Use RBF as a sampling method in Multistart global optimization method

Ioannis G. Tsoulos[(1)]*, Alexandros Tzallas[(1)], Dimitrios Tsalikakis[(2)]

[(1)]Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece
[(2)]University of Western Macedonia, Department of Engineering Informatics and Telecommunications, Greece

Abstract

In this paper, a new sampling technique is proposed that can be used in the Multistart global optimization technique as well as techniques based on it. The new method takes a limited number of samples from the objective function and then uses them to train an Radial Basis Function (RBF) neural network. Subsequently, several samples were taken from the artificial neural network this time, and those with the smallest network value in them are used in the global optimization method. The proposed technique was applied to a wide range of objective functions from the relevant literature and the results were extremely promising.

Keywords: Global optimization, stochastic methods, termination rules.

1 Introduction

A novel method to draw samples for global optimization methods is presented here. The process of locating the global minimum of a continuous and differentiable function $f : S \to R, S \subset R^n$ is described as, determine

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

with S:

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

The above problem is commonly used to describe problems in economics [1, 2, 3], physics [4, 5, 6], chemistry [7, 8, 9], medicine [10, 11] etc. The global optimization methods have two major categories: deterministic and stochastic methods. The most common methods of the first category are the so-called

*Corresponding author. Email: itsoulos@uoi.gr

Interval methods [12, 13, 14], where the set S is divided iteratively in subregions and some subregions that do not contain the global solution are discarded using some pre-defined criteria. The majority of the methods belong to the second category where the reader can find Controlled Random Search methods [15, 16, 17], Simulated Annealing methods [18, 19], Differential Evolution methods [20, 21], Genetic algorithms [22, 23, 24], Particle Swarm optimization methods [25, 26], Ant Colony methods [27, 28] etc. Also, many hybrid stochastic methods have appeared recently in the relevant literature. such as methods that combine Particle Swarm Optimization and Simulated Annealing [29, 30], methods that combine Genetic Algorithms and Differential Evolution [31, 32], combinations of Genetic Algorithms and Particle Swarm Optimization [33] etc. Also, due to the wide spread of parallel architectures in recent years as well as the widespread use of Graphics Processing Units (GPU), many methods have emerged that exploit such architectures [34, 35, 36].

This paper proposes an innovative sampling technique for the Multistart stochastic global sampling method. The Multistart technique is one of the simplest stochastic global optimization techniques and is the basis for many modern global optimization methods. In the Multistart method, a series of random samples are taken from the objective function and then a local optimization method is started from each sample. Regarding its simplicity, the method has been used with success in a wide area of practical applications, such as the Travelling Salesman Problem (TSP) [37, 38, 39], the vehicle routing problem [40, 41], the facility location problem [42], the maximum clique problem [43], the maximum fire risk insured capital problem [44], aerodynamic shape problems [45] etc. In addition, the Multistart method has been thoroughly studied by many researchers in recent years, and many works have been proposed on this method, such as methods for finding all local minima of a function [46, 47, 48], hybrid techniques [49, 50], GRASP methods[51], new termination rules [52, 53, 54], parallel techniques[55, 56]. Usually, in the Multistart method, samples are used from the objective function using some distribution such as the uniform distribution. In the present work, it is proposed that these samples are obtained from an RBF network [57] , which has already been trained on a limited number of real samples from the objective function. RBF networks have been widely used in many real world problems, such as face recognition [58], function approximation [59, 60], image classification [61], water quality prediction [62] etc.

The proposed sampling methodology generates an approximation of the objective function by first taking some samples from it and then trains a neural network to approximate the function. Once the neural network training process is completed, a bunch of points can be sampled from the neural network and those with the lowest functional value will be used as starting points for the Multistart method. This way, the actual function will not be sampled but the neural network approximating it, which should significantly reduce the required number of function calls. Furthermore, using points with a low function value as starting points is expected to speed up the location of the global minimum. In addition, the RBF neural network is incorporated since it has a very fast training technique.

The rest of this article is organized as follows: in section 2 the proposed sampling technique is outlined in detail, in section 3 the test functions used as well the experimental results are listed and finally in section 4 some conclusions are presented.

2 Method description

2.1 The Multistart method

A commonly used representation of the Multistart method is shown in Algorithm 1. In practice, the method takes N samples at each iteration and starts a local minimization method for each sample, without doing any other checking. However, despite its simplicity, it has two key components which, with proper adaptation, can make the method extremely efficient. The first component is the termination method used and the second is the sampling method within the central iteration. The local search procedure used here is an adaptation of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [63]. The used termination rule was also used in a variety of global optimization methods [64, 65]. This termination method is outlined in subsection 2.2 The second point, which this paper focuses on, is the sampling method. Usually, sampling is done with random samples from some distribution such as the uniform one. In this paper, samples will be taken from an approximation of the objective function $f(x)$ constructed using an RBF neural network. This approach is discussed in subsection 2.3.

Algorithm 1 Representation of the Multistart algorithm.

