On Methods of Controlling Optimization Software Packages with the Application of Parallel Computing

Kamil Aidazade[1], Samir Guliyev[2]
[1]Baku State University, Baku, Azerbaijan
[2]Institute of Control Systems, Baku, Azerbaijan
[1]*kamil_aydazade@rambler.ru*, [2]*copal@box.az*

***Abstract* — The paper is dedicated to the analysis of methods and algorithms of controlling computational process of solving complex problems with the use of multiprocessor and/or multicore computer systems. We have developed an automatic and dialogue systems of control of unconstrained optimization process, which have a graphical user interface.**

Keywords — optimization methods, parallel computations, multiprocessor and multicore systems, dialog systems

I. INTRODUCTION

It is known that in spite of a large number of methods for numerical solution to various classes of problems, the choice of the most efficient method for solving a particular problem under specific values of its parameters requires a large number of comparative experiments. As a rule, the end users tend to have difficulty both in carrying out such experiments, which requires the knowledge of domain of applicability of various numerical methods, and in proper conducting of the comparative analysis of the results, which is time consuming.

In the paper, for the class of problems of multivariate unconstrained optimization, we propose two approaches for facilitating the use of available applied software packages using modern multi-processor (multi-core) computer systems. One of the approaches involves active work of the user with the optimization program package in a dialogue mode. The other approach involves the packet control by means of a specially developed control program in automatic mode.

Let's note that beginning from the 70s of the last century, in different schools, they have been carrying out works on creating intellectual algorithms for controlling software packages. For example, Yu.G.Evtushenko's school have developed controlling dialog systems of unconstrained optimization ("DISO") and of optimal control ("DISOPT") [1,2]. In the works of one of the authors of the present paper and his colleagues, they developed controlling systems of solving vector optimization problems ("DIVO") and global optimization problems ("GLOPT") [3,4]. These systems are constantly improved and modified in connection with the development of information, computer, and software technologies. The results of the given paper are direct development of ideas employed in the above-mentioned systems, taking into account new possibilities provided by modern computing technologies.

II. PROBLEM STATEMENT

Let $P=\{p_i(x): i\in N\}$ be the class of optimization problems (tasks). Here N is a given set defining individual problems of the class; $x\in D_i\subset R^n$ are the arguments of each individual problem, which can take on values from some given admissible set D_i, defined by each specific optimization problem individually. It is assumed that for every problem $p_i(x)$ there exists a goal subset of extrema $D_i^*\subset D_i$ such that $D_i^*\neq\varnothing$. The problem $p_i(x)$ consists in finding at least one point $x^*\in D_i^*$. The set D_i^* is called a set of solutions to the problem $p_i(x)$.

To solve all the problems of the class P, there is usually a corresponding family of methods $M=\{M_j: j\in J\}$, each of which solves the problems $p_i(x)$ of the given class, i.e. they find a point $x^*\in D_i$. Moreover, each method $M_j, j\in J$, has different efficiency (in terms of time used, the accuracy of the solution, etc.) when solving the problem $p_i(x)$.

As the optimization techniques we use methods of direct search (zero-order methods), gradient-based methods (first-order methods), and Newton-type methods (second-order methods) [5-8]. These methods have a large number of options settings, thus providing the ability to adapt the system to any process quickly. Furthermore, the combination of direct search methods, gradient-based methods, and Newton-type methods allows us to find an optimal solution for a smaller number of steps and/or calculations of the objective function, which is important in terms of the cost of optimization process.

The report sets out the possible principles of management of optimization software package when solving a particular applied problem $p_i(x)\in P$, allowing us to increase the overall efficiency of solving the problem by combining the optimization methods in the process of solving the problem with the use of a multiprocessor (multicore) computer system.

A. The principle of sequential implementation on a single core (SISD) architecture

This principle is of important significance in its own right, and can be considered as the basic unit for the implementation on multiprocessor or multicore architectures. Let us describe

DOI: 10.25045/NCSoftEng.2017.04

one of the principles of the possible schemes of implementation of the algorithm for solving optimization problems on such architectures.

Let $M_1, M_2, ..., M_k$ be a list of optimization methods, composed of algorithms in the software package of unconstrained optimization. It is reasonable to include in the list diverse methods, if the structure of the objective function is, generally speaking, not known.

The process of solving the problem is carried on in stages, each of which consists of training and working steps. The first of these steps is intended to identify the locally efficient algorithm from the available list of algorithms. After that, the working step is carried out, which consists in solving the problem using only the algorithm that has proven to be the most efficient in the first step. Both the training and working steps are carried out within a certain time slice. One can use two variants of the training step:

1. To determine the local efficiency of the methods, the optimization process starts from the same point x^0. In this case there is somewhat wasteful consumption of machine time, and the training step is only used to identify a locally efficient algorithm;

2. The training step is used not only to find an efficient algorithm, but also to advance to an extreme point, because instead of the original point we use the current point to train each of the following algorithm.

