OPTIMUS: a multidimensional global optimization package

Vasileios Charilogis, Ioannis G. Tsoulos*

Department of Informatics and Telecommunications, University of Ioannina, Greece

---

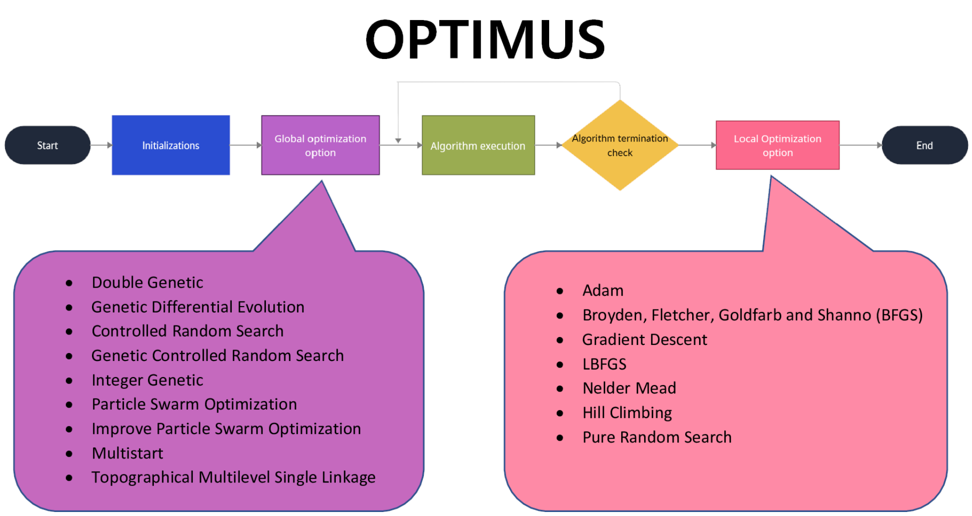
Abstract

A variety of problems from many research areas can be modeled using global optimization, such as problems 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.

Keywords: Global optimization, Termination rules, Stochastic methods

---

Graphical abstract



*Corresponding author

Email addresses: v.charilog@uoi.gr (Vasileios Charilogis), itsoulos@uoi.gr (Ioannis G. Tsoulos)

Specifications table

Subject area	Computer science
More specific subject area	Global Optimization
Method name	Optimus
Name and reference of original method	
Resource availability	`https://github.com/itsoulos/OPTIMUS/`

1. Introduction

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

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

with S:

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

Methods that aim to locate the global minimum finds application in economics [1, 2], physics [3, 4], chemistry [5, 6], medicine [7, 8] etc. Also, global optimization methods were used on some symmetry problems [9, 10, 11] as well as on inverse problems [12, 13, 14]. In the relevant literature there are a number of global optimization techniques, such as Adaptive Random Search methods [15, 16], Controlled Random Search methods [17, 18], Simulated Annealing [19, 20, 21], Genetic algorithms [22, 23], Ant Colony Optimization [24, 25], Particle Swarm Optimization [26, 27] etc. Moreover, many hybrid techniques have been proposed to tackle the global optimization problem, such as methods that combine Particle Swarm Optimization and Genetic algorithms [28, 29], methods that combine the Simplex method and Inductive search [30] etc. Because of the large demands that global optimization methods have on computing power, several techniques have been proposed that take advantage of modern parallel architectures [31, 32, 33].

In this work, an integrated computing environment is proposed for solving global optimization problems. In this the researcher can code the objective function in the C++ programming language and then 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 [34], the MERLIN optimization software [35], the DEoptim software [36] etc.