1. **Initialization** step.
   (a) **Set** N the number of samples, that will taken in every iteration.
   (b) **Set** ITER_{MAX}, the maximum number of allowed iterations.
   (c) **Set** Iter=0, the iteration number.
   (d) **Set** (x^*, y^*) as the global minimum. Initialy $y^* = \infty$
2. **Evaluation** step.
   (a) **Set** Iter=Iter+1
   (b) **For** $i = 1 \ldots N$ **Do**
      i. **Take** a new sample $x_i \in S$
      ii. $y_i = \text{LS}(x_i)$. Where LS(x) is a predefined local search method.
      iii. **If** $y_i \leq y^*$ then $x^* = x_i, y^* = y_i$
   (c) **EndFor**
3. **Termination** check. The termination criteria are checked and if they are true, then the method terminates.

2.2 The used termination rule

A typical termination method used is the maximum number of iterations, i.e. to terminate the method when Iter $\geq \text{ITER}_{\text{MAX}}$. However, this way of termination is not particularly efficient, since for small values of the ITER_{MAX} number the total minimum may not be found, while for larger values of it it may be found at an early stage of the search and then the computer wastes time on calls of local search method. As an example, consider the Hansen function, defined as:

$$f(x) = \sum_{i=1}^{5} i \cos\left[(i-1)x_1 + i\right] \sum_{j=1}^{5} j \cos\left[(j+1)x_2 + j\right], x \in [-10, 10]^2$$

The global minimum for the function is -176.541793. The progress of solving the above function with $\text{ITER}_{\text{MAX}} = 100$ is shown in Figure 1. The global minimum was discovered too early, at the 21th iteration, but the algorithm continues until Iter=100, spending 80% of computing time. The termination rule used in this work was first proposed in [64]: at every iteration n the variance of the quantity $f(x^*)$ is calculated. This quantity is denoted as $v^{(n)}$. If the variance falls below a predetermined threshold, then the method is terminated. This limit is half the value of this variance for the last time a new low was found for y^*. The algorithm terminates when

$$v^{(n)} \leq \frac{v^{(\text{nlast})}}{2} \tag{2}$$

where nlast is the last iteration where a new better estimation of the global minimum was discovered. A graphical representation for the proposed method and the function EXP8 is shown in Figure 2. The value $v^{(n)}$ is denoted as VARIANCE in the plot and the value $\frac{v^{(\text{nlast})}}{2}$ is denoted as STOPAT. The function EXP8 is given by

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{8} x_i^2\right), \quad -1 \leq x_i \leq 1$$

The method now terminates at generation 12.

Figure 1: Progress of Multistart for the Hansen function.

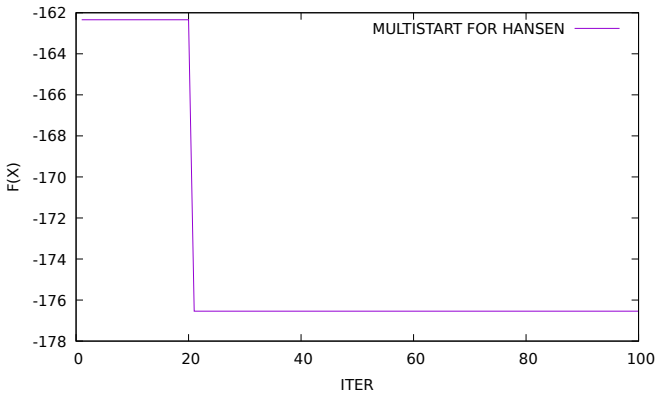
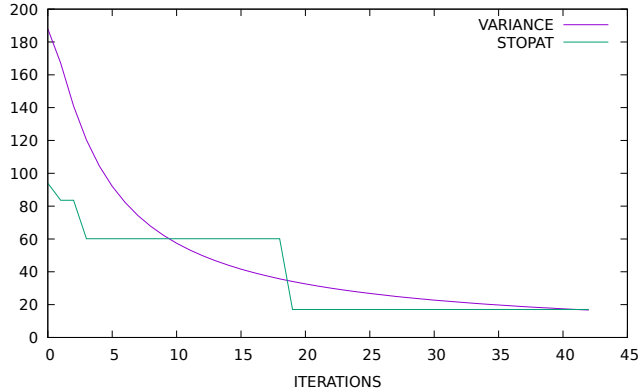


Figure 2: Plot of the used termination rule for the EXP8 test function.



2.3 RBF networks

An RBF neural network typically is expressed as a function:

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

where the vector $\overrightarrow{x}$ stands for the input vector of the network and the vector $\overrightarrow{w}$ is called weight vector with k elements. Typically, the function $\phi(x)$ is the so - called Gaussian function defined as:

$$\phi(x) = \exp\left(-\frac{(x-c)^2}{\sigma^2}\right) \tag{4}$$

where the value $\phi(x)$ depends mainly on the distance between x and x. The vector $\overrightarrow{c}$ is called centroid and the vector $\overrightarrow{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_k)$ is considered as the variance vector. A typical plot of this function is shown in Figure 3.

