2020 1st International Conference Problems of Informatics, Electronics, and Radio Engineering (PIERE) | 978-1-7281-8990-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/PIERE51041.2020.9314637

Development of the Library for Solving Scalar and Vector Optimization Problems with Stochastic Methods

Mikhail G. Grif, Pavel A. Zhurkin
Novosibirsk State Technical University, Novosibirsk, Russia

***Abstract* – different approaches to developing software for solving a wide range of optimization problems are discussed. Applying the particle swarm method and genetic algorithm is proposed. The rationale for stochastic methods is given. A brief description of methods for solving vector optimization problems is given the generalized criterion method and the compromise method being presented. The paper suggests approaches to increasing such universal features of software being developed as cross-platform features and compatibility with several programming languages. The results of the study of the implemented optimization methods effectiveness are presented, a comparative analysis is carried out, and conclusions are drawn on the applicability of the implemented methods.**

***Index Terms*– optimization methods, scalar and vector optimization problems, particle swarm method, genetic algorithm, translation methods.**

I. INTRODUCTION

CURRENTLY, such areas of information technology as artificial intelligence and machine learning are being actively developed. A large number of different software products appear in these areas. Such products aim to solve various kinds of optimization problems.

Modern approaches to software development involve the inclusion to the project and the use of outsourced components, not directly related to the business logic of the application. Thus, developers have a need for specialized software tools that allow them to solve optimization problems.

Most of the currently existing software products allow solving a fairly narrow class of optimization problems. Some software packages do not allow convenient integration into the outsourced software.

For example, the mathematical package Matlab, which allows solving a large number of different mathematical problems, including optimization problems, provides the ability to use its functionality to other programs through a COM object [1]. This means that the end user will need to supply the mathematical package itself, which is very cumbersome and very expensive, with the program that accesses the Matlab functions. Such a choice is far from efficient.

As already noted, many software tools, designed to solve optimization problems, provide the opportunity to solve a narrow range of optimization problems. This feature can be seen in libraries that implement specific optimization methods. For example, the libraries PySwarms [2] and GALib [3], which implement the particle swarm method and the genetic algorithm, do not allow solving vector optimization problems. PySwarms library does not allow setting restrictions for a task.

II. PROBLEM STATEMENT

The study is devoted to the development of a software library that allows solving a wide range of optimization problems using stochastic methods – a particle swarm method and a genetic algorithm.

The following requirements are imposed on the developed product:

1. Solutions to both scalar and vector optimization problems;
2. The statement of the optimization problem in the form of program code inside the program using the developed library;
3. Statement of the optimization problem without the use of programming tools, in a special language for setting optimization problems, using mathematical expressions.
4. The solution of optimization problems of any dimension;
5. The ability to impose restrictions of the following types on the problem being solved: the variable change interval, the integer and continuous types of some variables, arbitrary restrictions represented by inequalities;
6. Support for multiple operating systems - Windows, Linux;
7. The ability to connect the library to programs in different programming languages.

III. THEORY

As the optimization methods that the library offers, stochastic methods, i.e. the particle swarm method and the

978-1-7281-8990-1/20/$31.00 ©2020 IEEE

genetic algorithm [4] were chosen for implementation. These methods are selected for two reasons:

1. When solving the optimization problem by stochastic methods, the objective functions can be of any kind, e.g. linear functions, nonlinear functions, and functions that do not have an explicit analytical representation and are specified algorithmically using conditional and loop operators;

2. Unlike deterministic methods, stochastic methods allow finding a solution in an acceptable time even with a drastic increase in the dimension of the problem.

Methods that allow solving vector optimization problems, use an approach, where the solution of a multicriteria problem is reduced to solving one or several scalar optimization problems. In this project, the methods of generalized criterion and the method of assignment were chosen for implementation.

When using the generalized criterion method, it is necessary to obtain additional information about the criteria from the decision maker regarding the significance of each of the criteria relative to each other [5]. This information is presented as a set of positive numbers α_i. The number α_i characterizes the importance of the i-th criterion relative to others. In some cases, there might be a requirement of the form

$$\sum_i \alpha_i = 1. \quad (1)$$

The generalized criterion F has the form:

$$F(\bar{x}) = \sum_i(\alpha_i * f_i(\bar{x})), \quad (2)$$

where

$f_i(\underline{x})$ – i-th optimality criterion.

The condition for applicability of the linear convolution method is the homogeneity of the criteria: the ranges of the values of all objective functions must coincide, having the same order, the reference points must also be the same [6].

