OPTIMUS: a multidimensional global optimization package

Ioannis G. Tsoulos, Vasileios Charilogis, Alexandros Tzallas

Department of Informatics and Telecommunications, University of Ioannina,

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 $f : S \rightarrow R, S \subset R^n$ 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. In the relevant literature there are a number of global optimization techniques, such as Adaptive Random Search methods [17, 18], Controlled Random Search methods [19, 20], Simulated Annealing [21, 22, 23], Genetic algorithms [24, 25], Ant Colony Optimization [26, 27], Particle Swarm Optimization [28, 29] etc.

In this paper, a new integrated computing environment for performing global optimization methods for multidimensional functions is suggested, where the user can code the objective problem to ANSI 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. Similar software environments can be found, such as the BARON software package [30] , the MERLIN optimization software [31] , the DEoptim software [32] , the PDoublePop optimization software [33] 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 description

2.1. Distribution

The suggested software is entirely coded in ANSI C++, using the freely available QT programming library, which can be downloaded from `https://qt.io` . At the present time, the software package can only be installed on computers with the Linux operating system. 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 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*

When the compilation is complete, the *lib* folder will contain the supported global optimization methods, 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*.

2.2. Implemented optimization methods

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. The implemented global optimization methods are:

1. Differential Evolution[34], denoted as **de**.
2. Improved Differential Evolution. The modified Differential Evolution method as suggested by Charilogis et al [35] is implemented and denoted as **gende**.
3. Parallel Differential Evolution. A parallel implementation of the Differential Evolution method as suggested in [36] is considered with the name **ParallelDe**.
4. A double precision genetic algorithm [38] is included and it is denoted as **DoubleGenetic**.
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.
6. Improved Controlled Random Search [39] is denoted as **gcrs**.
7. Particle Swarm Optimization, denoted as **Pso**.
8. Improved Particle Swarm Optimization, denoted as **iPso**, an Particle Swarm method [40].
9. Multistart. A simple method that initiates local searches from different initial points is also implemented in the software.
10. Topographical Multi level single linkage [41] , denoted as **Tmlsl**.
11. The MinCenter method, denoted as **MinCenter** [42].
12. NeuralMinimizer. A novel method that incorporates Radial Basis Functions (RBF)[43] to create an estimation of the objective function introduced in [44] is implemented and denoted by the name **NeuralMinimizer**.

Also, the parameter used to determine the used local optimization procedure is the −−localsearch_method parameter and the implemented methods are:

1. The **bfgs** method, denoting The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [45].
2. The **lbfgs** method. The limited memory BFGS method [46].
3. The Gradient descent method, denoted as **gradient**.
4. The **adam** method. The adam local optimizer [47] is implemented also.
5. Hill climbing. The hill climbing local search procedure denoted as **hill** is also implemented.

2.3. Objective problem deployment

The objective problem must be coded in the C++ programming language. Figure 1 shows an example of objective function for the Rastrigin function. The implemented are as follows:

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.
2. **int** getDimension(), 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.
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 $f(x)$.

2.3.1. 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*

2.3.2. 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 −−filename determines the objective problem in shared library format.
2. The argument of −−opt_method sets the used global optimization procedure.
3. The argument of −−pso_particles sets the number of particles of the PSO optimizer.
4. The argument of −−pso_generations sets the maximum number of generations allowed.
5. The argument of −−localsearch_method sets the used local optimization procedure.

The output of the previous command is shown in figure 3.

3. Experiments

To assess the ability of the software package to adapt to different problems, a series of experiments were performed under different conditions and they are analyzed in the following subsections.

3.1. Test functions

To measure the effect of the proposed software, a series of experiments were performed on test functions from the relevant literature [48, 49, 50, 51]. The experiments were performed 30 times for every test case using a different seed for the random number generator each time. Two variations of the genetic algorithm (DoubleGenetic method) were used: 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 and the results are shown in Table 2. The numbers in parentheses show the percentage of finding the global minimum, where the absence of this number denotes that the algorithm discovered the global minimum in all executions. 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.

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 [52, 53] is used as a test case here. The function to be minimized is given by:

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

For testing purposes, the methods **NeuralMinimizer**, **DoubleGenetic** and **Pso** was applied for a variety of numbers of atoms and the results are shown in Table 3. 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.

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 4 and they indicate that the number of calls required to find the global minimum decreases as the total processing threads increases, even though the problem becomes increasingly difficult with increasing dimensions.

4. Conclusions

In this work, an environment for executing global optimization problems was presented, where 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. This programming environment is freely available and easy to extend to accommodate more global optimization techniques and it can be extended in the near future with the following improvements:

1. Port the Optimums tool to other operating systems.
2. Use of modern parallel techniques to speed up the generated results.
3. Implementing a GUI interface to control the optimization process.
4. The 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.

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] 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.

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

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

[20] 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).

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

[22] 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.

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

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

[25] 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.

[26] 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.

[27] 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.

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

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

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

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

[32] 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.