The network of equation 3 can be used to approximate functions $f(x),\ x \in S \subset R^n$ by minimizing the error:

$$E\left(y\left(\overrightarrow{x}\right)\right) = \sum_{i=1}^{M} \left(y\left(x_i\right) - f\left(x_i\right)\right)^2 \tag{5}$$

where the variable M denotes the number of training samples provided for the function $f(x)$. The RBF network is shown graphically in Figure 4. During a training procedure, the parameters of the RBF network are adapted in order to minimize the error of equation 5 .The RBF network us trained using a two - phase methodology:

1. During the first phase the k centers of and the associated variances are calculated through K-Means algorithm [66].
2. During the second phase, the weight vector $\overrightarrow{w} = (w_1, w_2, \ldots, w_k)$ is calculated by solving a linear system of equations with the following procedure:
   (a) **Set** $W = w_{kj}$, the matrix for the k weights
   (b) **Set** $\Phi = \phi_j\left(x_i\right)$
   (c) **Set** $T = \{t_i = f\left(x_i\right), i = 1, .., M\}$**.**
   (d) The system to be solved is defined as:

$$\Phi^T\left(T - \Phi W^T\right) = 0 \tag{6}$$

The solution is:

$$W^T = \left(\Phi^T\Phi\right)^{-1}\Phi^T T = \Phi^{\dagger} T \tag{7}$$

The matrix $\Phi^{\dagger} = \left(\Phi^T\Phi\right)^{-1}\Phi^T$ is the so - called pseudo-inverse of Φ, with the property

$$\Phi^{\dagger}\Phi = I \tag{8}$$

In the proposed technique, the previously defined network constructs an approximation of the objective function $f(x)$ and subsequently the method Multistart takes samples from the approximation of the objective function. The process starts by taking some samples from the actual $f(x)$ function. These samples are then used to train an RBF neural network. After training, many samples are taken from the neural network function and the best ones will be used in the global optimization method. The overall sampling procedure is shown in Algorithm 2.

The overall algorithm is graphically represented in figure 5 in the form of a flowchart.

Figure 3: Typical plot of the Gaussian function.

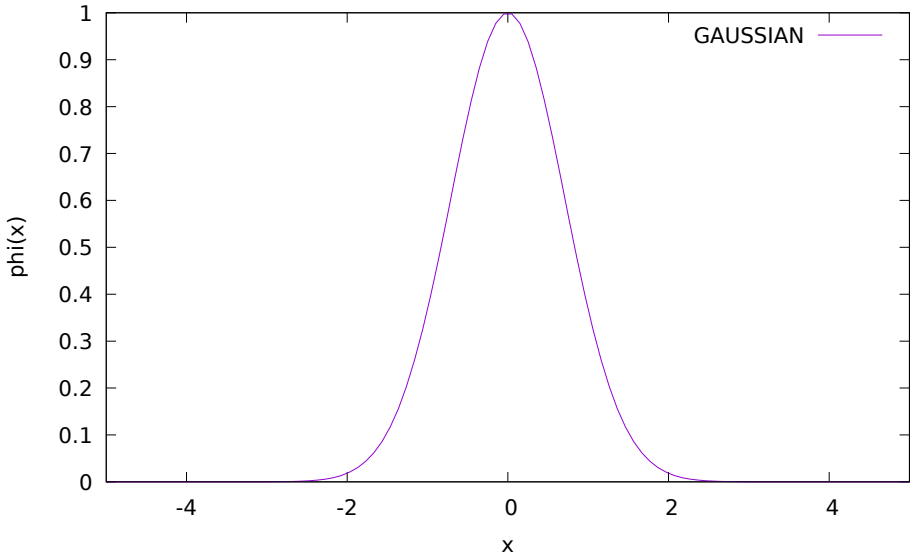
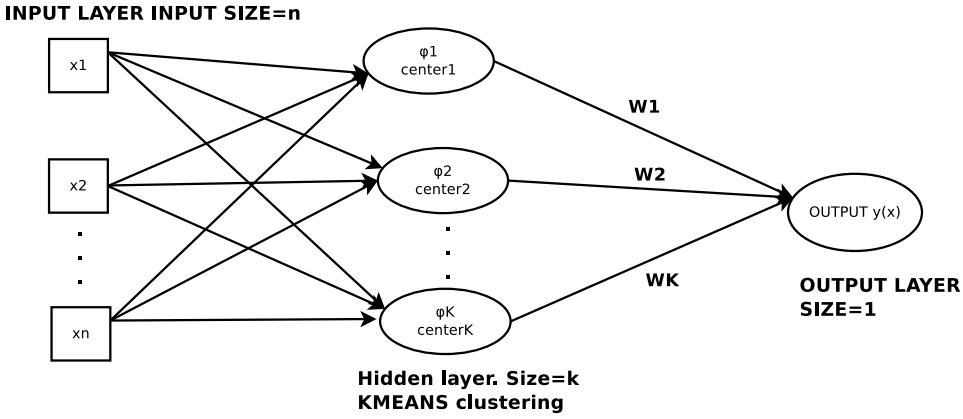


