Modelling of the process of a program compilation according to the training sample data using the collective intelligence of the simplest computers

V Protasov and Z Potapova*

Department of Mathematical Cybernetics, Moscow Aviation Institute (National Research University), 4 Volokolamskoe Highway, 125993, Moscow, Russian Federation

*E-mail: potapovaze@mai.ru

Abstract. The description of the system intended for genetic programming and consisting of a large number of elementary computers is given. Commands and data are placed in a cell consisting of four fields – the first address field, the second address field, the arithmetic field and the result field. This architecture made it possible to place a software binary tree formed by commands and addresses into sequentially located cells of an elementary computer. This technique made it possible to simplify the crossover and mutation operators, as well as the coding of the problem without involving complex programming systems and supercomputers. In the future, it is planned to use table processors tested on complex problems of genetic programming to significantly reduce power consumption in on-board computers of aviation and space technology.

1. Introduction

Currently, modern production planning tools allow you to describe entire factories in the form of electronic models [1-3]. They create the necessary prerequisites for the large-scale implementation of genetic programming (GP) methods.

John Koza is a pioneer in the use of genetic programming to optimize complex problems. He developed a general approach for the formal description of various subject areas. This approach allows us to search for ways to improve real technical systems using GP methods. J. Koza proposed to create robots that can make functional copies of artificial organisms. Several original solutions have already been found. Two of them are already protected by patents. For example, a new method for determining the optimal configuration of wire antennas with the required electromagnetic characteristics; as well as the design of the wings of large aircraft, which aroused the interest of Boeing [4].

Extremely high energy requirements are the main constraint on computing resources to commercialize these methods. Similar problems of reducing energy costs were discussed in the works [5-7]. These problems arise, for instance, when carrying out evolutionary computations for problems of planning a directed acyclic graph (DAG) using multiprocessor systems [5]. A systematic review of the literature on the problem of scheduling multiprocessor tasks is given in this work. It is noted that this issue has attracted significant attention over the past three decades.
The complex organization of computational algorithms is the second problem in GP. Programming languages such as LISP [8,9] or PROLOGUE [10] are traditionally used to describe and manipulate trees. The programming costs often exceed the computational costs of these programs.

A wide range of research is also focused on the development of evolutionary algorithms. These works relate, for example, to topics such as characteristics of problems, heterogeneity of the environment and optimization criteria [5]. Many fields of application for evolutionary algorithms exist in multiprocessor systems. A hybrid genetic approach is presented, for example, in [6]. This approach helps to distribute and schedule the parallelization of computational processes on a multiprocessor computer. A system of real-time tasks with balancing the load on processors is considered in work. In addition, “the proposed hybrid genetic approach has better results than the classical GAs in terms of load balancing, minimum response time and good flexibility”. An implementation method for solving synchronization problems in multiprocessor systems is given in [7]. This method is called the “Dynamic Genetic Algorithm for Real-Time Planning” (GA-RTS). “A significant feature of GA-RTS is that it can handle dynamic as well as static scheduling of inter-dependent tasks for real-time systems”. Generally speaking, such schedules for the operation of multiprocessor systems for the parallelization of tasks to be solved are drawn up using powerful software and hardware resources of third-party computers. Most often, supercomputers and / or cloud computing are used for this.

We propose to build specialized multiprocessor systems based on table processors with a command system that provides a simple implementation of software trees to significantly reduce energy costs for GP. The construction of specialized computing systems is one of the possible ways to solve these problems. These systems consist of a large number of elementary computers created on the basis of table processors. They replace floating point calculations with calculations using tables of logarithms and antilogarithms. This approach reduces energy consumption for computations by an order of magnitude [11]. Table processors tested on such a complex task as genetic programming can be widely used in on-board computers of aviation and space technology for solving a wide class of problems. This is the topic of our further research.

The architecture of elementary computers (EC) should be adapted to work with trees immediately. This will greatly simplify the programming of tasks and reduce the cost of computations due to a significant reduction in power consumption. A feature of this work that distinguishes it from the rest is that evolutionary computations are performed in the system itself using simple tree-like structures implemented in a sequential structure of the simplest two-address commands. The application of this approach makes it possible to significantly simplify the programming of complex problems of genetic programming and other types of evolutionary computations, using the software and hardware of the multiprocessor systems themselves.

The novelty of this work lies in the use of the multiprocessor systems themselves for genetic programming of planning problems. Their resources are proposed to be used both for creating a primary population of individual programs and for executing genetic programming operators – mutation, crossing and selection.