In a situation where this condition is not fulfilled, criteria should be normalized before applying the method. Normalization is performed by the following conversion:

$$\omega_i(\underline{x}) = \left[\frac{f_i(\underline{x}) - f_i^{min}}{f_i^{max} - f_i^{min}}\right]^{\mu}, \quad (3)$$

where

$\omega_i(\underline{x})$ – normalized i-th criterion,

f_i^{max} – global maximum of function in domain,

f_i^{min} – global minimum of function in domain,

$\mu = 1, 2\ldots$

As a result, after normalization, a generalized criterion of the following form is obtained:

$$F(\bar{x}) = \sum_i(\alpha_i * \omega_i(\bar{x})). \quad (4)$$

When using the method of assignment, the decision maker provides additional information on the preference of the optimality criteria (the criteria are sorted in descending order of importance), as well as the value that can be assigned according to each criterion to maximize/minimize the remaining criteria [6].

When solving a vector problem with N criteria by the assignment method, N scalar optimization problems are constructed; one for each of the criteria. Then, each task is successively solved in the established order of decreasing importance of the criteria.

The first task is solved without restrictions, the optimum is sought for the first criterion. In solving each of the following problems, one restriction is introduced, the meaning of restriction is that the value of the previous criterion should not deviate from the optimum found by a value bigger than the specified value of the assignment.

IV. DEVELOPMENT RESULTS

A. Selection of development tools

The library was developed in the C ++ programming language, the language was selected for the following reasons:

1. The language allows generating cross-platform code, which makes it possible to compile and run programs on various operating systems;

2. The language is native, which means that programs are compiled, not interpreted, and compiling is carried out in machine code executed directly by the processor under the operating system control, and not by any virtual machine. This allows implementing support to connect the library to programs in several other programming languages, which increases the versatility of the developed solution;

3. The language is object-oriented, which provides the ability to solve a wide range of optimization problems.

B. Implementation of the Algorithmic Module

Two facts influenced the features of the software implementation of the algorithmic module:

1. For stochastic methods, the type of the objective function is not important, only the ability to evaluate the value of the function at some point is important;

2. Methods for solving vector optimization problems involve reducing the solution to one or several scalar problems.

All this allows implementing optimization methods, abstracting from the type of problem and its details. It is enough to implement methods that can solve the scalar optimization problem.

The system of classes representing the solved optimization problems in the program is based on the mechanism of polymorphism [7]. The library has a basic abstract class, or an interface that represents a scalar optimization problem. The interface describes a set of methods allowing to obtain information of the following kind about the optimization problem to be solved:

1. A method for calculating the value of the objective function at a point;

2. A method to check whether a certain point satisfies the system of constraints of the problem;

3. A number of methods to obtain information about the variables of the problem.

All optimization tasks that need to be solved are presented in the program in the form of classes that inherit the interface and implement interface methods. After that, objects of such classes can be created and transferred to the input of the optimization algorithm.

Fig. 1 shows the UML diagram of classes representing optimization tasks in the library.

The ICreator class is an interface that represents optimization tasks. ScalarProblem and VectorProblem are classes of scalar and vector optimization problems respectively. The objects of these classes are fed to the input of optimization algorithms.

CustomProblem is a class that can be created by a developer using the library in his program.

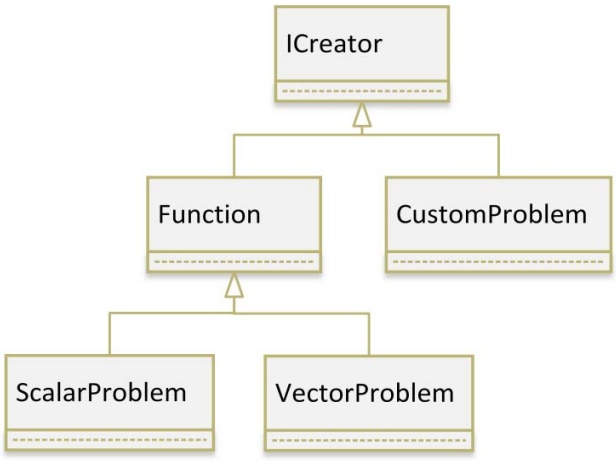

Fig. 1. A task class diagram.

This class represents a specific task that a developer needs to solve in his application. A class can be inherited both from the ICreator interface itself, and from one of the existing library classes, if they are suitable for the needs of the developer.

It should be explained how the possibility of solving vector problems is achieved using optimization methods implemented to solve scalar problems.