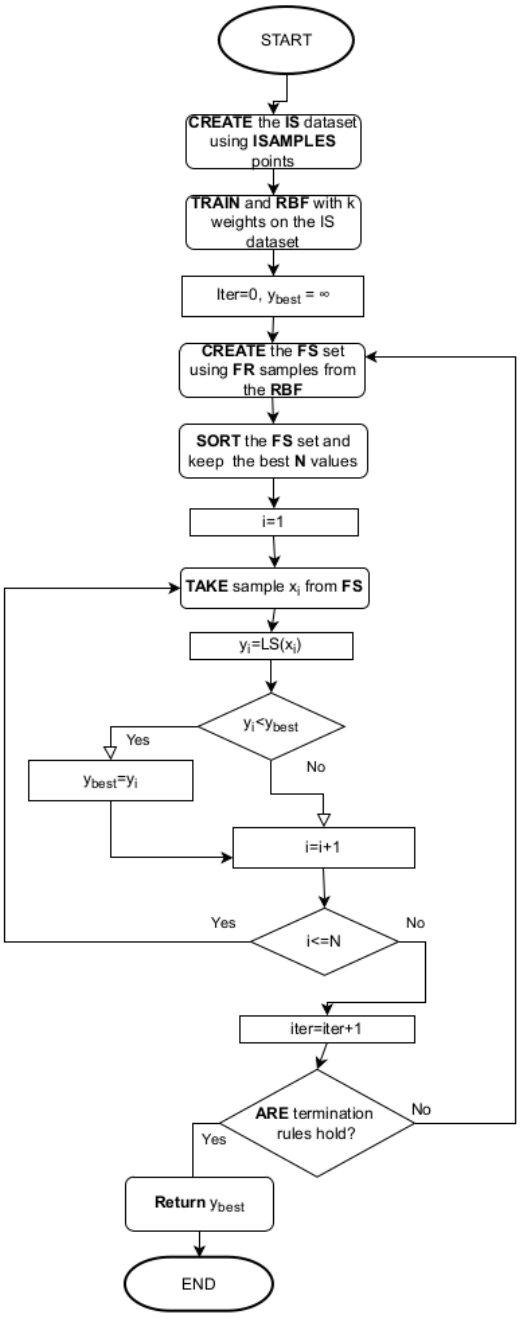
Figure 4: An example of an RBF network.



Algorithm 2 The proposed sampling procedure.

1. **Initialization** step.
   (a) **Set** N, the number of required samples.
   (b) **Set** ISAMPLES, the initial samples that will be drawn from the function $f(x)$.
   (c) **Set** FR a real number, with FR $> N$. For example, FR $= 10 \times N$
   (d) **Set** IS $= \emptyset$
   (e) **Set** FS $= \emptyset$. This is the final outcome of the algorithm.
   (f) **Set** k, the number of weights for the RBF network,
2. **Initial Sampling** step.
   (a) **For** $i = 1, \ldots,$ ISAMPLES**do**
      i. Take a sample $s_i = (x_i, f(x_i)), x_i \in S \subset R^n$
      ii. IS $=$ IS $\cup\, s_i$
   (b) **End For**
3. **Training** step.
   (a) **Construct** an RBF network $y(x)$ with k weights.
   (b) **Train** $y(x)$ using the set IS by minimizing the train error of equation 5.
4. **Final sampling** step.
   (a) **For** $i = 1, \ldots,$ FR **do**
      i. **Take** a sample $s_i = (x_i, y(x_i))$
      ii. FS $=$ FS $\cup\, s_i$
   (b) **End For**
   (c) **Sort** FS according to the function values.
   (d) **Keep** in the set only the N samples with the lowest functional value.

Figure 5: The overall algorithm as a flowchart.



3 Experiments

The effectiveness of the proposed method was evaluated using some benchmark functions from the relevant literature [67, 68].

3.1 Test functions

- ***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 $x \in [-100, 100]^2$. The value of global minimum is 0.0.

- **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 $x \in [-50, 50]^2$. The value of the global minimum is 0.0.

- **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 $x \in [-10, 10]^2$. The value of global minimum is -0.352386.

- **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 $x \in [-1, 1]^n$. The value of the global minimum is -0.4 and in our experiments we have used $n = 4$. The corresponding function is denotes as CM4

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

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

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

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

- **Griewank10** function. The function is given by the equation

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

In our experiments we have used $n = 10$ and the global minimum is 0.0 The function has several local minima in the specified range.

- **Hansen** function. $f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$, $x \in [-10, 10]^2$. The global minimum of the function is -176.541793.

- **Hartman 3** function. The function is given by

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

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

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

The value of global minimum is -3.862782.

- **Hartman 6** function.

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

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

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

the value of global minimum is -3.322368.

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

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

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

- **Potential** function. The molecular conformation corresponding to the global minimum of the energy of N atoms interacting via the Lennard-Jones potential[69] is used as a test case here. The function to be minimized is given by:

$$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] \tag{9}$$

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

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

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

The global minimum is located at $x^* = (0, 0)$ with value -2.0.

- **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 $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$. The value of global minimum is -10.342378.

- **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 $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$. The value of global minimum is -10.107749.

- **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 $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$. The value of global minimum is -10.536410.

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

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

  The global minimum is located at $x^* = (2.09435, 2.09435, ..., 2.09435)$ with $f\left(x^*\right) = -3.5$. In our experiments we used $n = 4, 8, 16$ and $z = \frac{\pi}{6}$ and the corresponding functions are denoted by the labels SINU4, SINU8 and SINU16 respectively.

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

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

  The function has 2^n in the specified range and in our experiments we used $n = 4, 5, 6, 7$. The corresponding values of global minimum is -156.664663 for $n = 4$, -195.830829 for $n = 5$, -234.996994 for $n = 6$ and -274.163160 for $n = 7$.