2. Formulation of the problem and suggested solution methods

2.1. Formulation of the problem
Let a function $F$ be given on the set of real numbers, connecting $n$ arguments $X_i (i=1,2,3,…,n)$ and $m$ constants $C_j (j=1,2,3,…,m)$ by the arithmetic operations of addition, subtraction and multiplication. Also, a table is given, consisting of $k$ rows containing elements of the known values and the values of the function calculated for them. Suppose that the function $F$ has been “lost” and it is required to find an algorithm for recovering this function from a given set of values using a toroidal matrix of elementary computers. For small $n$ and $m$, the problem is solved using personal computers. Problems of medium and high difficulty are solved with varying success using powerful software systems on supercomputers, on computer clusters, or using cloud computing.
2.2. Suggested solution methods

2.2.1. The graph of connection of the system of elementary computers and the system of commands.

Let us consider ways of solving such problems using a system which consists of a large number of ECs with a special architecture and which uses the rules of interactions taken from the GA.

To connect ECs to each other, various architectures can be used [11], including homogeneous computing environments [12]. Our research was carried out using a computer model of a specialized computing system in the form of a toroidal EC matrix with a size of 10×10.

Figure 1 shows a similar communication graph of an EC matrix with a size of 5×5. The vertex numbers correspond to an ordered pair of EC numbers for each coordinate, and to provide cyclic links, the last EC is connected to the first.

Figure 1. Communication graph of a group of elementary computers.

To test the possibility of solving GP problems using such a system in the Delphi environment, a computer model was built [13]. The basic set of operations included addition, subtraction and multiplication. For testing, a training sample was taken, placed in table 1.

Table 1. Test training sample.

| X1 | X2 | X3 | Y   |
|----|----|----|-----|
| 1  | 2  | 3  | 3   |
| 2  | 3  | 3  | 4   |
| 3  | 3  | 4  | 10  |
| 4  | 4  | 10 | 26  |
| 5  | 10 | 26 | 88  |

This sampling is generated by the function \( Y = X1 \times X3 - 2 \times X2 + 4 \) for five sets of different combinations of arguments \( X1, X2, X3 \).

EC memory cell under number \( i \) consists of four fields \( A1[i], D[i], A2[i] \) – numbers of type integer and \( P[i] \) – of type real. Each operator of the program tree is modeled by one command \( P[i] = P[A1[i]] \ | D[i] | P[A2[i]] \), where \( D[i] \) takes values from the set \{1; 2; 3\} and corresponds to the signs of the arithmetic operations of addition \{1\}, subtraction \{2\} and multiplication \{3\}, and \( A1[i] \) and \( A2[i] \) are the addresses of the first and second operands.
2.2.2. Description of the computer model of the system and simulation results. The complete binary tree of the program being restored is shown in figure 2. In the fourth fields of the first three cells of the program, the sample lines with data X1, X2, X3 are entered in order. In the fourth fields from the fourth to the eighth cells of the EC, random constants C1-C5 are entered, taking integer values from -5 to 5. Further, in the first three fields from the ninth to the fifteenth cells, seven random two-address commands are formed. In the fourth fields from the ninth to the fourteenth cells, the intermediate results of calculations will be entered, and the result of the program will be entered in the fourth field of the fifteenth cell. When generating a program, a simple rule is fulfilled: operands of the i-th instruction can only be operands with addresses from 1 to i-1.

![Figure 2. One of the implementations of the program tree.](image)

Such a simple rule allows to avoid many complex procedures used by developers of software shells. For example, it could be removing unnecessary program codes (the so-called introns).

Further, each EC executes a random program formed by it, makes an assessment of the specimen - calculates the sum of the squares of the differences between the results obtained according to the program and taken from the training sample (the so-called fitness function in GA). According to the given communication graph, it exchanges programs and constants generated by it with neighboring ECs. In accordance with the rules of interaction, the EC crosses randomly selected pairs of someone else's and his own program using one-point crossing over, calculates the fitness function and keeps the best program for itself. With a predetermined probability, which is a tuning parameter of the system, the EC makes a mutation in the constants or the commands, and calculates the fitness function for the mutated specimen.

The diagram of the crossing and mutation operators is shown in figure 3. The execution order of the operators of the GP is shown in numbers.

![Figure 3. Crossover and mutation operators diagram.](image)
Then, the processes of exchange, crossing over, mutation and natural selection continue until a specimen appears with a fitness function value equal to zero. Thus, within the limits of the binary tree, the EC system finds the required sequence of commands, the number and the values of the constants that satisfy the training sampling. For the test training sampling presented above, emulated into a personal computer (PC) by a toroidal EC matrix, more than ten thousand program restorations from various starting positions were carried out. The collective EC system has demonstrated stable results of the restoration of the “lost” function. One of these results is shown in table 2.