The library implements a separate class VectorProblem, which represents a vector optimization problem. An object of this class stores a list of all the criteria for optimality of the problem. However, this class inherits the interface of the scalar optimization problem. But its implementation is based on the method of generalized criterion described earlier. The method of this class, which allows obtaining the value of the objective function, calculates the values of all the criteria, and then calculates and returns their weighted sum, which is a generalized criterion.

In addition, the library has a separate class that implements the assignment method. When constructing, objects of this class receive an input object-task, a list of criteria, sorted in descending order of importance, value of assignment for each of the criteria. The method automatically builds scalar task objects for each criterion and restrictions for them, and also automatically sends these objects to the input of the optimization algorithm.

C. Language for setting optimization problems

As a part of this work, a language for setting optimization problems was developed. The language operates with mathematical expressions, describing the criteria of optimality and restrictions in the form of inequalities. This language makes it possible to formulate an optimization problem without resorting to programming languages and tools.

In order for the tasks set with the help of the task setting language to be solved by the developed algorithmic module, a translator for this language was created.

The task of the translator is to convert the text setting of the problem received at the input into an object of one of the ScalarProblem or VectorProblem classes shown in Fig. 1. Since the implemented optimization algorithms - a swarm of particles and a genetic algorithm - support the solution of problems represented by any classes that inherit the ICreator interface, the presence of such a translator will allow solving problems described in a textual analytical form.

An example of an analytical statement of the problem is shown in Fig. 2.

```
f1(x1, x2) = 3*x1 - x2 -> max,
f2(x1, x2) = x2 -> max

where
2*x1 + x2 <= 6,
x1 >= 0,
x1 <= 2,
x2 >= 0,
x2 <= 4
```

Fig. 2. An example of a text statement of the optimization problem.

The text of the statement of the optimization problem consists of two blocks: a criteria description block and a constraint description block.

The criteria description block consists of a sequence of objective functions, each having a name, a list of variables, a mathematical expression for calculating the criterion value, and the direction of optimization.

The constraint description block begins with the **where** keyword, followed by a set of constraints represented by inequalities. On the left and right sides of the inequalities can be arbitrary mathematical expressions.

The mathematical expressions in the description of the criteria and in the description of restrictions can contain numerical constants, task variables, mathematical functions (sin, cos, log, exp, and so on) as operands.

The translator for the language of statement of optimization problems was developed using the theory of formal languages and compilers [8]. For the language, the

vocabulary was formulated, presented in the form of a set of regular expressions, the syntax was developed, presented in the form of a context-free grammar related to the class LL (1) [9].

Lexical and syntax parsers were built automatically using flex and bison generators [10].

The structure of the translator is shown in Fig. 3.

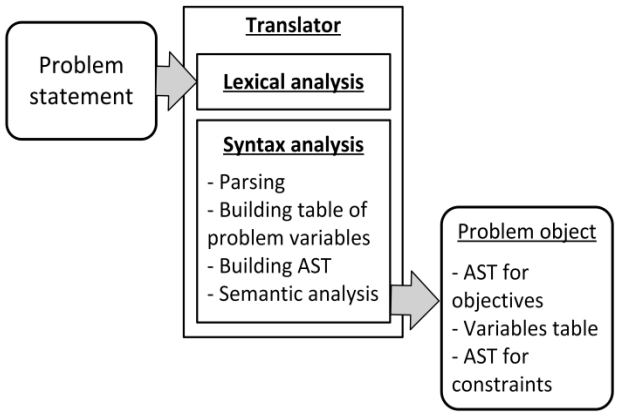

Fig. 3. Translator structure.

The text setting of the problem is fed to the input of the translator, and at the output the translator builds abstract syntax trees (AST) for evaluating objective functions, ASTs for evaluating constraints and a table of task variables. These artifacts are combined inside an instance of a class that is inherited from the ICreator interface. The object constructed by the translator can be fed to the input of the optimization algorithm. Thus, it is possible to solve optimization problems posed in an analytical form.

D. Support of integration with other programming languages

Using a native programming language in the development of libraries allows connecting these libraries, after compilation, to programs in other programming languages. To make this possible, the following approach was used.

For the required programming language, a new adapter library is implemented. This adapter library contains analogues of the interface classes of the connected library - wrapper classes. Wrapper classes have methods similar to those of classes wrapped by them.

When creating an object of a wrapper class, an object of a wrapped class is automatically created as its field. When invoking methods of the wrapper class, the passed parameters are converted and similar methods of the object of the wrapped class are called.

In the framework of this work, an adapter library for the C# programming language was developed. This made it possible to verify the practical suitability of the developed library for connecting to programs in other languages, as well as to ensure real coverage of two programming languages in which the developed solution can be used.

When developing the adapter for the C# language, the C++/CLI language was used [11].