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

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

  with $x \in [-10, 10]$. The function has 30^n local minima in the specified range and we used $n = 3, 4$ in our experiments. The value of global minimum for this function is 0.0

3.2 Experimental results

The proposed sampling method was tested against the uniform sampling, for the Multistart global optimization technique. The uniform distribution used to sample points is defined as:

$$x_i = a_i + r \times \left(b_i - a_i\right), \; i = 1 \ldots n \tag{10}$$

with $r \in [0, 1]$ a random number. The parameters for the experiments are listed in Table 1. All the experiments were executed 30 times with different random numbers each time. In all cases, the stopping rule of subsection 2.2 was incorporated. The random number function used was the drand48() function of the C programming language. The used software was implemented using the OPTIMUS global optimization environment freely available from `https://github.com/itsoulos/OPTIMUS`. All the experiments were conducted on an AMD Ryzen 5950X equipped with 128GB of RAM. The operating system used was Debian Linux and all the programs are compiled using the GNU C++ compiler. The experimental results for used test functions are listed in Tables

2, 3 and 4. The number in the cells denotes the average function calls for the 30 independent runs. The fraction in parentheses stands for the fraction of runs where the global optimum was found. If this number is missing then the global minimum was discovered in every independent run (100% success). At the end of each table, an additional line named total has been added, representing the total number of function calls and, in parentheses, the average success rate in finding the total minimum.

From the experimental results, it follows in principle that the use of neural networks significantly reduces the required number of function calls needed to find the total minimum. This reduction is proportional to the objective function and can reach up to 60% of the original number of function calls. In addition, the usage of the ISAMPLES parameter significantly increases the reliability of the new sampling method. For example, in the case where N=20 there is an increase in the average success rate for finding the total minimum from 90% to approximately 96%. However, improving the reliability of the method does not imply an increase in the number of function calls. For example, for N=20 the calls are in the interval $[75000, 90000]$ without showing any clear increase. However, the use of the new sampling technique requires significantly more computing time than the uniform distribution because of the need to train the RBF networks and also because of the classification that precedes taking the final samples. This difference is demonstrated in Figure 6. In this we see the significant difference in running time for the SINU problem with a different number of dimensions each time.

Table 1: Parameters for the experiments.

PARAMETER	VALUE
ITER_{MAX}	100
k	10
N	20,50
FR	$10 \times N$

Table 2: Experimental results for the Multistart method, using uniform distribution for the samples as defined in Equation 10.

FUNCTION	N=20	N=50
BF1	3004	5975
BF2	2828	5826
BRANIN	2409	5415
CAMEL	2661	5599
CIGAR	5588	8410
CM4	3551(0.87)	6431(0.80)
DISCUS	2817	5965
EASOM	2204	5202
EXP4	2769	5772
EXP16	2836	5837
EXP64	2912	5914
GRIEWANK2	3938(0.40)	6572(0.30)
GRIEWANK10	4536(0.97)	7520
POTENTIAL3	3121	6120
POTENTIAL5	4363	7320
HANSEN	5344(0.93)	9536(0.90)
HARTMAN3	2618	5608
HARTMAN6	3014	6037
HIGHELLIPTIC	4398	7306
RASTRIGIN	3850(0.83)	6401(0.77)
ROSENBROCK4	6456	8584
ROSENBROCK8	7646	10095
SHEKEL5	3144	6215
SHEKEL7	3354	6508
SHEKEL10	3388	6860
SINU4	3935	6670(0.97)
SINU8	5547	8056
SINU16	19313	35751(0.97)
TEST2N4	3035(0.87)	6002(0.97)
TEST2N5	3127(0.73)	6042(0.67)
TEST2N6	3393(0.40)	6169(0.47)
TEST2N7	4075(0.37)	6443(0.33)
TEST30N3	3723	6322
TEST30N4	3736	6465
TOTAL	**142632(0.923)**	**254988(0.916)**

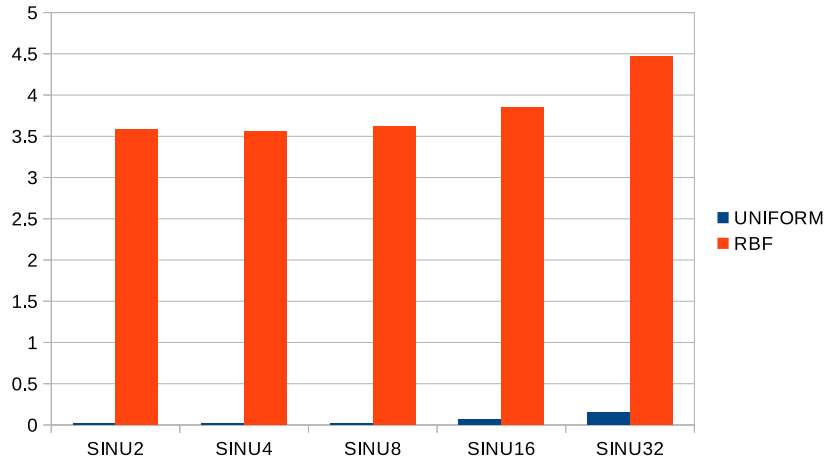
Table 3: Experimental results for the proposed method with N=20