Table 2. An example of restoring a “lost” function.

| Cell i | A1[i] | D[i] | A2[i] | P[i] |
|--------|-------|------|-------|------|
| Field 1 | Field 2 | Field 3 | Field 4 |
| 1 | 0 | 0 | 0 | X1 |
| 2 | 0 | 0 | 0 | X2 |
| 3 | 0 | 0 | 0 | X3 |
| 4 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | –6 |
| 7 | 0 | 0 | 0 | 4 |
| 8 | 0 | 0 | 0 | 2 |
| 9 | 7 | 2(−) | 2 | 4−X2 |
| 10 | 4 | 1(+) | 2 | X2 |
| 11 | 2 | 2(−) | 4 | X2 |
| 12 | 3 | 2(−) | 7 | X3−4 |
| 13 | 3 | 3(*) | 1 | X1*X3 |
| 14 | 13 | 2(−) | 2 | X1*X3−X2 |
| 15 | 14 | 1(+) | 9 | Y=X1*X3−2*X2+4 |

3. Verification of the correctness of the resulting GP scheme

In [14] the results of the analytical construction of the mathematical model are presented. We compared them with the results obtained using the GPU circuit to further check the correctness of the resulting circuit.

In [15], the method of the evolutionary decision reconciliation (EDR) is described. It is a new method of making collective decisions by a group of experts, based on the application of interaction rules taken from the genetic algorithms.

Let us consider a modification of this method. Each expert acts in two roles: as a generator of decisions or their parts at the first stage (the stage of generation) and as an evaluator of the other people’s decisions at the second stage (the stage of reconciliation). If the case is especially difficult, the experts may answer “I do not know”. Suppose the group consists of M experts and they are assigned a task that they cannot solve alone.

Let us introduce the notation. Let \( G_R \) be the average probability that an expert would offer the correct decision; \( G_V \) – the average probability that an expert would answer “I don’t know”. Consequently, \( M \times G_R \) is the number of the experts who would answer “I do not know”. At the second stage (the stage of reconciliation) the experts who said “I do not know” are provided with all the solutions, correct and incorrect, obtained at the generation stage.

Let us assume that they will choose correct decisions from this list with probability \( E_R \).

Let the average probability of an expert making the correct decision at the second stage in the group equals \( P_R \). It is clear that \( P_R \geq G_R \) and \( P_R \) depends on \( M, G_R, E_R, G_V \). It is required to find this dependence.

Let's find it using the GP. Let's apply the Monte Carlo method to the computer model that simulates the work of a group of experts who are using the EDR method. Let us construct the training
sample from various random combinations of initial values and match them with the $P_R$ values calculated by the Monte Carlo method.

For simplicity, let $M = 3$. The training sample is shown in the table 3. The values of the probabilities are calculated with an accuracy of seven significant digits.

**Table 3.** Training sample for GP.

| $E_R$  | $G_R$  | $G_V$  | $P_R$  |
|-------|-------|-------|-------|
| X1    | X2    | X3    | Y     |
| 0.99631576 | 0.99503320 | 0.00135852 | 0.99638668 |
| 0.83201839 | 0.78583498 | 0.06913842 | 0.84072097 |
| 0.40131234 | 0.33181223 | 0.35434369 | 0.41052488 |
| 0.19781611 | 0.15446527 | 0.59868766 | 0.18822624 |
| 0.64565631 | 0.50000000 | 0.35434369 | 0.67158818 |
| 0.40131234 | 0.26894142 | 0.59868766 | 0.64565631 |
| 0.19781611 | 0.11920292 | 0.80218389 | 0.05448728 |
| 0.08317270 | 0.04742587 | 0.91682730 | 0.01910033 |
| 0.03229546 | 0.01798621 | 0.96770454 | 0.01910033 |

This sample was entered into the GP program consisting of 31 cells. At the first launch, the sought dependence was restored in the form

$$P[31] = X1 \times X2 \times X3 \times (2 - X2) + X2$$

and at the second launch

$$P[31] = X1 \times X2 \times X3 \times (2 - X2) + X2$$

The restored function differed from the original one in both cases starting from the seventh significant digit. Thus, we have obtained an expression as a result of restoring the desired function after changing the notation of variables:

$$P_R = G_R + E_R \times G_V \times (2 - G_R)$$

Taking into account that the restoration of the function was carried out at $M = 3$, we obtain the final expression for the desired function as follows:

$$P_R = G_R + E_R \times G_V \times (1 - (1 - G_R)^{M-1})$$

which is fully consistent with the analytical conclusion of [11].

