ON METHODS OF MANAGING OPTIMIZATION SOFTWARE PACKAGES WITH THE APPLICATION OF PARALLEL COMPUTATIONS

S.Z. GULIYEV,
Institute of Control Systems of ANAS,
Baku, Azerbaijan
azcopal@gmail.com

Abstract. *The paper conducts the analysis of methods and algorithms of managing computational process of solving complex optimization problems using multiprocessor and/or multicore computer systems. We have developed an automatic and dialogue systems of unconstrained optimization with a graphical user interface. The developed systems are equipped with an extensive library of optimization algorithms.*
Keywords: *optimization methods, parallel computations, multiprocessor and multicore systems, dialog systems.*

Let $P=\{p_i(x): i \in N\}$ is the class of optimization problems (tasks), where N is a given set that defines the individual problems of the class, $x \in D_i \subset \mathbb{R}^n$ are the arguments of each individual problem that can take values from some given admissible set D_i, determined by each specific optimization problem individually. It is assumed that for each problem $p_i(x)$ there is some target subset of extrema $D_i^* \subset D_i$, and $D_i^* \neq \varnothing$. Problem $p_i(x)$ consists in finding at least one point $x^* \in D_i^*$. The set D_i^* is called the set of solutions to the problem $p_i(x)$.

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

As a set of optimization methods, direct search methods (zero-order methods), gradient methods (first-order methods), and Newtonian methods (second-order methods) are used. These methods have a large number of settings options, thereby providing the ability to quickly adapt the system to any process [1, 2].

The report outlines possible principles for managing the package of optimization programs when solving a specific applied problem $p_i\left(x\right)\in P$, allowing to increase the overall efficiency of solving the problem by combining programs in the process of solving the problem itself using a multiprocessor (multicore) computing system.

The principle of sequential implementation with a single-core (sequential) architecture has an important independent meaning and can be considered as a basic block of implementations for multiprocessor or multicore architectures. Let us describe one of the principles of possible schemes for implementing an algorithm for solving optimization problems on such architectures.

Let $M_1, M_2, ..., M_k$ be a list of optimization methods made up of algorithms in the unconstrained optimization software package. It is advisable to include methods of different characteristics in the list if, generally speaking, nothing is known about the structure of the objective function. The process of solving the problem is carried out in stages, each of which consists of training and work steps. The first of these stages is designed to identify a locally efficient algorithm from the available list of algorithms. After that, a working step is carried out, which consists in solving the problem using only the most efficient algorithm that was identified at the first stage. Both the training and working steps are given certain time slices. At the training stage, you can use the following two options:

i. To determine the local efficiency of methods, optimization is carried out from the same point x^0. In this case, there is a somewhat uneconomical consumption of machine time, and training is used only to identify a locally effective algorithm;
ii. The training time is used not only to find a locally effective algorithm, but also to advance to a minimum point, since for training each next algorithm, not the initial, but the current point is used.

At the training stage, all algorithms from the initial list $M_1, M_2, ..., M_k$ are given the opportunity to prove themselves during a given initial time slice. The only exceptions are those methods that turned out to be the least effective twice in a row at the learning step. They are not given a time slice and are temporarily excluded from the list. The following formula is used to calculate the values of the local efficiencies of the methods:

$$E_i = \frac{\left| f\left(x^{k+1}\right) - f\left(x^k\right) \right|}{\left| f\left(x^k\right) \right| + \varepsilon} + \frac{\left\| x^{k+1} - x^k \right\|}{\left\| x^k \right\| + \varepsilon}. \qquad (1)$$

Here E_i is the local efficiency of the i^{th} algorithm; x^{k+1} and x^k are the start and end points obtained by applying the i^{th} algorithm; $f\left(x^{k+1}\right)$ and $f\left(x^k\right)$ are the values of the objective function at these points; $\|\cdot\|$ is the Euclidean norm; ε is a small positive number.

The criterion for exiting the proposed procedure is the fulfillment of the search termination condition for all methods. In the end, the user receives the accumulated information about the search progress, which includes the optimal chain of methods that worked at the working steps, the total time to find a solution, as well as the values of the objective function, coordinates and local efficiencies obtained at the training stages. Next, we consider the principle of parallel implementation in a multi-core architecture. The simplest implementation of the multithreaded version of the solution to the posed optimization problem is an approach in which the threads independently perform the operations of the sequential algorithm described above.

The solution to the unconditional optimization problem is carried out in stages. At each stage, the following actions (steps) are carried out:

i. At the initial step, from the list of all available unconditional optimization algorithms $M_1, M_2, ..., M_k$, several algorithms $M_{s_1}, M_{s_2}, ..., M_{s_N}$ are randomly selected, the total number, N, of which is chosen equal to the number of cores installed in the system.
ii. At the stage of the working step, the most efficient algorithms are identified. The duration, T_i, of the work step

can be increased if any method is found to be most effective in several successive steps.

iii. The current values of the local efficiencies E_i of the methods are calculated using the formula (1). Half of the working algorithms that have found the lowest efficiency are excluded from the list.

iv. The list of working algorithms is supplemented with the same number of other algorithms as excluded in the previous step, and steps 2-4 are repeated.

When working with automatic and dialog systems, the user, in accordance with standard requirements, formalize the optimization problem in any programming language in the form of a module (library file with the .dll extension), enters it into the system by specifying the full path to the file of the created library; using directives (commands) calls the most suitable (from his/her point of view) algorithms from the library of modules, selects their parameters. The managing program organizes the interaction of modules from the body of the package, manages the input of initial and current information, interprets user directives, dynamically loads optimization modules into the computer's memory, displays the computation results in the form prescribed by the user on the display screen (you can simultaneously receive results on a printer).

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

References.

1. Евтушенко Ю. Г. Методы решения экстремальных задач и их применение в системах оптимизации. – М.: Наука, 1982. [English translation: Yevtushenko Y.G. Methods of solution to extremum problems and their application in optimization systems. – Moscow: Nauka, 1982.]
2. Айда-заде К. Р., Сидоренко Н. С. Об одном подходе к построению комбинированных алгоритмов оптимизации // Техническая кибернетика. – 1982. – №6. – С.87-93. [English translation: Aida-zade K. R., Sidorenko N. S. On one approach to the construction of combined optimization algorithms // Technical cybernetics. – 1982. – no.6. – p.87-93].