FUNCTION	ISAMPLES=100	ISAMPLES=200	ISAMPLES=500
BF1	1086	1159	1500
BF2	922	1026	1304
BRANIN	503	590	899
CAMEL	670	756	1060
CIGAR	3482	3236	2849
CM4	1583(0.83)	1716(0.83)	1861(0.90)
DISCUS	931	1206	1525
EASOM	1063	401	704
EXP4	766	803	1049
EXP16	912	1009	1303
EXP64	968	1070	1359
GRIEWANK2	2409(0.53)	1641(0.40)	2069(0.57)
GRIEWANK10	2607(0.97)	2609	2902(0.93)
POTENTIAL3	1211	1297	1613
POTENTIAL5	2414	2521	2835
HANSEN	6079(0.87)	4785(0.83)	6504(0.77)
HARTMAN3	729	830	1143
HARTMAN6	1111(0.90)	1290(0.93)	1525(0.97)
HIGHELLIPTIC	2618	2671	3098
RASTRIGIN	1727(0.57)	1043(0.87)	1386
ROSENBROCK4	4111	2672	4357
ROSENBROCK8	5417	6253	5609
SHEKEL5	1751(0.73)	2152(0.90)	1245(0.90)
SHEKEL7	1667(0.87)	1627(0.83)	1676(0.93)
SHEKEL10	2329(0.80)	2946(0.73)	3678(0.77)
SINU4	938	991	1227
SINU8	1194	1360	1479
SINU16	14305(0.87)	32647(0.97)	21363(0.97)
TEST2N4	904(0.57)	936(0.73)	1227
TEST2N5	1881(0.80)	1218	1351
TEST2N6	1092(0.67)	1224(0.87)	1435(0.97)
TEST2N7	1452(0.70)	1397(0.80)	1477(0.90)
TEST30N3	1244	2054	2584
TEST30N4	2027	2644	2638
TOTAL	**74103(0.902)**	**91780(0.932)**	**89834(0.958)**

Table 4: Experimental results for the proposed method with N=50

FUNCTION	ISAMPLES=100	ISAMPLES=200	ISAMPLES=500
BF1	1093	1175	1527
BF2	943	1022	1319
BRANIN	502(0.97)	594	900
CAMEL	642	729	1046
CIGAR	3527	3228	6729
CM4	1491(0.87)	1884(0.90)	1799(0.97)
DISCUS	828	1365	1215
EASOM	2320	398	723
EXP4	766	827	1050
EXP16	912	1007	1298
EXP64	983	1064	1358
GRIEWANK2	1788(0.50)	1762(0.43)	2345(0.50)
GRIEWANK10	2505	2677	2868
POTENTIAL3	1244	1313	1609
POTENTIAL5	2420	2502	2795
HANSEN	6711(0.70)	4278(0.70)	7264(0.67)
HARTMAN3	728	830	1144
HARTMAN6	1027(0.93)	1202(0.93)	1492
HIGHELLIPTIC	3455	2889	3078
RASTRIGIN	977(0.53)	1269(0.77)	1397(0.97)
ROSENBROCK4	2348	2453	3278
ROSENBROCK8	3928	4461	4865
SHEKEL5	5630(0.67)	7498(0.87)	1510(0.93)
SHEKEL7	2135(0.67)	1973(0.67)	1815(0.97)
SHEKEL10	1864(0.73)	1245(0.60)	3165(0.83)
SINU4	984	1020	1355
SINU8	10502	1517	1456
SINU16	95225(0.83)	21658(0.90)	21330(0.87)
TEST2N4	820(0.63)	1079(0.90)	1274
TEST2N5	1140(0.67)	1107(0.80)	1333
TEST2N6	1203(0.73)	1371(0.97)	1440(0.97)
TEST2N7	1602(0.50)	1200(0.77)	1618(0.97)
TEST30N3	1494	1903	2279
TEST30N4	1164	2287	2284
TOTAL	**164901(0.880)**	**82787(0.918)**	**91958(0.960)**

Figure 6: Comparison of execution times for the SINU function between the uniform sampling and the proposed method.



4 Conclusions

In this paper, an innovative sampling technique was proposed for the Multistart global optimization method. The new technique improves on using a limited number of samples from the objective function in order to construct an estimator of the function. The estimator in the present work was an RBF neural network. After the neural network is trained, a large number of samples are taken from the estimator without using the objective function anymore. Of these samples, only those with the lowest functional value of the estimator are used by the global optimization method. From the experiments performed on a wide range of objective functions, many of which had a large number of dimensions, it appears that the proposed technique significantly outperforms the traditionally used uniform sampling. The gain in the number of calls in many cases exceeds 60%. Nevertheless, the new technique requires more computational time than the uniform distribution, since it is required to train the network as well as to classify a series of samples from it. However, this increase in time could be significantly reduced with the potential use of parallel computing techniques to train neural networks. Moreover, in large computational problems, where the cost of evaluating the objective function is extremely large, the training time of a neural network will be almost negligible. Future research may include:

1. Application of the proposed technique to other more efficient global optimization methods.
2. Parallelization of the training method for the neural network.
3. Usage of more efficient methods to train the RBF networks such as Genetic Algorithms.

References

[1] D. Cheong,Y.M. Kim, H.W. Byun, K.J. Oh, T.Y. Kim, Using genetic algorithm to support clustering-based portfolio optimization by investor information, Applied Soft Computing **61**, pp. 593-602, 2017.

[2] J.G. Díaz, B.G. Rodríguez, M. Leal, J. Puerto, Global optimization for bilevel portfolio design: Economic insights from the Dow Jones index, Omega **102**, 102353, 2021.

[3] J. Gao, F. You, Shale Gas Supply Chain Design and Operations toward Better Economic and Life Cycle Environmental Performance: MINLP Model and Global Optimization Algorithm, ACS Sustainable Chem. Eng. **3**, pp. 1282–1291, 2015.