The rest of this article is organized as follows: in section 2 the software is described 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 software is entirely written in ANSI C++ using the freely available QT programming library, which can be downloaded from `https://qt.io`. The user can code the objective problem in C++ by defining some critical functions such as the dimension of the function or the objective function. 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 some new method. In the following subsections, the installation process of the proposed 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`
2. Download 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.
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*

After 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. The implemented global optimization methods are:

1. Differential Evolution. The differential evolution method is included in the software as suggested by Storn[37] and denoted as **de**. This global optimization technique has been widely used in areas such as community detection [38], structure prediction of materials [39], motor fault diagnosis [40], automatic clustering techniques [41] etc.
2. Improved Differential Evolution. The modified Differential Evolution method as suggested by Charilogis et al [42] is implemented and denoted as **gende**.
3. Double precision genetic algorithm. A modified genetic algorithm [43] 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 [44], the vehicle routing problem [45], combinatorial optimization [46], architectural design etc [47].
4. 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 [48, 49], path planning [50], Grammatical Evolution applications [51] etc.
5. Improved Controlled Random Search. An improved version of Controlled Random Search as suggested by Charilogis et al [52] is implemented and it is denoted as **CCRS**.
6. 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 [53], crystal structure prediction [54], molecular simulations [55] etc.
7. Improved Particle Swarm Optimization. The improved Particle Swarm method as suggested by Charilogis and Tsoulos [56]. The implemented method is denoted as **iPso**.
8. 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 [57], the vehicle routing problem [58], the facility location problem [59], the maximum clique problem [60], the maximum fire risk insured capital problem [61], aerodynamic shape problems [62] etc
9. Topographical Multi level single linkage. This method is proposed by Ali et al [63] and it is denoted as **Tmlsl** in the implementation.
10. The MinCenter method. In the software presented here, another multistart method has been included, which forms, with the use of the K-Means clustering algorithm, 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 [64].

2.3. Implemented local optimization methods

The proposed software uses, in addition to global optimization methods and local optimization methods, which can be used in most global minimization techniques, with 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 [65].
2. The **lbfgs** method. The limited memory BFGS method [66] 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 [67], image registration [68] etc.
4. The **adam** method. The adam local optimizer [69] is implemented also.
5. The Nelder Mead method. The Nelder - Mead simplex procedure for local optimization [70] 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 [71], load balancing in cloud computing [72] etc.

2.4. Objective problem deployment

The objective problem must be coded in the C++ programming language. The programmer must provide the software with a series of functions that describe key components of the problem, such as the problem dimension, the objective function, and the derivative.

2.4.1. Objective function coding

Figure 1 shows an objective function written with the functions required by this 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 in the optimization of the Lennard Jones potential [**?**] the user can pass the number of individuals in the potential using an assignment as follows

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

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 that represents double precision intervals eg $[2, 4]$ 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.
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 [74, 75] 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 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 command line arguments have as follows:

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

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
```

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
```

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.

3. Experiments

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. The function is defined as:

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

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

- **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

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

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**

The experiments were performed using the above objective functions and ran 30 times using a different seed for the random number generator each time. In 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 each 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 show that the usage of a local search method in combination with the genetic algorithm significantly reduces the required number of function calls and at the same time improves the reliability of the method in finding the global minimum.

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 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.
3. Implementing a GUI interface to control the optimization process.
4. Creating a scripting language to efficiently guide the optimization of objective functions.

Declaration of interests

x The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

□ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

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] 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.

[10] 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.

[11] 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.

[12] Barbara Kaltenbacher and William Rundell, The inverse problem of reconstructing reaction–diffusion systems, Invese Problems **36**, 2020.

[13] 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.

[14] 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.

[15] 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.

[16] S. Andradóttir, A.A. Prudius, A.A., Adaptive random search for continuous simulation optimization. Naval Research Logistics **57**, pp. 583-604, 2010.

[17] W.L. Price, Global optimization by controlled random search, J Optim Theory Appl **40**, pp. 333–348, 1983.

[18] 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).

[19] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.

[20] 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.

[21] L.M. Rasdi Rere, M.I. Fanany, A.M. Arymurthy, Simulated Annealing Algorithm for Deep Learning, Procedia Computer Science **72**, pp. 137-144, 2015.

[22] J. Mc Call, Genetic algorithms for modelling and optimisation, Journal of Computational and Applied Mathematics **184**, pp. 205-222, 2005.

[23] 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.

[24] 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.

[25] 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.