At the training step all the algorithms of the initial list $M_1, M_2, ..., M_k$ have the opportunity to work within the given initial time slice, with the exception of only those methods that have been the least efficient for two consecutive training steps. These methods are not allocated any time slice and are temporarily excluded from the list.

To calculate the values of the local efficiencies of the methods, we make use of the following formula:

$$E_i = | f(x^{k+1}) - f(x^k) | / (| f(x^k) | + \varepsilon) + \\ + ||x^{k+1} - x^k|| / (||x^k|| + \varepsilon).$$

Here E_i is the local efficiency of the i^{th} algorithm; x^{k+1}, x^k are the final and initial points obtained using the i^{th} algorithm; $f(x^{k+1}), f(x^k)$ are the values of the objective function at these points; $||.||$ is the Euclidean norm; ε is a small positive number.

The cycle criterion of the proposed procedure is the fulfillment of the exit criteria for all methods. In conclusion, the user receives the accumulated information on the search process, which includes the optimal chain of methods that worked at the working steps; the total time of search for solutions; the values of the objective function, of the coordinates, and of local efficiencies of the methods obtained during the training step.

B. The principle of parallel implementation on a multicore architecture

The simplest implementation of a multi-threaded version of the solution to the given optimization problem seems to be an approach that involves several threads independently performing operations of the sequential algorithm described above.

The solution to an unconstrained optimization problem is carried out in stages. At each stage, the following steps are implemented:

1. At the initial step, from the list of all available algorithms $M_1, M_2, ..., M_k$ of unconstrained optimization, we randomly select several algorithms $M_{s_1}, M_{s_2}, ..., M_{s_N}$, the number N of which is chosen equal to the number of cores present on the computer system.

2. At the working step, we identify the most efficient algorithms. The duration T_i of the working step may increase if any method has proven to be the most efficient for several consecutive stages.

3. The current values of the local efficiencies E_i of the methods are calculated. From the list of working algorithms, we exclude a half of those who have exhibited the lowest efficiency.

4. To the list of working algorithms we then add as many other algorithms as were excluded in the previous step, and repeat steps 2 through 4 again.

When working with the automatic and dialogue systems, the user, in accordance with the standard requirements, formulates an optimization problem in any programming language in the form of a module (dynamic link library), and then enters it into the system by specifying the full path to the created library file; using the directives (instructions), the user runs the most appropriate (in his/her opinion) algorithms of the library of modules, and tunes their various settings. The control program will then organize the interaction of the modules from the package; manages the input of the initial and current information; interpret the user's directives (instructions); load optimization modules into the computer memory dynamically; output the results of computations on the display (at the same time you can get results on a printer) in a prescribed form.

Analyzing the results of the computations, the user decides on the further calculations, thus obtaining the possibility to monitor the progress of solving the problem, to intervene promptly in the computation process, to choose the working methods, and to adjust, if necessary, their parameters. The user determines how often and in what form the results should be displayed on the screen, and then, using a predefined set of directives carries out calculations.

The report will contain the protocols and results of computer-based experiments for the class of unconstrained optimization problems using different principles of management of the developed software package.

DOI: 10.25045/NCSoftEng.2017.04

CONSLUSION

In the paper, we proposed an approach to control of computational process of solving complex applied problems by an example of multivariate unconstrained optimization problems using appropriate software packages on multi-processor (multi-core) computer systems. The proposed approaches essentially facilitate the end-users’ work of using existing standard software packages. They require a different level of users’ knowledge of methods implemented in the software packages.

REFERENCES

[1] Yu.G.Evtushenko, Methods of solving extreme problems and their application in optimization systems, Moscow, Nauka, 1982. (in Russian)

[2] Yu.G.Evtushenko and V.P.Mazurik, Optimization systems software, Moscow, Znanie, 1989. (in Russian)

[3] K.R.Aidazade and N.S.Sidorenko, “An approach to the construction of combined optimization algorithms”, Technical Cybernetics, 1982, Issue 6, pp.87-93. (in Russian)

[4] K.R.Aidazade and I.G.Novruzbekov, “Dialog system of multicriteria optimization”, Proceedings of the Academy of Sciences of Azerbaijan SSR, series of Physical-Technical and Mathematical Sciences, Issue 2, 1987.

[5] F.P.Vasilyev, Optimization methods, vol. 1 and 2., MTsNMO, 2011. (in Russian)

[6] B.T.Polyak, Introduction to optimization, Lenand, 2014. (in Russian)

[7] J.Nocedal and S.Wright, Numerical optimization, Springer Series in Operations Research and Financial Engineering, 2nd edition, 2006.

[8] R.Baldick, Applied optimization: formulation and algorithms for engineering systems, Cambridge University Press, 2009.

DOI: 10.25045/NCSoftEng.2017.04