C++/CLI is a programming language from Microsoft, which is an extension of the C++ language. It makes it possible to use data types from the .NET environment in programs, in addition, support is provided for interacting with the so-called native code, a binary code, that is not run under the control of the .NET environment. These features of the C++/CLI technology allow it to organize the interaction of code written in "pure" C++ with code written for the .NET platform, in particular, in the C# programming language. In order for C# code to be able to call code from a C++ library, it is necessary to create analogs of all library classes in the adapter using the approach described above.

V. DISCUSSION OF RESULTS

To test the time and accuracy of solving the optimization problem by the particle swarm method and the genetic algorithm, a test optimization problem was implemented in software form, the objective function of which is as follows:

$$f(\bar{x}) = \sum_{i=1}^{v}(x_i - 4.65)^2 + 100 \rightarrow min, \quad (5)$$

where v – the number of task variables.

The following parameters of the particle swarm method were used: 10 particles, 50 thousand iterations. The parameters of the genetic algorithm are: 5 individuals, 2 million iterations.

The first testing was carried out in order to identify the dependence of the running time of the algorithm on the number of task variables. The test results are shown in Table I.

TABLE I
TIME TEST RESULTS

Variables number	Particle swarm	Genetic algorithm
	time, seconds	time, seconds
10	7,50	10,60
50	25,40	30,19
100	51,20	58,90
200	109,40	91,30

Graphs of the dependence of the operating time of each of the algorithms on the dimension of the problem are shown in Fig. 4.

The second test was conducted in order to identify the error that each of the algorithms gives, depending on the dimension of the problem. Table II shows the relative errors in percent obtained during the operation of the algorithms. The error is calculated by the formula:

$$d = \frac{|f^* - f|}{f^*} * 100, \quad (6)$$

where

d – percentage calculation error,

f* - true function minimum,

f – the minimum found by the algorithm.

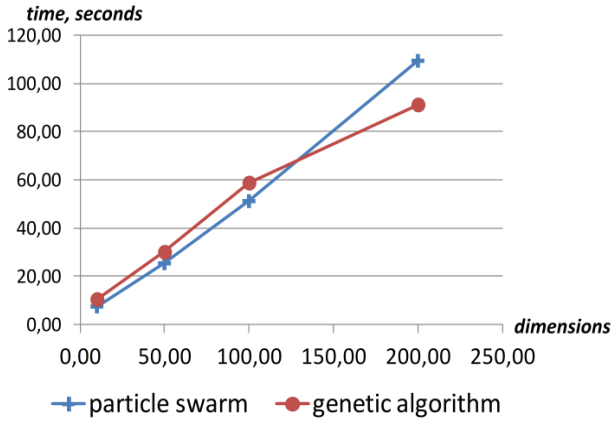

Fig. 4. Dependencies of working time on the dimension of the task.

TABLE II
ERROR TEST RESULTS

Variables number	Particle swarm	Genetic algorithm
	error, %	error, %
10	0	$1,7 * 10^{-6}$
50	$2,1 * 10^{-13}$	0,0007
100	0,01	0,2
200	107,8	302,3

Tables I, II as well as Fig. 4 show that having the given parameters of the optimization algorithms, solutions similar in time and accuracy were obtained, but in general the particle swarm method proved to be better: all solutions turned out to be more accurate, and in most cases the operating time was shorter.

Nevertheless, the parameters of the particle swarm method can be changed as follows: to make the number of particles larger and reduce the number of iterations. This will allow us to find a more accurate solution, while reducing the computation time. An experiment was done with the following parameters: 1000 particles and 1000 iterations. The result of the experiment is shown in table III.

Fig. 5 shows graphs of the dependence of the running time of the algorithm on the number of variables. The graph compares the experimental results for the genetic algorithm with the initial parameters (the data from table I were used to construct the graph) and the experimental results for the particle swarm method with new parameters (the graph is based on the data from table III).

TABLE III
Particle Swarm Method Results with Parameters: 1000 Particles and 1000 Iterations

Variables number	time, seconds	error, %
10	1,8	0
50	6,1	$4,3 * 10^{-14}$
100	10,8	$1,3 * 10^{-7}$
200	17,7	4,4

The results show that in a situation where the number of task variables is small, a genetic algorithm may be applicable. However, with the right selection of parameters, the particle swarm method is much more efficient and accurate.