[4] X.L. Luo, J. Feng, H.H. Zhang, A genetic algorithm for astroparticle physics studies, Computer Physics Communications **250**, 106818, 2020.

[5] T. Biekötter, M.O. Olea-Romacho, Reconciling Higgs physics and pseudo-Nambu-Goldstone dark matter in the S2HDM using a genetic algorithm, J. High Energ. Phys. **2021**, 215, 2021.

[6] T. Gu, W. Luo, H. Xiang, Prediction of two-dimensional materials by the global optimization approach, WIREs Computational Molecular Science **7**, e1295, 2017.

[7] H. Fang, J. Zhou, Z. Wang et al, Hybrid method integrating machine learning and particle swarm optimization for smart chemical process operations, Front. Chem. Sci. Eng. **16**, pp. 274–287, 2022.

[8] D. Furman, B. Carmeli, Y. Zeiri, R. Kosloff, Enhanced Particle Swarm Optimization Algorithm: Efficient Training of ReaxFF Reactive Force Fields, J. Chem. Theory Comput. **14**, pp. 3100–3112, 2018.

[9] S. Heiles, R.L. Johnston, Global optimization of clusters using electronic structure methods, Int. J. Quantum Chem. **113**, pp. 2091-2109, 2013.

[10] Eva K. Lee, Large-Scale Optimization-Based Classification Models in Medicine and Biology, Annals of Biomedical Engineering **35**, pp 1095-1109, 2007.

[11] I. Hilali-Jaghdam, A. Ben Ishak, S. Abdel-Khalek, A. Jamal, Quantum and classical genetic algorithms for multilevel segmentation of medical images: A comparative study, Computer Communications **162**, pp. 83-93, 2020.

[12] M.A. Wolfe, Interval methods for global optimization, Applied Mathematics and Computation **75**, pp. 179-206, 1996.

[13] M. Allahdadi, H. M. Nehi, H. A. Ashayerinasab, M. Javanmard, Improving the modified interval linear programming method by new techniques, Information Sciences **339**, pp. 224-236, 2016.

[14] I. Araya, V. Reyes, Interval Branch-and-Bound algorithms for optimization and constraint satisfaction: a survey and prospects, J Glob Optim **65**, pp. 837–866, 2016.

[15] W. L. Price, Global optimization by controlled random search, Journal of Optimization Theory and Applications **40**, pp. 333-348, 1983.

[16] N.M. Filho, R.B.F. Albuquerque, B.S. Sousa, L.G.C. Santos, A comparative study of controlled random search algorithms with application to inverse aerofoil design, Engineering Optimization **50**, pp. 996-1015, 2018.

[17] P. Kaelo, M. M. Ali, Numerical studies of some generalized controlled random search algorithms, Asia-Pacific Journal of Operational Research 29, 2012.

[18] S. Kirkpatrick, CD Gelatt, , MP Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.

[19] A.M. Ferreiro, J.A. García, J.G. López-Salas et al, An efficient implementation of parallel simulated annealing algorithm in GPUs, J Glob Optim **57**, pp. 863–890, 2013.

[20] F. Neri, V. Tirronen, Recent advances in differential evolution: a survey and experimental analysis, Artif Intell Rev **33**, pp. 61–106, 2010.

[21] S. Das, P. N. Suganthan, Differential Evolution: A Survey of the State-of-the-Art, IEEE Transactions on Evolutionary Computation **15**, pp. 4-31, 2011.

[22] O. Kramer, Genetic Algorithms. In: Genetic Algorithm Essentials. Studies in Computational Intelligence, vol 679. Springer, Cham, 2017.

[23] S. Katoch, S.S. Chauhan, V.A. Kumar, A review on genetic algorithm: past, present, and future. Multimed Tools Appl **80**, pp. 8091–8126, 2021.

[24] S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.

[25] Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.

[26] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Comput **22**, pp. 387–408, 2018.

[27] M. Dorigo, M. Birattari and T. Stutzle, Ant colony optimization, IEEE Computational Intelligence Magazine **1**, pp. 28-39, 2006.

[28] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European Journal of Operational Research 185, pp. 1155-1173, 2008.

[29] H.L. Shieh, C.C. Kuo, C.M. Chiang, Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification, Applied Mathematics and Computation **218**, pp. 4365-4383, 2011.

[30] S. Zhoua, X. Liu, Y. Hua, X. Zhou, S.Yang, Adaptive model parameter identification for lithium-ion batteries based on improved coupling hybrid adaptive particle swarm optimization- simulated annealing method, Journal of Power Sources **482**, Article number 228951, 2021.

[31] D. He, F. Wang, Z. Mao, A hybrid genetic algorithm approach based on differential evolution for economic dispatch with valve-point effect, International Journal of Electrical Power & Energy Systems **30**, pp. 31-38, 2008.

[32] A. Trivedi, D. Srinivasan, S. Biswas, T. Reindl, A genetic algorithm – differential evolution based hybrid framework: Case study on unit commitment scheduling problem, Information Sciences **354**, pp. 275-300, 2016.

[33] Y.T. Kao, E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing **8**, pp. 849-857, 2008.

[34] Barkalov, K., Gergel, V. Parallel global optimization on GPU. J Glob Optim **66**, pp. 3–20, 2016.