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

[27] N.K. Jain, U. Nangia, J. Jain, A Review of Particle Swarm Optimization. J. Inst. Eng. India Ser. B **99**, pp. 407–411, 2018.

[28] 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.

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

[30] 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. https://doi.org/10.1007/3-540-45718-6_73

[31] Y. Zhou and Y. Tan, "GPU-based parallel particle swarm optimization," In: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 1493-1500.

[32] 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.

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

[34] N.V. Sahinidis, BARON: A general purpose global optimization software package, J Glob Optim **8**, pp. 201–205, 1996.

[35] D.G. Papageorgiou, I.N. Demetropoulos, I.E. Lagaris, Computer Physics Communications **159**, pp. 70-71, 2004.

[36] 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.

[37] R. Storn, On the usage of differential evolution for function optimization, In: Proceedings of North American Fuzzy Information Processing, pp. 519-523, 1996.

[38] 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.

[39] 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.

[40] C.Y. Lee, C.H. Hung, Feature Ranking and Differential Evolution for Feature Selection in Brushless DC Motor Fault Diagnosis , Symmetry **13**, 2021.

[41] 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.

[42] V. Charilogis, I.G. Tsoulos, A. Tzallas, E. Karvounis, Modifications for the Differential Evolution Algorithm, Symmetry **14**, 447, 2022.

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

[44] 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.

[45] 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.

[46] J.F. Gonçalves, M.G.C. Resende, Biased random-key genetic algorithms for combinatorial optimization. J Heuristics **17**, pp. 487–525, 2011.

[47] 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.

[48] 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.

[49] 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.

[50] F. Ahmed, K. Deb, Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms, Soft Comput **17**, pp. 1283–1299, 2013.

[51] M. O'Neill, C. Ryan, Grammatical Evolution, IEEE Trans. Evolutionary Computation **5**, pp. 349-358, 2001.

[52] V. Charilogis, I.G. Tsoulos, A. Tzallas, N. Anastasopoulos, An Improved Controlled Random Search Method, Symmetry **13**, 1981, 2021.

[53] M. Ye, X. Wang, Y. Xu, Parameter extraction of solar cells using particle swarm optimization, Journal of Applied Physics **105**, 094502, 2009.

[54] Y. Wang, J. Lv, L. Zhu, Y. Ma, Crystal structure prediction via particle-swarm optimization, Phys. Rev. B **82**, 094116, 2010.

[55] 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.

[56] V. Charilogis, I.G. Tsoulos, Toward an Ideal Particle Swarm Optimizer for Multidimensional Functions, Information **13**, 217, 2022.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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

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

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

[64] V. Charilogis, I.G. Tsoulos, MinCentre: using clustering in global optimisation, International Journal of Computational Intelligence Studies **11**, pp. 24-35, 2022.

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

[66] D.C. Liu, J. Nocedal, On the Limited Memory Method for Large Scale Optimization, Mathematical Programming B **45**, pp. 503-528, 1989.

[67] S.I. Amari, Backpropagation and stochastic gradient descent method, Neurocomputing **5**, pp. 185-196, 1993.

[68] 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.

[69] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, ICLR (Poster), 2015.

[70] D.M. Olsson,L.S. Nelson, The Nelder-Mead Simplex Procedure for Function Minimization, Technometrics **17**, pp. 45-51, 1975.

[71] 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.

[72] 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.

[73] J.E. Lennard-Jones, On the Determination of Molecular Fields, Proc. R. Soc. Lond. A **106**, pp. 463–477, 1924.

[74] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[75] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.

[76] M.M. Ali and P. Kaelo, Improved particle swarm algorithms for global optimization, Applied Mathematics and Computation **196**, pp. 578-593, 2008.

[77] 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.

[78] Patrick Siarry, Gérard Berthiau, François Durdin, Jacques Haussy, ACM Transactions on Mathematical Software **23**, pp 209–228, 1997.

[79] I.G. Tsoulos, I.E. Lagaris, GenMin: An enhanced genetic algorithm for global optimization, Computer Physics Communications **178,** pp. 843-851, 2008.