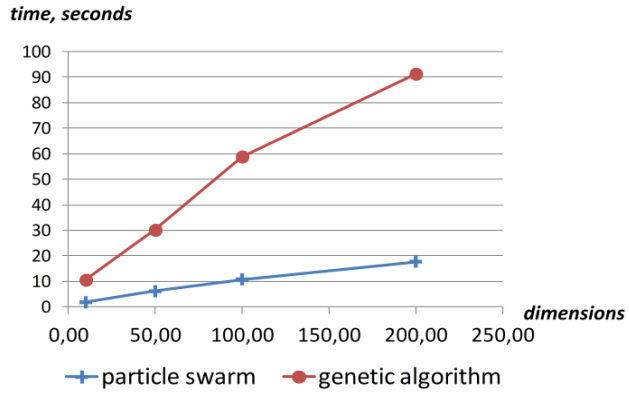

Fig. 5. The running time of the genetic algorithm with initial parameters and the particle swarm method with new parameters.

Increasing the number of particles allows us to cover more potential points close to the optimum, and also it is possible to reduce the number of iterations.

In the genetic algorithm, a small change in population size does not lead to a significant change in accuracy, but with an increase in the number of individuals in large tasks, the accuracy of the solution decreases sharply. The error can be reduced by increasing the number of iterations, which leads to an increase in the operating time of the algorithm, which, in comparison with the particle swarm method, is already quite large.

VI. CONCLUSIONS

In this work, a software library for solving a wide range of optimization problems using stochastic methods was developed.

The particle swarm method and the genetic algorithm were chosen as optimization algorithms for implementation.

Using stochastic methods made it possible to solve optimization problems in which objective functions can be of any kind. Methods for solving multicriteria optimization problems were developed. The necessity of solving problems of arbitrary dimensions is taken into account and the possibility of solving optimization problems with restrictions is provided.

The library code is written in a cross-platform style, which allows compiling and using this software product in the operating system Windows and Linux.

The possibility of using the library in programs in various programming languages is provided. This possibility is confirmed in practice as a result of the development of an adapter for the C# programming language. Thus, at the moment the library supports two programming languages: C++ and C#.

The possibilities described above, i.e. a wide range of optimization problems to be solved, cross-platform support

for several programming languages, allow us to consider that the developed solution has a high degree of versatility.

A research of the operating time and accuracy was made for optimization methods implemented in this software library - the particle swarm method and the genetic algorithm. A comparative analysis of the results obtained during testing for each of the methods allows us to conclude that the particle swarm method is more preferable, since with the right selection of parameters it allows you to get a more accurate solution in a less time. The genetic algorithm is applicable to problems of small dimensions.

The quality of the current implementation of optimization methods and the possibility of increasing their effectiveness is for further study.

REFERENCES

[1] Call MATLAB Function from C# Client. [Online]. Available: https://www.mathworks.com/help/matlab/matlab_external/call-matlab-function-from-c-client.html. Accessed: Apr. 3, 2020.

[2] PySwarms documentation. [Online]. Available: https://pyswarms.readthedocs.io/en/latest/index.html (accessed: Mar. 3, 2020).

[3] GAlib. A C++ Library of Genetic Algorithm Components. [Online]. Available: http://lancet.mit.edu/ga/ (accessed: Apr. 03, 2020).

[4] Simon D., "Evolutionary Optimization Algorithms", John Wiley & Sons, 2013.

[5] Caramia, Massimiliano, and Paolo Dell'Olmo, "Multi-objective optimization," *Multi-objective Management in Freight Logistics*, Springer, Cham, 2020, pp. 21–51.

[6] O. V. Kazanskaya, S. G. Yun, O. K. Alsova, "Models and methods of linear and vector optimization: tutorial," Novosibirsk, Russia: NSTU Publishing house, 2007, 192 p. (in Russian)

[7] H. Schildt, *C++ from the Ground Up*, 3d ed., Berkeley, California, U.S.A.: McGraw-Hill/Osborne, 2018.

[8] A. V. Aho, M. S. Lam, R. Sethi, J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, 2d ed., 2007.

[9] S. Sippu, E. Soisalon-Soininen, Parsing Theory: Volume II LR (k) and LL (k) Parsing, Springer Science & Business Media, 2013, vol. 20.

[10] J. Levine, *Flex & Bison*: Text Processing Tools, O'Reilly Media Inc., 2009.

[11] G. Hogenson, *C++/Cli The Visual C++ Language for. Net*, Dreamtech Press, 2007.

Mikhail Grif, professor of the Automated Control System Department of Novosibirsk State Technical University. Research interests: design of complex systems, computer sign language interpreter systems. Published about 300 scientific papers.

Pavel Zhurkin, second year master student of the Automated Control System Department of Novosibirsk State Technical University. Research interests: system software. Published 5 scientific papers.