[35] G. Kan et al., A multi-core CPU and many-core GPU based fast parallel shuffled complex evolution global optimization approach, IEEE Transactions on Parallel and Distributed Systems **28**, pp. 332-344, 2017.

[36] A.M. Ferreiro, J.A. García-Rodríguez, C. Vázquez, E. Costa, A. Correia, Parallel two-phase methods for global optimization on GPU, Mathematics and Computers in Simulation **156**, pp. 67-90, 2019.

[37] 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.

[38] R. Martí, M.G.C. Resende, C.C. Ribeiro, Multi-start methods for combinatorial optimization, European Journal of Operational Research **226**, pp. 1-8, 2013.

[39] V. Pandiri, A. Singh, Two multi-start heuristics for the k-traveling salesman problem, OPSEARCH **57**, pp. 1164–1204, 2020.

[40] 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.

[41] J. Michallet, C. Prins, L. Amodeo, F. Yalaoui, G. Vitry, Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services, Computers & Operations Research **41**, pp. 196-207, 2014.

[42] 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.

[43] 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.

[44] 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

[45] Streuber, Gregg M. and Zingg, David. W., Evaluating the Risk of Local Optima in Aerodynamic Shape Optimization, AIAA Journal **59**, pp. 75-87, 2012.

[46] M.M. Ali, C. Storey, Topographical multilevel single linkage, J. Global Optimization **5**, pp. 349–358,1994

[47] S. Salhi, N.M. Queen, A hybrid algorithm for identifying global and local minima when optimizing functions with many minima, European J. Oper. Res. **155**, pp. 51–67, 2004.

[48] I. G. Tsoulos and I. E. Lagaris, MinFinder: Locating all the local minima of a function, Computer Physics Communications **174**, pp. 166-179, 2006.

[49] H. C. B. d. Oliveira, G. C. Vasconcelos and G. B. Alvarenga, "A Multi-Start Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows," 2006 Ninth Brazilian Symposium on Neural Networks (SBRN'06), Ribeirao Preto, Brazil, 2006, pp. 137-142.

[50] R.F. Day, P.Y. Yin, Y.C. Wang, C.H. Chao, A new hybrid multi-start tabu search for finding hidden purchase decision strategies in WWW based on eye-movements, Applied Soft Computing 48, pp. 217-229, 2016.

[51] Festa P., Resende M.G.C. (2009) Hybrid GRASP Heuristics. In: Abraham A., Hassanien AE., Siarry P., Engelbrecht A. (eds) Foundations of Computational Intelligence Volume 3. Studies in Computational Intelligence, vol 203. Springer, Berlin, Heidelberg.

[52] B. Betro, F. Schoen, Optimal and sub-optimal stopping rules for the multistart algorithm in global optimization, Math. Program. **57**, pp. 445–458, 1992.

[53] W.E. Hart, Sequential stopping rules for random optimization methods with applications to multistart local search, Siam J. Optim. **9**, pp. 270–290, 1998.

[54] I.E. Lagaris and I.G. Tsoulos, Stopping Rules for Box-Constrained Stochastic Global Optimization, Applied Mathematics and Computation **197**, pp. 622-632, 2008.

[55] K. Rocki, R. Suda, An efficient GPU implementation of a multi-start TSP solver for large problem instances, In: GECCO '12: Proceedings of the 14th annual conference companion on Genetic and evolutionary computation, pp. 1441–1442, 2012.

[56] J. Larson and S.M. Wild, Asynchronously parallel optimization solver for finding multiple minima, Mathematical Programming Computation **10**, pp. 303-332, 2018.

[57] J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation **3**, pp. 246-257, 1991.

[58] S.H. Yoo, S.K. Oh, W. Pedrycz, Optimized face recognition algorithm using radial basis function neural networks and its practical applications, Neural Networks **69,** pp. 111-125, 2015.

[59] G.B. Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation, IEEE Transactions on Neural Networks **16**, pp. 57-67, 2005.

[60] Z. Majdisova, V. Skala, Radial basis function approximations: comparison and applications, Applied Mathematical Modelling **51**, pp. 728-743, 2017.

[61] B.C. Kuo, H.H. Ho, C. H. Li, C. C. Hung , J. S. Taur, A Kernel-Based Feature Selection Method for SVM With RBF Kernel for Hyperspectral Image Classification, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. **7**, pp. 317-326, 2014.

[62] H.G. Han, Q.L. Chen, J.F. Qiao, An efficient self-organizing RBF neural network for water quality prediction, Neural Networks **24**, pp. 717-725, 2011.

[63] M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.

[64] I.G. Tsoulos, Modifications of real code genetic algorithm for global optimization, Applied Mathematics and Computation **203**, pp. 598-607, 2008.

[65] I.G. Tsoulos, A. Tzallas, E. Karvounis, Improving the PSO method for global optimization problems. Evolving Systems **12**, pp. 875–883, 2021

[66] MacQueen, J.: 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.

[67] M. Montaz Ali, Charoenchai Khompatraporn, Zelda B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization **31**, pp 635-672, 2005.

[68] C.A. Floudas, P.M. Pardalos, C. Adjiman, W. Esposoto, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, C. Schweiger, Handbook of Test Problems in Local and Global Optimization, Kluwer Academic Publishers, Dordrecht, 1999.

[69] J. Zhang, V.A. Glezakou, Global optimization of chemical cluster structures: Methods, applications, and challenges, Int J Quantum Chem. **121**, 2021.