[33] 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.

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

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

[36] V. Charilogis, I.G. Tsoulos, A Parallel Implementation of the Differential Evolution Method, Analytics **2**, pp. 17-30, 2023.

[37] R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.

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

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

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

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

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

[43] J. Park, I.W. Sandberg, Approximation and Radial-Basis-Function Networks, Neural Computation **5**, pp. 305-316, 1993.

[44] I.G. Tsoulos, A. Tzallas, E. Karvounis, D. Tsalikakis, NeuralMinimizer, a novel method for global optimization that incorporates machine learning, Information **14**, 2, 2023.

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

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

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

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

[49] 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.

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

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

[52] J.A. Northby, Structure and binding of Lennard-Jones clusters: $13 \leq n \leq 147$, J. Chem. Phys. **87**, pp. 6166–6178, 1987.

[53] 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.

[54] O.A. Sauer, D.M. Shepard, T.R. Mackie, Application of constrained optimization to radiotherapy planning, Medical Physics **26**, pp. 2359-2366, 1999.

[55] F.P. Seelos, R.E. Arvidson, Bounded variable least squares - application of a constrained optimization algorithm to the analysis of TES Emissivity Spectra, in: 34th Annual Lunar and Planetary Science Conference, March 17-21, 2003, League City, Texas, abstract no.1817.

[56] M.J. Field, Constrained optimization of ab initio and semiempirical Hartree-Fock wave functions using direct minimization or simulated annealing , Journal of physical chemistry **95**, pp. 5104-5108, 1991.

[57] G.A. Williams, J.M. Dugan, R.B. Altman, Constrained global optimization for estimating molecular structure from atomic distances, Journal of Computational Biology **8**, pp. 523-547, 2001.

[58] P.E. Gill, W. Murray, The computation of Lagrange-multiplier estimates for constrained minimization, Mathematical Programming **17**, pp. 32-60, 1979.

[59] M.J.D. Powell, Y. Yuan, A trust region algorithm for equality constrained optimization, Mathematical Programming **49**, pp. 189-211, 2005.

[60] K. Ichida, Constrained optimization using interval analysis, Computers and Industrial Engineering **31**, pp. 933-937, 1996.

[61] R.L. Becerra, C.A.C. Coello, Cultured differential evolution for constrained optimization, Computer Methods in Applied Mechanics and Engineering **195**, pp. 4303-4322, 2006.

[62] A.V. Levy, A. Montalvo, The tunneling algorithm for global optimization of functions, SIAM Journal of Scientific and Statistical Computing **6**, pp. 15-29, 1985.

[63] H.M. Salkin, Integer programming, Edison Wesley Publishing Com., Amsterdam, 1975.

[64] R. Hess, A heuristic search for estimating a global solution of non convex programming problems, Operations Research **21**, pp. 1267-1280, 1973.

[65] P. Chootinan, A. Chen, Constraint handling in genetic algorithms using a gradient - based repair method, Computer and Operations Research **33**, pp. 2263-2281, 2006.

[66] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A.C. Coello, K. Deb, Problem definitions and evaluation criteria for the CEC2006 special session on constrained real-parameter optimization, `http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC-06/CEC06.htm`

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 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
```

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**

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)**

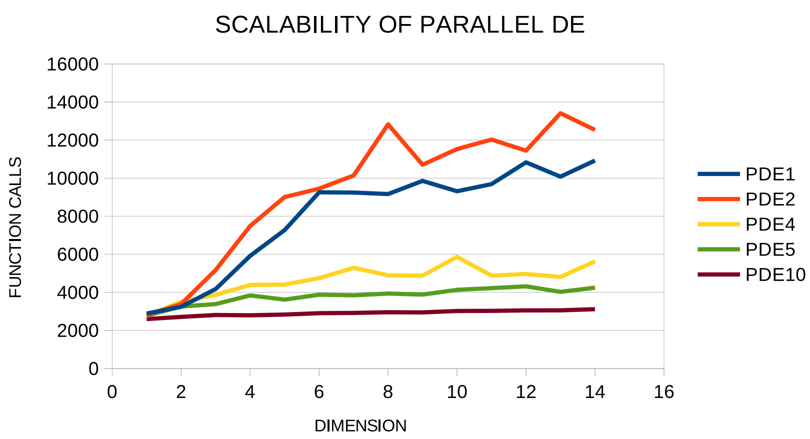

Figure 4: Scalabilty of the ParallelDe method.

Required Metadata

Current code version

Nr.	**Code metadata description**	
C1	Current code version	1.0
C2	Permanent link to code/repository used for this code version	`https://github.com/itsoulos/OPTIMUS/`
C3	Legal Code License	GNU General Public License (GPL)
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	C++
C6	Compilation requirements, operating environments & dependencies	Linux , QT Library
C7	If available Link to developer documentation/manual	`https://raw.githack.com/itsoulos/OPTIMUS/master/MANUAL/docs/html/index.html`
C8	Support email for questions	itsoulos@uoi.gr