Let the group consist of $M$ experts. To obtain probability $Q_R$ of the correct decision by a majority of votes, we apply the Condorcet formula [16] and obtain the complete model of this EDR method type:

$$Q_R = \sum_{i=0}^{M-1} C_M^i p_R^i (1 - p_R)^{M-i}$$

4. Conclusion

We can conclude that when using a collective intelligence system that works according to the rules of interaction taken from genetic algorithms, the classical GP problem of compiling a program code for training sampling is solved not only on supercomputers using the most complex software systems, but also on a matrix of the simplest EC. The results obtained are of fundamental importance, since they demonstrate the efficiency of the basic principles underlying the collective intelligence systems when solving a complex GP problem with the use of interaction rules taken from the genetic algorithms. It can also be concluded that the proposed approach to replacing floating point calculations with tabular calculations can be applied to significantly reduce the consumption of electrical energy in the power
supplies of on-board computers of aviation and space technology systems, in which reducing the weight of equipment is of fundamental importance.

References
[1] Poli R and Koza J 2005 Genetic Programming Search Methodologies eds E Burke et. al. (New York: Springer US) chapter 5 pp 127–164
[2] Koza J 2010 Human-competitive results produced by genetic programming. Genet. Program. Evolv. M. 11 251 doi:10.1007/s10710-010-9112-3
[3] Segaran T 2007 Programming Collective Intelligence: Building Smart Web 2.0 Applications (New York: O'Reilly Media) p 360
[4] Koza J, Andre D, Bennett F and Keane M, U.S. Patent No. 5,867,397 (19 March 2002)
[5] Silva E and Gabriel P 2020 A Comprehensive Review of Evolutionary Algorithms for Multiprocessor DAG Scheduling. Computation 8(2) 26 https://doi.org/10.3390/computation8020026
[6] Gharbi I, Gharsellaoui H and Bouamama S 2019 A New Hybrid Genetic Algorithm-based Approach for Critical Multiprocessor Real-Time Scheduling with Low Power Optimization. Procedia Comput. Sci. 159 1547 https://doi.org/10.1016/j.procs.2019.09.325
[7] My Muhuri P, Rauniyar A and Nath R 2019 On arrival scheduling of real-time precedence constrained tasks on multi-processor systems using genetic algorithm. Future Gener. Comput. Syst. 93 702 https://doi.org/10.1016/j.future.2018.10.013
[8] Mikelsons M 1983 Interactive program execution in Lispedit. ACM SIGPLAN Notices 18(8) 71 doi:10.1145/1006142.1006164
[9] Monnie S and Sperber M 2020 Evolution of Emacs Lisp. Proc. ACM Program. Lang. 1 74 https://doi.org/10.1145/3386324
[10] Clocksin W and Mellish S 2005 Programming in Prolog. Using the ISO Standard. Theor. Pract. Log. Prog. 5(03) 391 doi: 10.1017/S1471068405212449
[11] Zatuliveter Y and Fishchenko E 2018 About the universal algorithmic space of distributed and parallel computing. Proc. Conf. 11th Manag. Large-Scale Syst. Develop. (Moscow: IEEE) p 5
[12] Khoroshevsky V, Mamoilenko S, Kurnosov M and Medvedeva N 2006 Space-distributed multi-cluster computer system for training in parallel computational technologies. Computer Science: International Workshop and Tutorials on Electron Devices and Materials (Russia: Novosibirsk) p 218
[13] Protasov V 2009 Construction of the Metasystem Transitions (Moscow: Physicotechnical Informatics Institute) p 186
[14] Protasov V, Potapova Z and Akhobadze G 2020 How to reduce the probability of erroneous decisions in the systems of collective intelligence. Conf. Ser.: Mater. Sci. Eng. 927(1) 012069 doi: 10.1088/1757-899X/927/1/012069
[15] Protasov V, Potapova Z and Melnikov E 2014 Method for evolutionary decision reconciliation, and expert theorems. The 3d Int. Conf. on Intell. Syst. Appl. Think Mind 3(30) 70074
[16] Austen-Smith D and Banks J 1996 Information Aggregation, Rationality and the Condorcet Jury Theorem. Am. Political Sci. Rev. 90(1) p34 https://doi.org/10.2307/2082796