal of energy, information and communications **3**, pp. 21-33, 2012.
119. P. Teng, Machine-learning quantum mechanics: Solving quantum mechanics problems using radial basis function networks, Phys. Rev. E **98**, 033305, 2018.

120. R. Jovanović, A. Sretenovic, Ensemble of radial basis neural networks with K-means clustering for heating energy consumption prediction, FME Transactions **45**, pp. 51-57, 2017.
121. D.L. Yu, J.B. Gomm, D. Williams, Sensor fault diagnosis in a chemical process via RBF neural networks, Control Engineering Practice **7**, pp. 49-55, 1999.
122. V. Shankar, G.B. Wright, A.L. Fogelson, R.M. Kirby, A radial basis function (RBF) finite difference method for the simulation of reaction–diffusion equations on stationary platelets within the augmented forcing method, Int. J. Numer. Meth. Fluids **75**, pp. 1-22, 2014.
123. W. Shen, X. Guo, C. Wu, D. Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, Knowledge-Based Systems 24, pp. 378-385, 2011.
124. J. A. Momoh, S. S. Reddy, Combined Economic and Emission Dispatch using Radial Basis Function, 2014 IEEE PES General Meeting | Conference & Exposition, National Harbor, MD, pp. 1-5, 2014.
125. P. Sohrabi, B. Jodeiri Shokri, H. Dehghani, Predicting coal price using time series methods and combination of radial basis function (RBF) neural network with time series. Miner Econ 2021.
126. U. Ravale, N. Marathe, P. Padiya, Feature Selection Based Hybrid Anomaly Intrusion Detection System Using K Means and RBF Kernel Function, Procedia Computer Science **45**, pp. 428-435, 2015.
127. M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, B. Carro, Network Intrusion Detection Based on Extended RBF Neural Network With Offline Reinforcement Learning, IEEE Access **9**, pp. 153153-153170, 2021.
128. J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, No. 14, pp. 281-297, 1967.
129. I.G. Tsoulos, Learning Functions and Classes Using Rules, AI **3**, pp. 751-763, 2022.
130. I.G. Tsoulos, QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution, Algorithms **15**, 295, 2022.
131. C. Sanderson, R. Curtin, Armadillo: a template-based C++ library for linear algebra, Journal of Open Source Software **1**, pp. 26, 2016.